

# House-Prices

June 18, 2025

```
[2]: import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

test = pd.read_csv("D:/ProjectsKaggle/house/test.csv")
train = pd.read_csv("D:/ProjectsKaggle/house/train.csv")
sample = pd.read_csv("D:/ProjectsKaggle/house/sample_submission.csv")
pd.set_option('display.max_columns', None)

print("Number of rows and columns:")
print(train.shape)
print("All rows and columns")
train.info()

print("Descriptive statistics for quantitative columns for train")
print(train.describe().head())
print("All missing values in columns")
print("We create a DF that has 3 columns. The first is the number of missing_
    ↳ values, the second is the percentage of missing values, and the third is the_
    ↳ data types.")
print("missing_values - Shows the number of missing values")
print("missing_percentage - Shows in percentage how many values are missing in_
    ↳ a column")
print("missing_data - creating DF")
missing_values = train.isnull().sum().sort_values(ascending=False)
missing_percentage = (train.isnull().sum() / len(train) * 100).
    ↳ sort_values(ascending=False)
missing_type = train.dtypes
missing_data = pd.concat([missing_values, missing_percentage, missing_type],_
    ↳ axis=1, keys=["Total Missing for train", "Percentage for train", "Type for_
    ↳ train"])
print("Missing Value Columns for train: ")
print(missing_data[missing_data["Total Missing for train"] > 0])

print("Descriptive statistics for quantitative columns for test")
print(test.describe().head())
```

```

missing_values_test = test.isnull().sum().sort_values(ascending=False)
missing_percentage_test = (test.isnull().sum() / len(test) * 100).
    ↪sort_values(ascending=False)
missing_type_test = test.dtypes
missing_data_test = pd.concat([missing_values_test, missing_percentage_test,
    ↪missing_type_test], axis=1, keys=["Total Missing for test", "Percentage for",
    ↪test", "Type for test"])
print("Missing Value Columns for test: ")
print(missing_data_test[missing_data_test["Total Missing for test"] > 0])

print("Replacing empty values with mode and mean values for train")
mode_columns_train = ["GarageQual", "GarageFinish", "GarageType", "GarageCond",
    ↪"BsmtFinType2", "BsmtExposure", "BsmtCond", "BsmtQual",
    ↪"BsmtFinType1", "Electrical"]
for column in mode_columns_train:
    train[column] = train[column].fillna(train[column].mode()[0])

mean_columns_train = ["MasVnrArea", "GarageYrBlt"]
for column in mean_columns_train:
    train[column] = train[column].fillna(train[column].mean())

print("Saving statistics from train")
train_modes = {col: train[col].mode()[0] for col in mode_columns_train}
train_means = {col: train[col].mean() for col in mean_columns_train}

print("Replacing empty values with mode and mean values for test")
mode_columns_test = ["GarageQual", "GarageFinish", "GarageType", "GarageCond",
    ↪"Utilities", "Functional", "Exterior1st", "Exterior2nd",
    ↪"BsmtFinType2", "BsmtExposure", "BsmtCond", "BsmtQual",
    ↪"BsmtFinType1", "MSZoning", "SaleType", "KitchenQual"]
for column in mode_columns_test:
    if column in test.columns:
        test[column] = test[column].fillna(train_modes.get(column, test[column].
    ↪mode()[0]))

print("For test we use the mean value or mode, where the gaps are 10%<x<25%")
mean_columns_test = ["MasVnrArea", "GarageYrBlt", "BsmtHalfBath",
    ↪"BsmtFullBath", "BsmtFinSF1", "TotalBsmtSF", "BsmtUnfSF", "BsmtFinSF2",
    ↪"GarageCars", "GarageArea", "LotFrontage"]
for column in mean_columns_test:
    test[column] = train[column].fillna(train[column].mean())
for column in mean_columns_test:
    if column in test.columns:
        fill_value = train_means.get(column, test[column].mean())
        test[column] = test[column].fillna(fill_value)

print("Using interpolation to replace empty values where gaps are 10%<x<25%")

```

```

train["LotFrontage"] = train["LotFrontage"].interpolate()

print("Removing empty values that have a gap value greater than 45% for train")
drop_columns_train = ["PoolQC", "MiscFeature", "Alley", "Fence", "MasVnrType",
    ↪ "FireplaceQu"]
train = train.drop(columns=drop_columns_train)

print("Removing empty values that have a skip value greater than 45% for test")
drop_columns_test = ["PoolQC", "MiscFeature", "Alley", "Fence", "MasVnrType",
    ↪ "FireplaceQu"]
test = test.drop(columns=drop_columns_test)

print("Let's see if there are any duplicate lines for train")
print(train.duplicated().sum())

print("Let's see if there are any duplicate lines for test")
print(test.duplicated().sum())

print("Statistics for SalePrice")
print(train["SalePrice"].describe())

print("Graph without normalization")
plt.figure(figsize=(10, 6))
sns.histplot(train['SalePrice'], kde=True, bins=50)
plt.title("SalePrice Scatter")
plt.xlabel("Price (USD)")
plt.ylabel("Frequency")
plt.show()

print("Normalization. Logarithmization.")
train["SalePrice_log"] = np.log1p(train["SalePrice"])
plt.figure(figsize=(10, 6))
sns.histplot(train['SalePrice_log'], kde=True, bins=50, color='green')
plt.title("Log-Transformed SalePrice Scatter")
plt.xlabel("Log(Price)")
plt.ylabel("Frequency")
plt.show()
print("After normalization, it can be seen that the data is centered, which
    ↪ will improve the performance of the model.")

print("Creating a heat map to find correlation")
numerical_cols = train.select_dtypes(include=["int64", "float64"]).columns.
    ↪ tolist()
print(numerical_cols)

corr_matrix = train[numerical_cols].corr()
k = 12

```

```

corr_top15 = corr_matrix.nlargest(k, "SalePrice")["SalePrice"].index
cm = np.corrcoef(train[corr_top15].values.T)

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, square=True, fmt=".2f",
            annot_kws={"size": 10}, yticklabels=corr_top15.values,
            xticklabels=corr_top15.values,
            cmap="viridis")
plt.title("SalePrice Most Correlated Feature")
plt.show()

print("Identifying outliers for train")
print("First, you need to plot a graph and make sure that there are a lot of
    emissions and they are not necessary. For example, a house can really be
    expensive.")
plt.figure(figsize = (10,6))
sns.boxplot(x = train["OverallQual"], color = "green")
plt.title("OverallQual with outliers")
plt.show()
plt.figure(figsize = (10,6))
sns.boxplot(x = train["GrLivArea"], color = "green")
plt.title("GrLivArea with outliers")
plt.show()
plt.figure(figsize = (10,6))
sns.boxplot(x = train["GarageArea"], color = "green")
plt.title("GarageArea with outliers")
plt.show()

print("Identifying outliers for test")
plt.figure(figsize = (10,6))
sns.boxplot(x = test["OverallQual"], color = "green")
plt.title("OverallQual with outliers")
plt.show()
plt.figure(figsize = (10,6))
sns.boxplot(x = test["GrLivArea"], color = "green")
plt.title("GrLivArea with outliers")
plt.show()
plt.figure(figsize = (10,6))
sns.boxplot(x = test["GarageArea"], color = "green")
plt.title("GarageArea with outliers")
plt.show()

print("Histogram before outliers for GrLivArea for train")
sns.histplot(train['GrLivArea'], kde=True, color='orange')
plt.title("Before Handling Outliers for train")
plt.show()
print("Histogram before outliers for GarageArea for train")

```

```

sns.histplot(train['GarageArea'], kde=True, color='orange')
plt.title("Before Handling Outliers for train")
plt.show()

print("Histogram before outliers for GrLivArea for test")
sns.histplot(test['GrLivArea'], kde=True, color='orange')
plt.title("Before Handling Outliers for test")
#plt.show()
print("Histogram before outliers for GarageArea for test")
sns.histplot(test['GarageArea'], kde=True, color='orange')
plt.title("Before Handling Outliers for test")
plt.show()

print("IQR method for data that has many outliers for train")
print("If the values go beyond lower_bound = Q1 - 1.5 * IQR and upper_bound = Q3 + 1.5 * IQR - these are outliers")

Q1 = train["GrLivArea"].quantile(0.25)
Q3 = train["GrLivArea"].quantile(0.75)
IQR = Q3 - Q1
train = train[(train["GrLivArea"] >= Q1 - 1.5*IQR) & (train["GrLivArea"] <= Q3 + 1.5*IQR)]

Q1 = train["GarageArea"].quantile(0.25)
Q3 = train["GarageArea"].quantile(0.75)
IQR = Q3 - Q1
train = train[(train["GarageArea"] >= Q1 - 1.5*IQR) & (train["GarageArea"] <= Q3 + 1.5*IQR)]

print("IQR method for data that have many outliers for test")
test["GrLivArea"] = test["GrLivArea"].clip(lower=Q1 - 1.5*IQR, upper=Q3 + 1.5*IQR)

print("Resetting indices for train")
train = train.reset_index(drop=True)

print("Resetting indices for test")
test = test.reset_index(drop=True)

print("Histogram without outliers for GrLivArea for train")
sns.histplot(train["GrLivArea"], kde=True, color='red')
plt.title("After Handling Outliers for train")
plt.show()

print("Outlier-free histogram for GarageArea for train")
sns.histplot(train["GarageArea"], kde=True, color='red')
plt.title("After Handling Outliers for train")

```

```

plt.show()

print("Histogram without outliers for GrLivArea for test")
sns.histplot(test["GrLivArea"], kde=True, color='red')
plt.title("After Handling Outliers for test")
plt.show()

print("Histogram without outliers for GarageArea for test")
sns.histplot(test["GarageArea"], kde=True, color='red')
plt.title("After Handling Outliers for test")
plt.show()

print("Using Categorical Data for Analysis. Neighborhood for train")
print("Data processing")
train["Neighborhood"] = train["Neighborhood"].str.lower().str.strip()
print("One-hot encoding")
df_encoded = pd.get_dummies(train, columns=["Neighborhood"], drop_first=True)
neighborhood_columns = [col for col in df_encoded.columns if col.
    ↳startswith('Neighborhood_')]

print(df_encoded.head())

print("Using Categorical Data for Analysis. Neighborhood for test")
test["Neighborhood"] = test["Neighborhood"].str.lower().str.strip()
print("One-hot encoding")
df_encoded_t = pd.get_dummies(test, columns=["Neighborhood"], drop_first=True)
for col in neighborhood_columns:
    if col not in df_encoded_t.columns:
        df_encoded_t[col] = 0
print(df_encoded_t.head())

print("In which area are houses more expensive?")
plt.figure(figsize=(12,6))
sns.boxplot(x="Neighborhood", y='SalePrice', data=train)
plt.xticks(rotation=45)
plt.title("SalePrice by Neighborhood")
plt.show()

print("Creating a new df for the model for train")
numeric_features = train[["OverallQual", "GrLivArea", "GarageArea"]]
neighborhood_features = df_encoded.filter(regex='^Neighborhood_')
all_features = pd.concat([numeric_features, neighborhood_features], axis=1)
print(all_features.head())

print("Creating a new df for the model for test")
numeric_features_t = test[["OverallQual", "GrLivArea", "GarageArea"]]
neighborhood_features_t = df_encoded_t.filter(regex='^Neighborhood_')

```

```

all_features_t = pd.concat([numeric_features_t, neighborhood_features_t],
    ↪axis=1)
print(all_features_t.head())

print("Column alignment")
all_features_t = all_features_t[all_features.columns]
print(all_features_t.head())

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(all_features,
    ↪train["SalePrice_log"], test_size = 0.2, random_state = 42)

print("Model training")
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
print("Linear")
lr = LinearRegression()
lr.fit(x_train, y_train)
y_pred_lr = lr.predict(x_test)
print("RandomForestRegressor")
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(x_train, y_train)
y_pred_rf = rf.predict(x_test)

print("Metrics for linear regression. Log")
rmse_lr = np.sqrt(mean_squared_error(y_test, y_pred_lr))
r2_lr = r2_score(y_test, y_pred_lr)
print("RMSE", rmse_lr)
print("R2", r2_lr)

print("Metrics for RandomForestRegressor. Log")
rmse_rf = np.sqrt(mean_squared_error(y_test, y_pred_rf))
r2_rf = r2_score(y_test, y_pred_rf)
print("RMSE", rmse_rf)
print("R2", r2_rf)

y_test_original = np.expml(y_test)

print("Metrics for linear regression.")
y_pred_lr_original = np.expml(y_pred_lr)
rmse_lr_original = np.sqrt(mean_squared_error(y_test_original,
    ↪y_pred_lr_original))
r2_lr_original = r2_score(y_test_original, y_pred_lr_original)
print("RMSE", rmse_lr_original)
print("R2", r2_lr_original)

```

```

print("Metrics for RandomForestRegressor.")
y_pred_rf_original = np.expm1(y_pred_rf)
rmse_rf_original = np.sqrt(mean_squared_error(y_test_original,
    ↪ y_pred_rf_original))
r2_rf_original = r2_score(y_test_original, y_pred_rf_original)
print("RMSE", rmse_rf_original)
print("R2", r2_rf_original)

print("Cross-validation. Mean. Standard deviation. Linear model")
from sklearn.model_selection import cross_val_score
scores_lr = cross_val_score(lr, all_features, train["SalePrice_log"], cv = 3)
mean_lr = scores_lr.mean()
std_lr = scores_lr.std()
print("Average RMSE", mean_lr, "Standard Deviation", std_lr)

print("Cross-validation. Average. Standard Deviation. Random Forest")
scores_rf = cross_val_score(rf, all_features, train["SalePrice_log"], cv=3)
rmse_scores_rf = -scores_rf
mean_rf = rmse_scores_rf.mean()
std_rf = rmse_scores_rf.std()
print("Average RMSE:" , mean_rf, "Standard Deviation", std_rf)

print("Comparison of models")
models = ["Linear Regression", "Random Forest"]
rmse_values = [rmse_lr_original, rmse_rf_original]

plt.figure(figsize=(10, 6))
sns.barplot(x=models, y=rmse_values, hue=models, palette="viridis",
    ↪ legend=False)
plt.title("Comparison of models by RMSE")
plt.ylabel("RMSE (original scale)")
plt.show()

print("Train the best model on all data")
best_model = RandomForestRegressor(n_estimators=100, random_state=42)
best_model.fit(all_features, train["SalePrice_log"])

print("Predictions for test data")
test_predictions_log = best_model.predict(all_features_t)
test_predictions = np.expm1(test_predictions_log)

print("Creating a file")
submission = pd.DataFrame({
    "Id": test["Id"],
    "SalePrice": test_predictions
})
submission.to_csv('D:/ProjectsKaggle/house/submission.csv', index=False)

```



```

print("Visualization of Predictions")
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.histplot(train["SalePrice"], bins=50, kde=True, color="blue")
plt.title("Actual prices(train)")

plt.subplot(1, 2, 2)
sns.histplot(submission["SalePrice"], bins=50, kde=True, color="red")
plt.title("Predicted prices(test)")

plt.tight_layout()
plt.show()

```

Number of rows and columns:

(1460, 81)

All rows and columns

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1460 entries, 0 to 1459

Data columns (total 81 columns):

#	Column	Non-Null Count	Dtype
0	Id	1460 non-null	int64
1	MSSubClass	1460 non-null	int64
2	MSZoning	1460 non-null	object
3	LotFrontage	1201 non-null	float64
4	LotArea	1460 non-null	int64
5	Street	1460 non-null	object
6	Alley	91 non-null	object
7	LotShape	1460 non-null	object
8	LandContour	1460 non-null	object
9	Utilities	1460 non-null	object
10	LotConfig	1460 non-null	object
11	LandSlope	1460 non-null	object
12	Neighborhood	1460 non-null	object
13	Condition1	1460 non-null	object
14	Condition2	1460 non-null	object
15	BldgType	1460 non-null	object
16	HouseStyle	1460 non-null	object
17	OverallQual	1460 non-null	int64
18	OverallCond	1460 non-null	int64
19	YearBuilt	1460 non-null	int64
20	YearRemodAdd	1460 non-null	int64
21	RoofStyle	1460 non-null	object
22	RoofMatl	1460 non-null	object
23	Exterior1st	1460 non-null	object
24	Exterior2nd	1460 non-null	object

25	MasVnrType	588 non-null	object
26	MasVnrArea	1452 non-null	float64
27	ExterQual	1460 non-null	object
28	ExterCond	1460 non-null	object
29	Foundation	1460 non-null	object
30	BsmtQual	1423 non-null	object
31	BsmtCond	1423 non-null	object
32	BsmtExposure	1422 non-null	object
33	BsmtFinType1	1423 non-null	object
34	BsmtFinSF1	1460 non-null	int64
35	BsmtFinType2	1422 non-null	object
36	BsmtFinSF2	1460 non-null	int64
37	BsmtUnfSF	1460 non-null	int64
38	TotalBsmtSF	1460 non-null	int64
39	Heating	1460 non-null	object
40	HeatingQC	1460 non-null	object
41	CentralAir	1460 non-null	object
42	Electrical	1459 non-null	object
43	1stFlrSF	1460 non-null	int64
44	2ndFlrSF	1460 non-null	int64
45	LowQualFinSF	1460 non-null	int64
46	GrLivArea	1460 non-null	int64
47	BsmtFullBath	1460 non-null	int64
48	BsmtHalfBath	1460 non-null	int64
49	FullBath	1460 non-null	int64
50	HalfBath	1460 non-null	int64
51	BedroomAbvGr	1460 non-null	int64
52	KitchenAbvGr	1460 non-null	int64
53	KitchenQual	1460 non-null	object
54	TotRmsAbvGrd	1460 non-null	int64
55	Functional	1460 non-null	object
56	Fireplaces	1460 non-null	int64
57	FireplaceQu	770 non-null	object
58	GarageType	1379 non-null	object
59	GarageYrBlt	1379 non-null	float64
60	GarageFinish	1379 non-null	object
61	GarageCars	1460 non-null	int64
62	GarageArea	1460 non-null	int64
63	GarageQual	1379 non-null	object
64	GarageCond	1379 non-null	object
65	PavedDrive	1460 non-null	object
66	WoodDeckSF	1460 non-null	int64
67	OpenPorchSF	1460 non-null	int64
68	EnclosedPorch	1460 non-null	int64
69	3SsnPorch	1460 non-null	int64
70	ScreenPorch	1460 non-null	int64
71	PoolArea	1460 non-null	int64
72	PoolQC	7 non-null	object

```

73 Fence                281 non-null    object
74 MiscFeature          54 non-null    object
75 MiscVal              1460 non-null  int64
76 MoSold               1460 non-null  int64
77 YrSold               1460 non-null  int64
78 SaleType             1460 non-null  object
79 SaleCondition        1460 non-null  object
80 SalePrice            1460 non-null  int64

```

dtypes: float64(3), int64(35), object(43)

memory usage: 924.0+ KB

Descriptive statistics for quantitative columns for train

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	\
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	
std	421.610009	42.300571	24.284752	9981.264932	1.382997	
min	1.000000	20.000000	21.000000	1300.000000	1.000000	
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	

	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	\
count	1460.000000	1460.000000	1460.000000	1452.000000	1460.000000	
mean	5.575342	1971.267808	1984.865753	103.685262	443.639726	
std	1.112799	30.202904	20.645407	181.066207	456.098091	
min	1.000000	1872.000000	1950.000000	0.000000	0.000000	
25%	5.000000	1954.000000	1967.000000	0.000000	0.000000	

	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF	1stFlrSF	2ndFlrSF	\
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	
mean	46.549315	567.240411	1057.429452	1162.626712	346.992466	
std	161.319273	441.866955	438.705324	386.587738	436.528436	
min	0.000000	0.000000	0.000000	334.000000	0.000000	
25%	0.000000	223.000000	795.750000	882.000000	0.000000	

	LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath	FullBath	\
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	
mean	5.844521	1515.463699	0.425342	0.057534	1.565068	
std	48.623081	525.480383	0.518911	0.238753	0.550916	
min	0.000000	334.000000	0.000000	0.000000	0.000000	
25%	0.000000	1129.500000	0.000000	0.000000	1.000000	

	HalfBath	BedroomAbvGr	KitchenAbvGr	TotRmsAbvGrd	Fireplaces	\
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	
mean	0.382877	2.866438	1.046575	6.517808	0.613014	
std	0.502885	0.815778	0.220338	1.625393	0.644666	
min	0.000000	0.000000	0.000000	2.000000	0.000000	
25%	0.000000	2.000000	1.000000	5.000000	0.000000	

	GarageYrBlt	GarageCars	GarageArea	WoodDeckSF	OpenPorchSF	\
count	1379.000000	1460.000000	1460.000000	1460.000000	1460.000000	

mean	1978.506164	1.767123	472.980137	94.244521	46.660274
std	24.689725	0.747315	213.804841	125.338794	66.256028
min	1900.000000	0.000000	0.000000	0.000000	0.000000
25%	1961.000000	1.000000	334.500000	0.000000	0.000000

	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	MiscVal \
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000
mean	21.954110	3.409589	15.060959	2.758904	43.489041
std	61.119149	29.317331	55.757415	40.177307	496.123024
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000

	MoSold	YrSold	SalePrice
count	1460.000000	1460.000000	1460.000000
mean	6.321918	2007.815753	180921.195890
std	2.703626	1.328095	79442.502883
min	1.000000	2006.000000	34900.000000
25%	5.000000	2007.000000	129975.000000

All missing values in columns

We create a DF that has 3 columns. The first is the number of missing values, the second is the percentage of missing values, and the third is the data types.

missing\_values - Shows the number of missing values

missing\_percentage - Shows in percentage how many values are missing in a column

missing\_data - creating DF

Missing Value Columns for train:

	Total Missing for train	Percentage for train	Type for train
PoolQC	1453	99.520548	object
MiscFeature	1406	96.301370	object
Alley	1369	93.767123	object
Fence	1179	80.753425	object
MasVnrType	872	59.726027	object
FireplaceQu	690	47.260274	object
LotFrontage	259	17.739726	float64
GarageQual	81	5.547945	object
GarageFinish	81	5.547945	object
GarageType	81	5.547945	object
GarageYrBlt	81	5.547945	float64
GarageCond	81	5.547945	object
BsmtFinType2	38	2.602740	object
BsmtExposure	38	2.602740	object
BsmtCond	37	2.534247	object
BsmtQual	37	2.534247	object
BsmtFinType1	37	2.534247	object
MasVnrArea	8	0.547945	float64
Electrical	1	0.068493	object

Descriptive statistics for quantitative columns for test

	Id	MSSubClass	LotFrontage	LotArea	OverallQual \
count	1459.000000	1459.000000	1232.000000	1459.000000	1459.000000

mean	2190.000000	57.378341	68.580357	9819.161069	6.078821
std	421.321334	42.746880	22.376841	4955.517327	1.436812
min	1461.000000	20.000000	21.000000	1470.000000	1.000000
25%	1825.500000	20.000000	58.000000	7391.000000	5.000000
	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1 \
count	1459.000000	1459.000000	1459.000000	1444.000000	1458.000000
mean	5.553804	1971.357779	1983.662783	100.709141	439.203704
std	1.113740	30.390071	21.130467	177.625900	455.268042
min	1.000000	1879.000000	1950.000000	0.000000	0.000000
25%	5.000000	1953.000000	1963.000000	0.000000	0.000000
	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF	1stFlrSF	2ndFlrSF \
count	1458.000000	1458.000000	1458.000000	1459.000000	1459.000000
mean	52.619342	554.294925	1046.117970	1156.534613	325.967786
std	176.753926	437.260486	442.898624	398.165820	420.610226
min	0.000000	0.000000	0.000000	407.000000	0.000000
25%	0.000000	219.250000	784.000000	873.500000	0.000000
	LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath	FullBath \
count	1459.000000	1459.000000	1457.000000	1457.000000	1459.000000
mean	3.543523	1486.045922	0.434454	0.065202	1.570939
std	44.043251	485.566099	0.530648	0.252468	0.555190
min	0.000000	407.000000	0.000000	0.000000	0.000000
25%	0.000000	1117.500000	0.000000	0.000000	1.000000
	HalfBath	BedroomAbvGr	KitchenAbvGr	TotRmsAbvGrd	Fireplaces \
count	1459.000000	1459.000000	1459.000000	1459.000000	1459.000000
mean	0.377656	2.854010	1.042495	6.385195	0.58122
std	0.503017	0.829788	0.208472	1.508895	0.64742
min	0.000000	0.000000	0.000000	3.000000	0.00000
25%	0.000000	2.000000	1.000000	5.000000	0.00000
	GarageYrBlt	GarageCars	GarageArea	WoodDeckSF	OpenPorchSF \
count	1381.000000	1458.000000	1458.000000	1459.000000	1459.000000
mean	1977.721217	1.766118	472.768861	93.174777	48.313914
std	26.431175	0.775945	217.048611	127.744882	68.883364
min	1895.000000	0.000000	0.000000	0.000000	0.000000
25%	1959.000000	1.000000	318.000000	0.000000	0.000000
	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	MiscVal \
count	1459.000000	1459.000000	1459.000000	1459.000000	1459.000000
mean	24.243317	1.794380	17.064428	1.744345	58.167923
std	67.227765	20.207842	56.609763	30.491646	630.806978
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
	MoSold	YrSold			

```

count  1459.000000  1459.000000
mean    6.104181  2007.769705
std     2.722432   1.301740
min     1.000000  2006.000000
25%     4.000000  2007.000000

```

Missing Value Columns for test:

	Total Missing for test	Percentage for test	Type for test
PoolQC	1456	99.794380	object
MiscFeature	1408	96.504455	object
Alley	1352	92.666210	object
Fence	1169	80.123372	object
MasVnrType	894	61.274846	object
FireplaceQu	730	50.034270	object
LotFrontage	227	15.558602	float64
GarageYrBlt	78	5.346127	float64
GarageCond	78	5.346127	object
GarageFinish	78	5.346127	object
GarageQual	78	5.346127	object
GarageType	76	5.209047	object
BsmtCond	45	3.084304	object
BsmtQual	44	3.015764	object
BsmtExposure	44	3.015764	object
BsmtFinType1	42	2.878684	object
BsmtFinType2	42	2.878684	object
MasVnrArea	15	1.028101	float64
MSZoning	4	0.274160	object
BsmtHalfBath	2	0.137080	float64
Utilities	2	0.137080	object
Functional	2	0.137080	object
BsmtFullBath	2	0.137080	float64
BsmtFinSF1	1	0.068540	float64
Exterior1st	1	0.068540	object
TotalBsmtSF	1	0.068540	float64
BsmtUnfSF	1	0.068540	float64
BsmtFinSF2	1	0.068540	float64
SaleType	1	0.068540	object
KitchenQual	1	0.068540	object
GarageCars	1	0.068540	float64
GarageArea	1	0.068540	float64
Exterior2nd	1	0.068540	object

Replacing empty values with mode and mean values for train

Saving statistics from train

Replacing empty values with mode and mean values for test

For test we use the mean value or mode, where the gaps are  $10\% < x < 25\%$

Using interpolation to replace empty values where gaps are  $10\% < x < 25\%$

Removing empty values that have a gap value greater than 45% for train

Removing empty values that have a skip value greater than 45% for test

Let's see if there are any duplicate lines for train

0

Let's see if there are any duplicate lines for test

0

Statistics for SalePrice

count 1460.000000

mean 180921.195890

std 79442.502883

min 34900.000000

25% 129975.000000

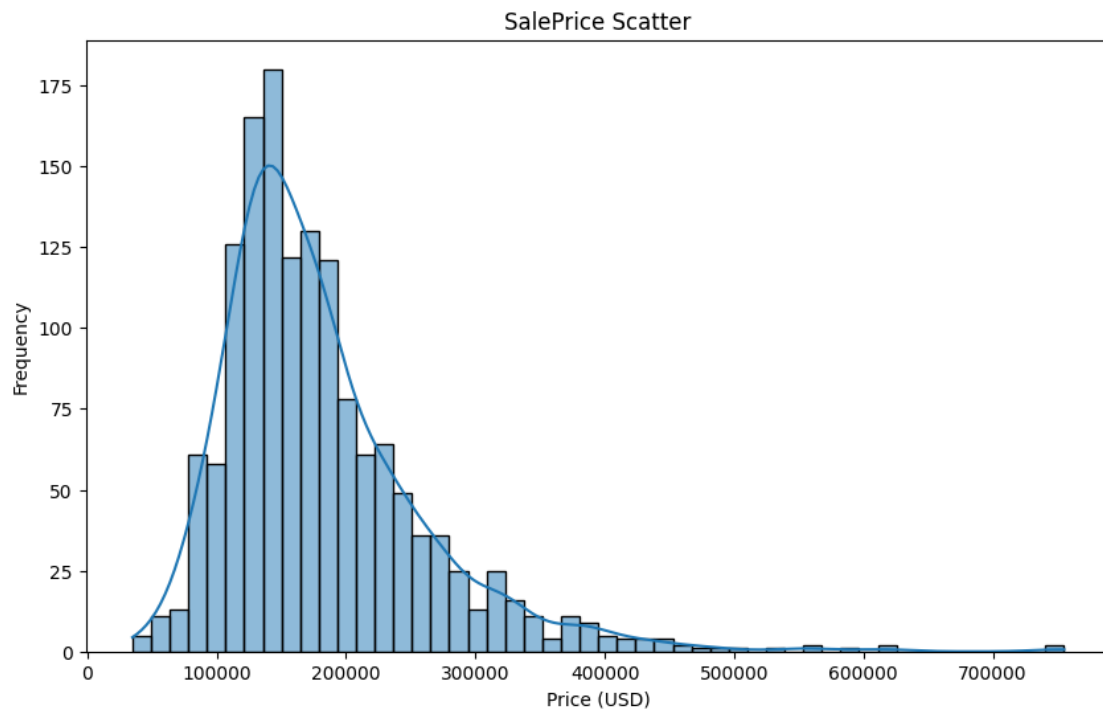
50% 163000.000000

75% 214000.000000

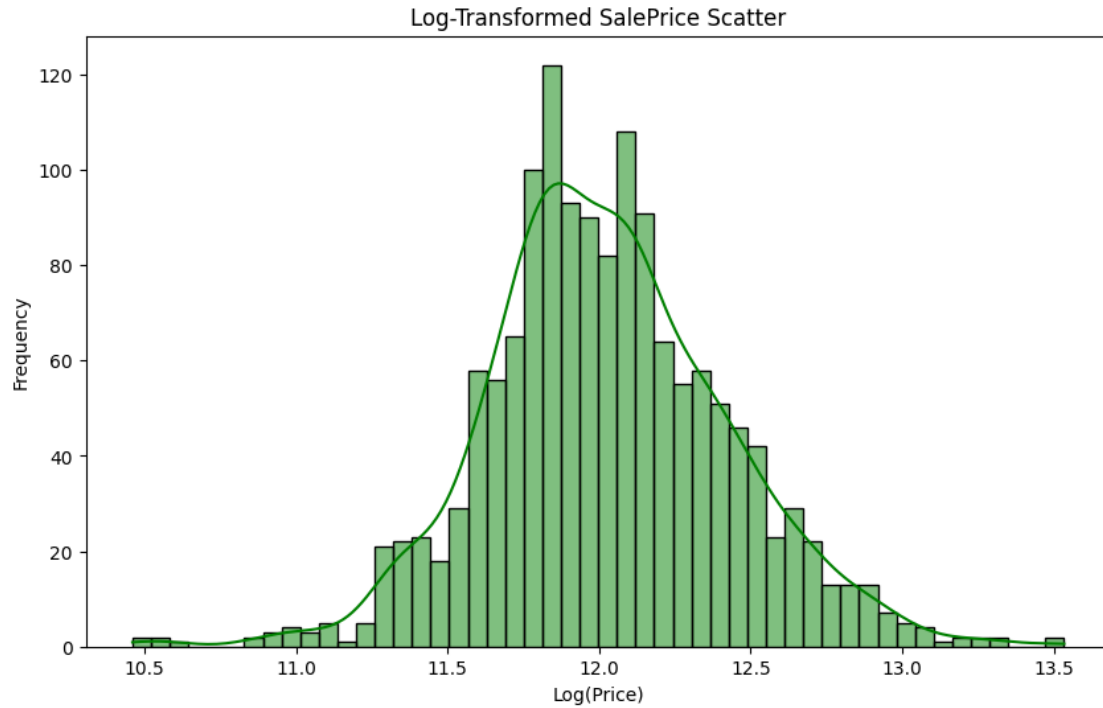
max 755000.000000

Name: SalePrice, dtype: float64

Graph without normalization



Normalization. Logarithmization.

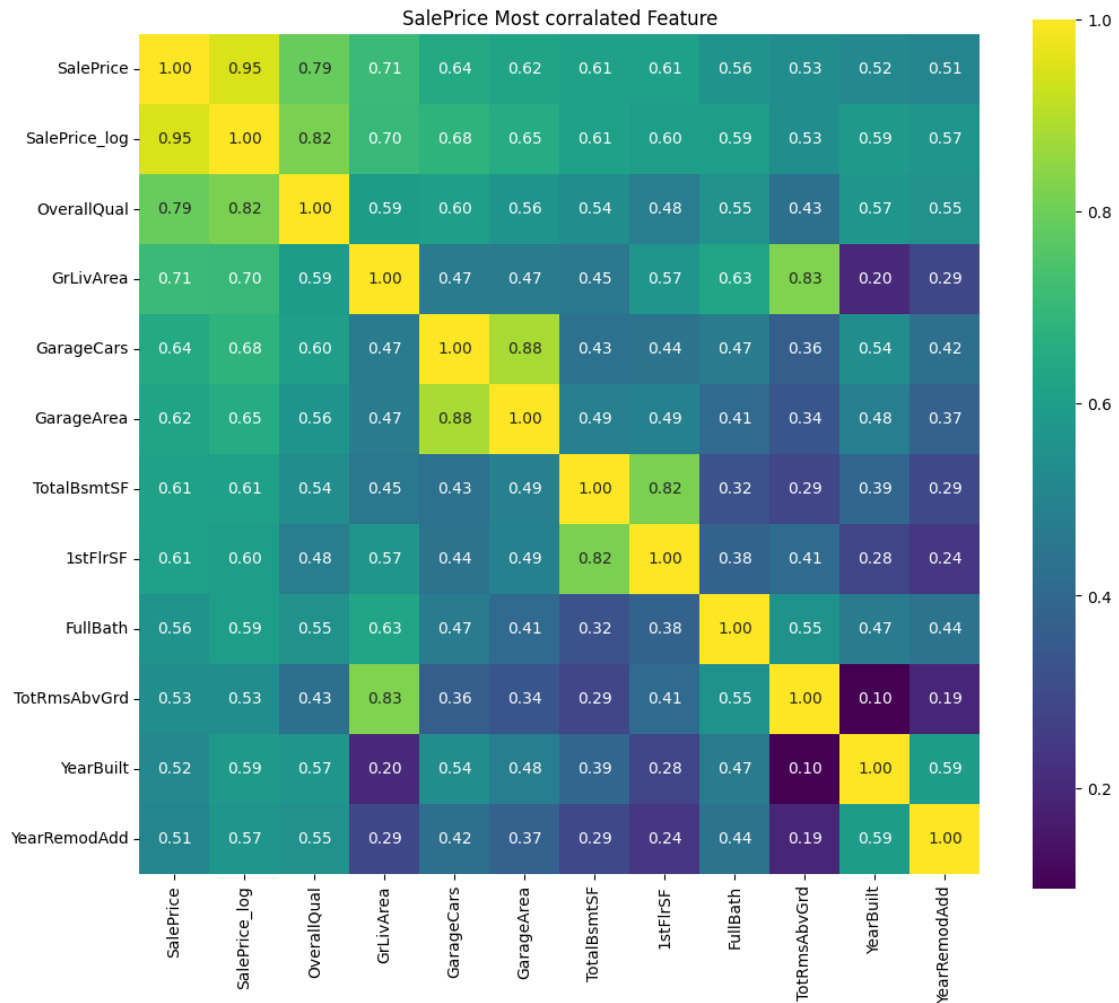


After normalization, it can be seen that the data is centered, which will improve the performance of the model.

Creating a heat map to find correlation

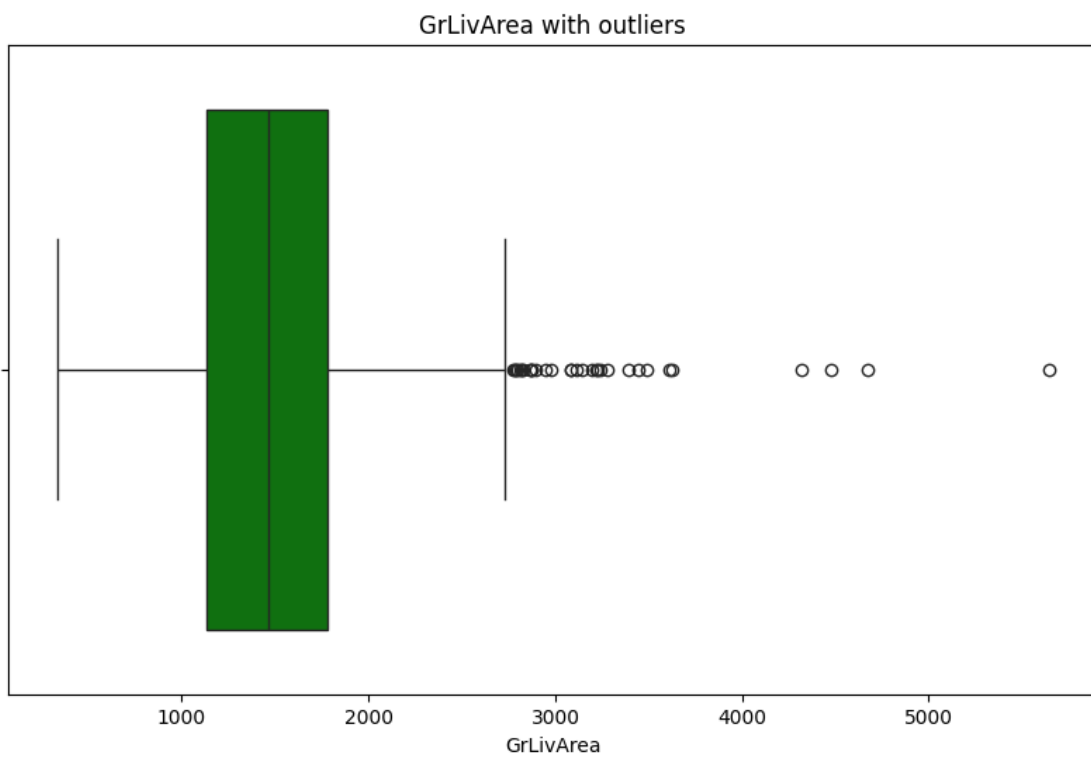
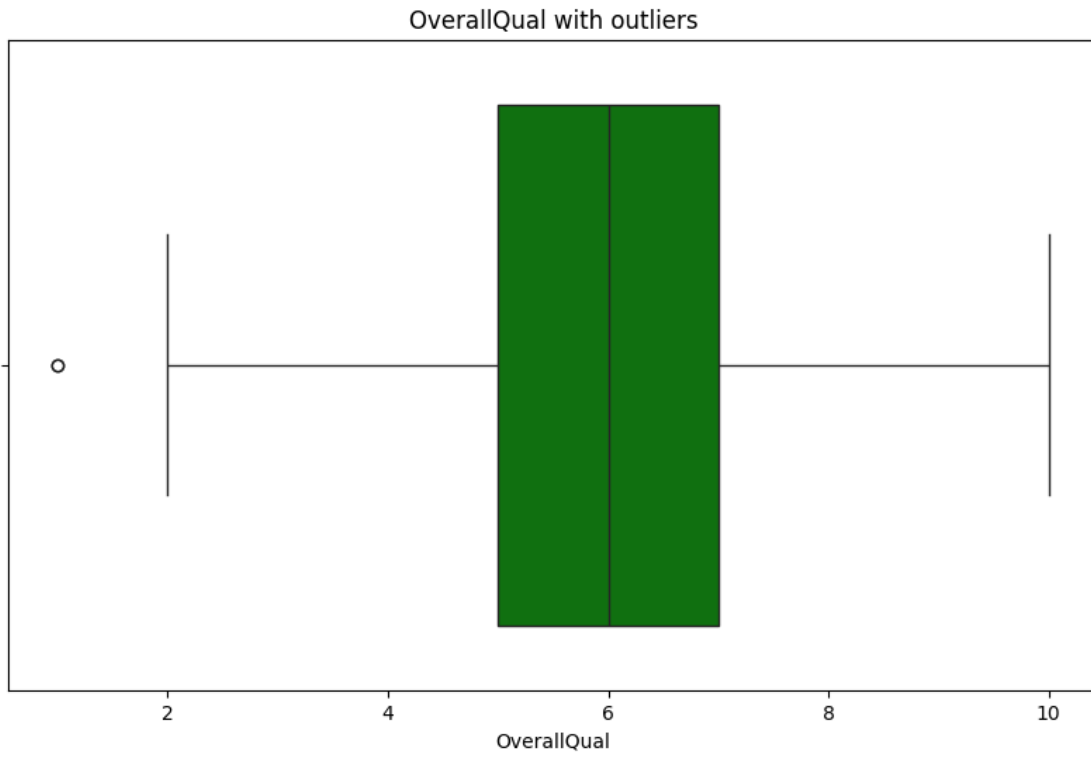
```
['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond',
'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2',
'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea',
'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr',
'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageYrBlt', 'GarageCars',
'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SalePrice',
'SalePrice_log']
```

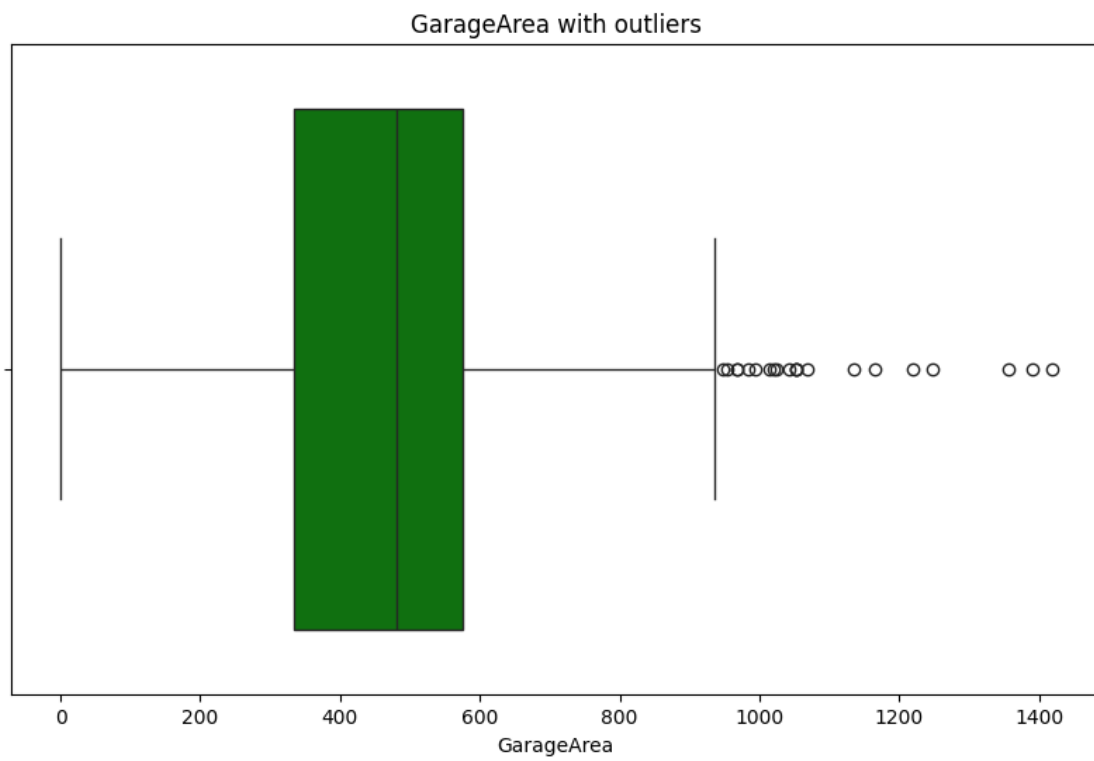




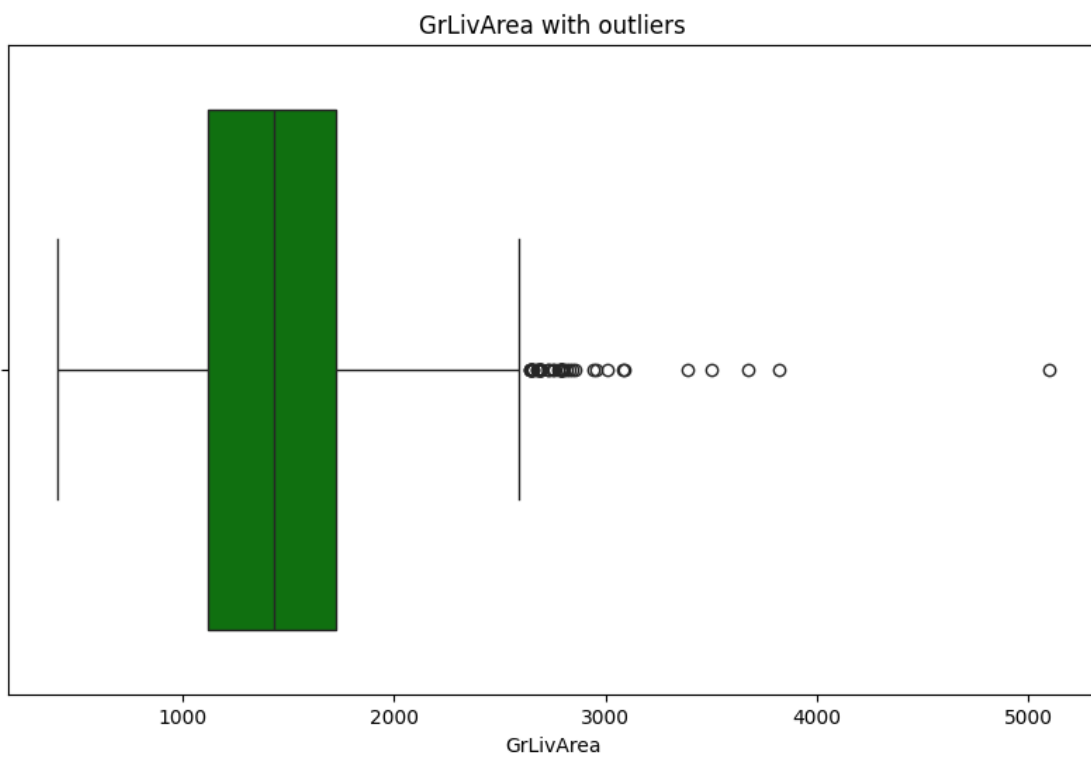
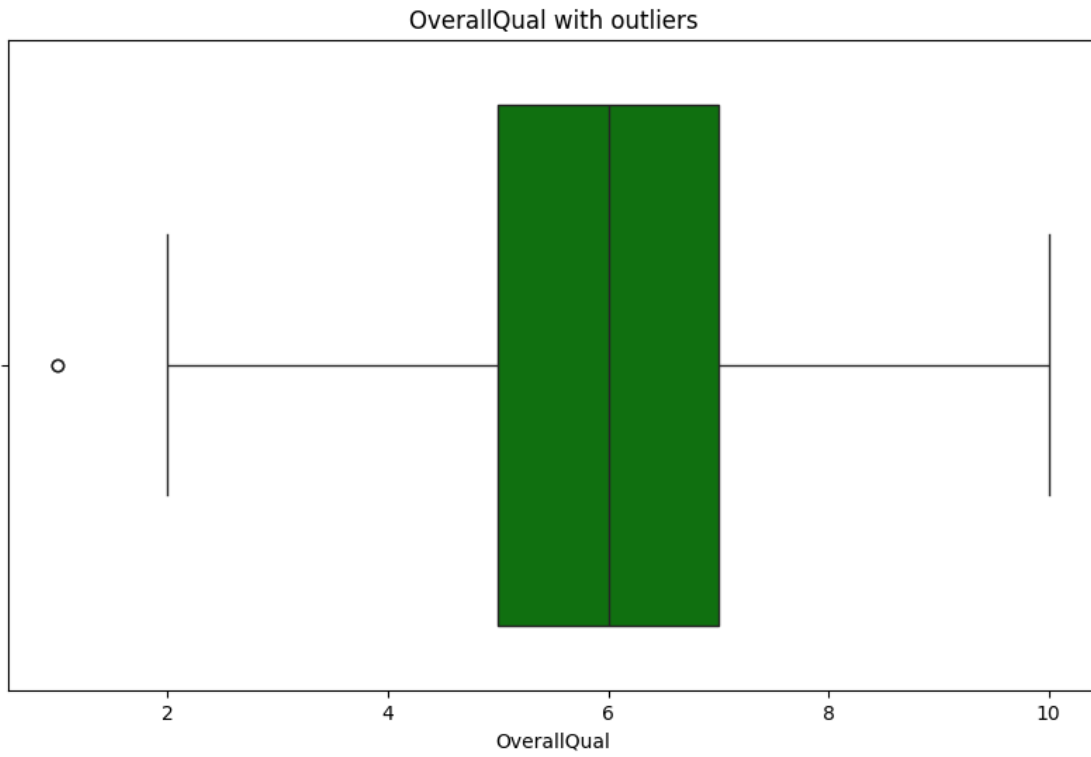
Identifying outliers for train

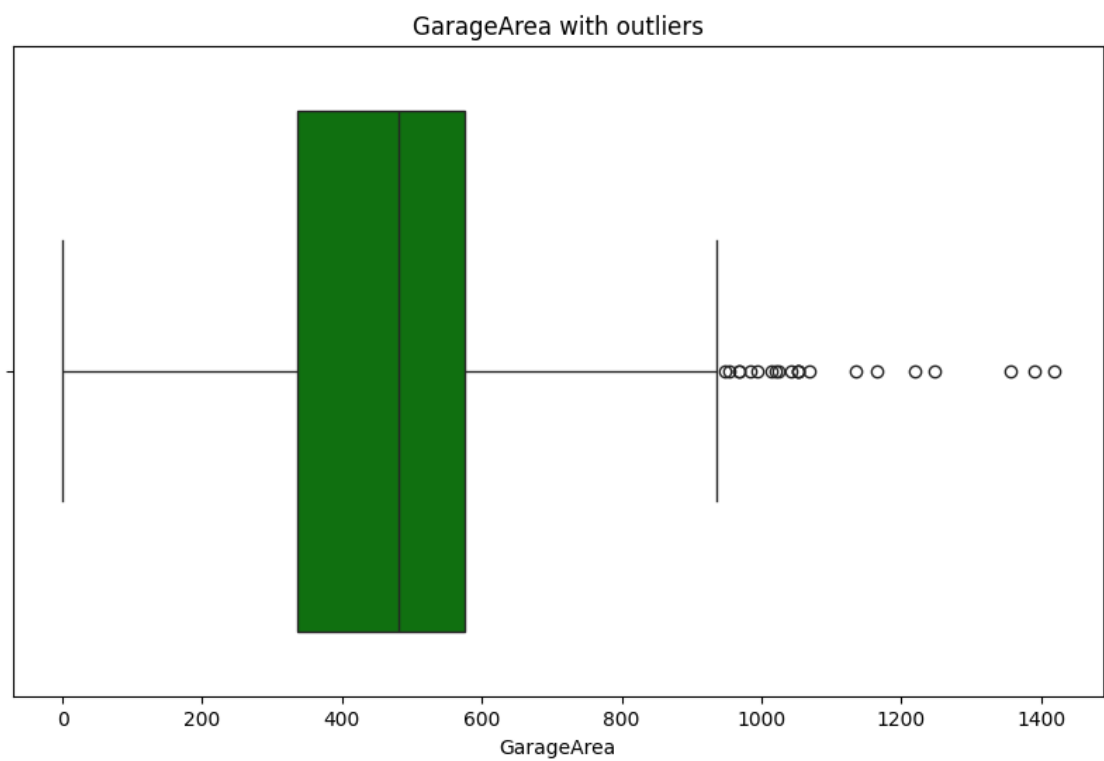
First, you need to plot a graph and make sure that there are a lot of emissions and they are not necessary. For example, a house can really be expensive.



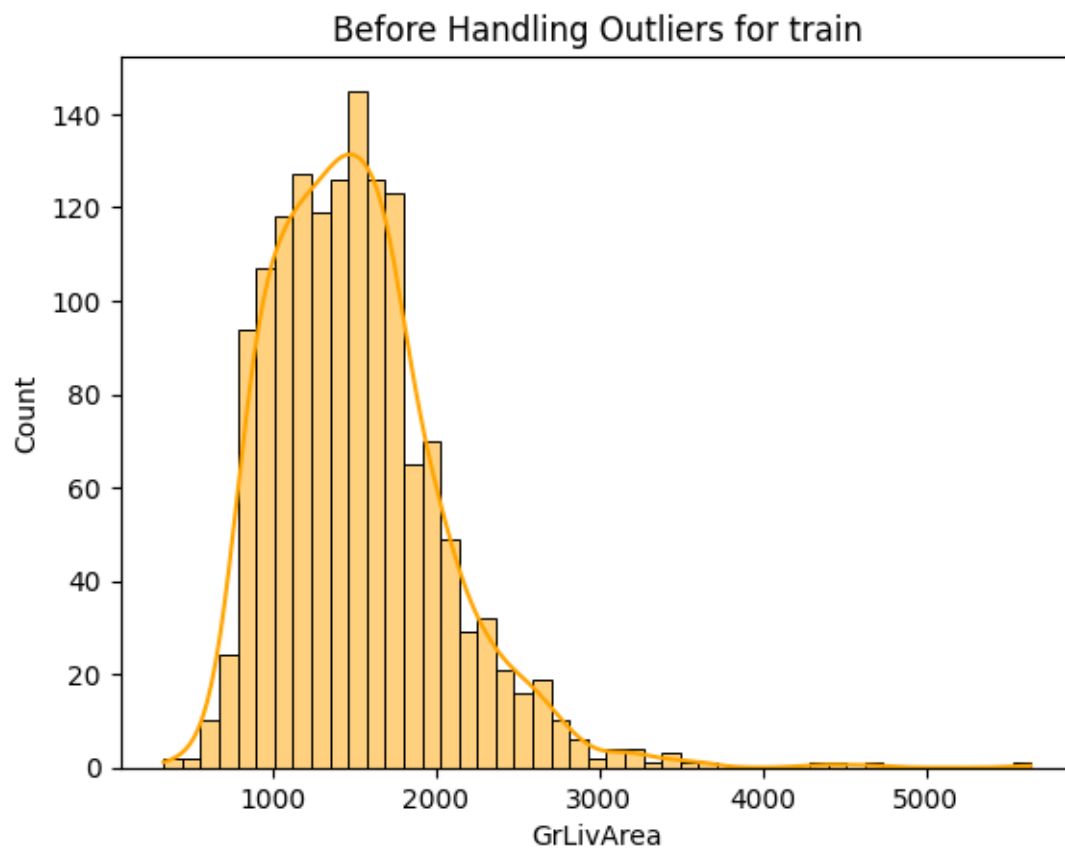


Identifying outliers for test

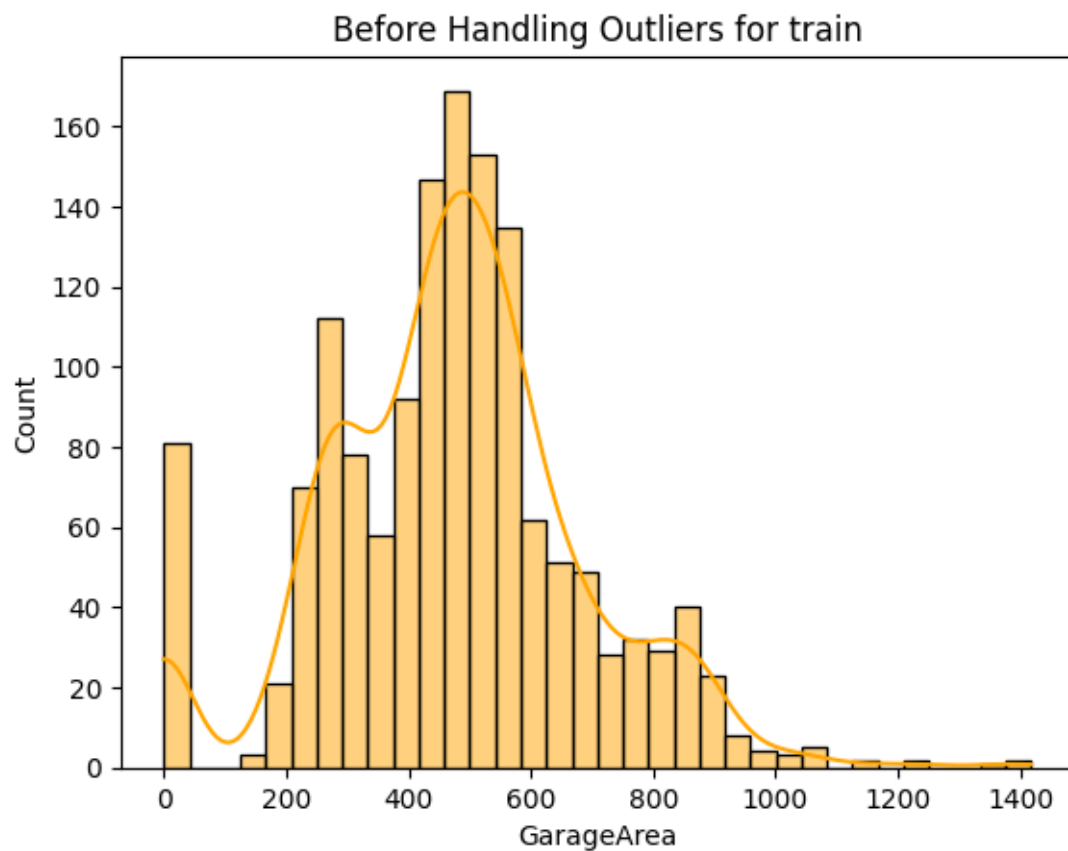




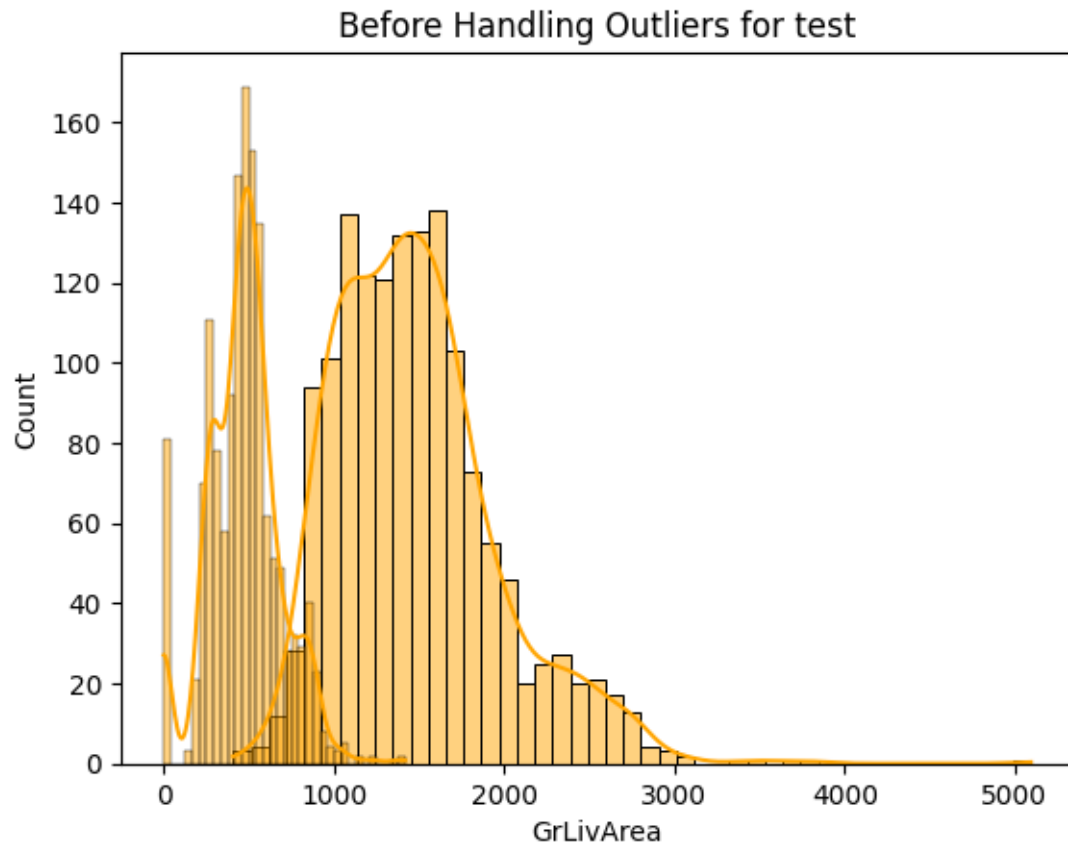
Histogram before outliers for GrLivArea for train



Histogram before outliers for GarageArea for train



Histogram before outliers for GrLivArea for test  
Histogram before outliers for GarageArea for test



IQR method for data that has many outliers for train

If the values go beyond  $\text{lower\_bound} = Q1 - 1.5 * \text{IQR}$  and  $\text{upper\_bound} = Q3 + 1.5 * \text{IQR}$  - these are outliers

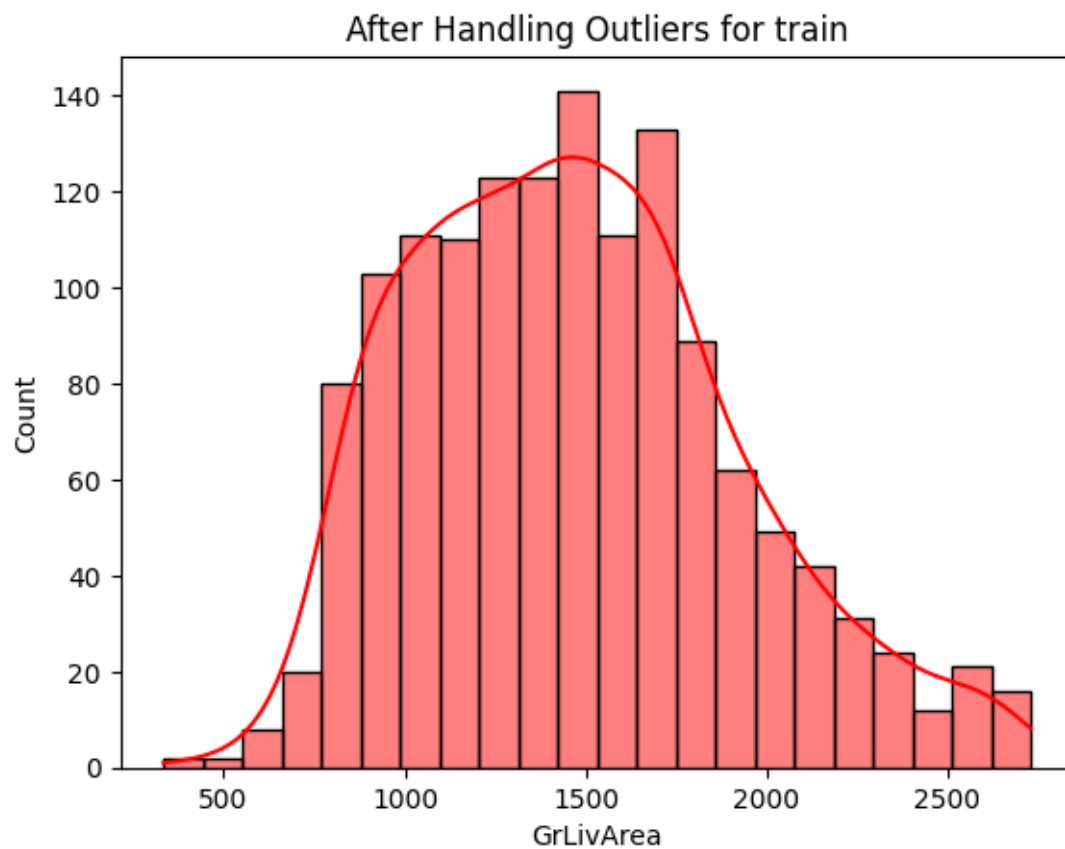
IQR method for data that have many outliers for test

Resetting indices for train

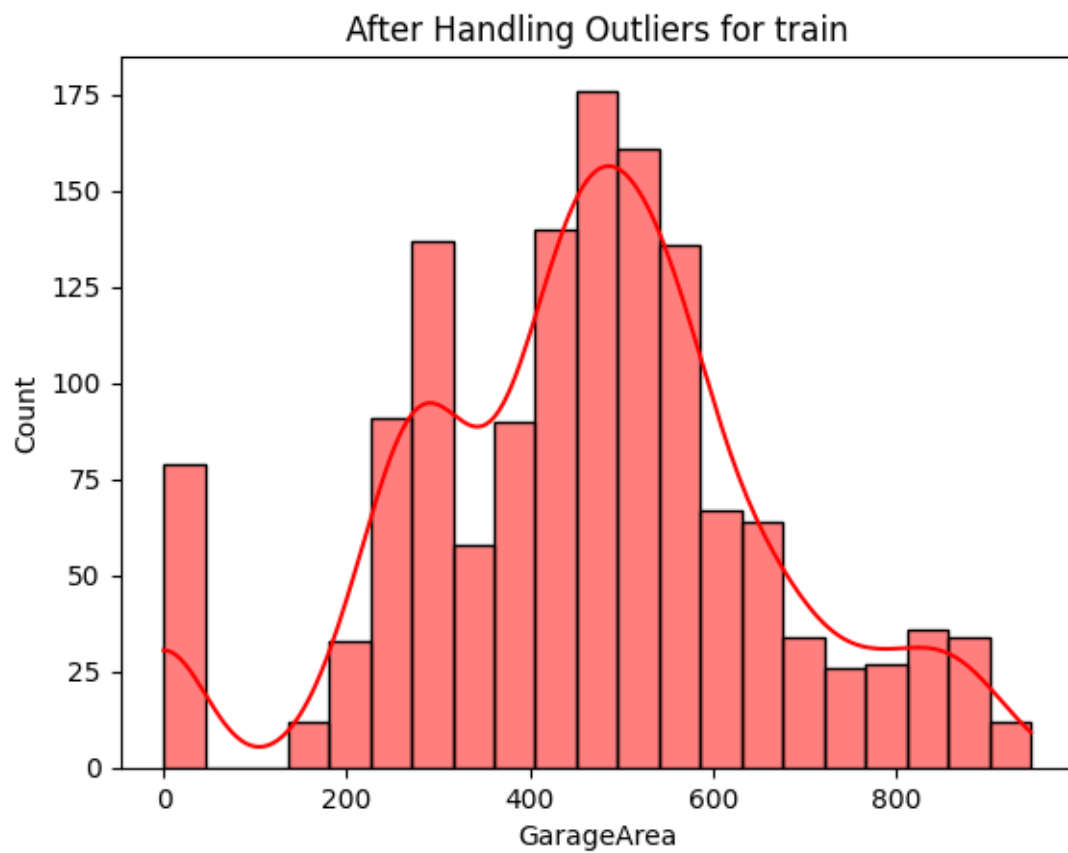
Resetting indices for test

Histogram without outliers for GrLivArea for train

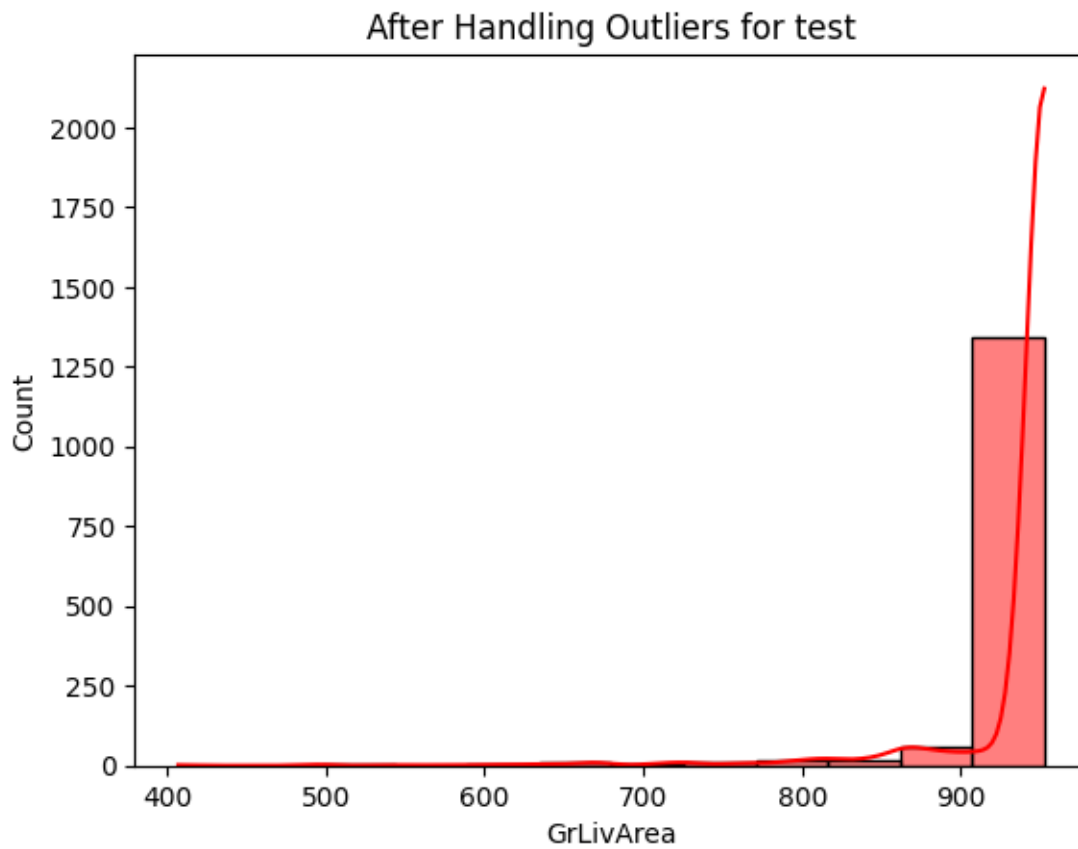




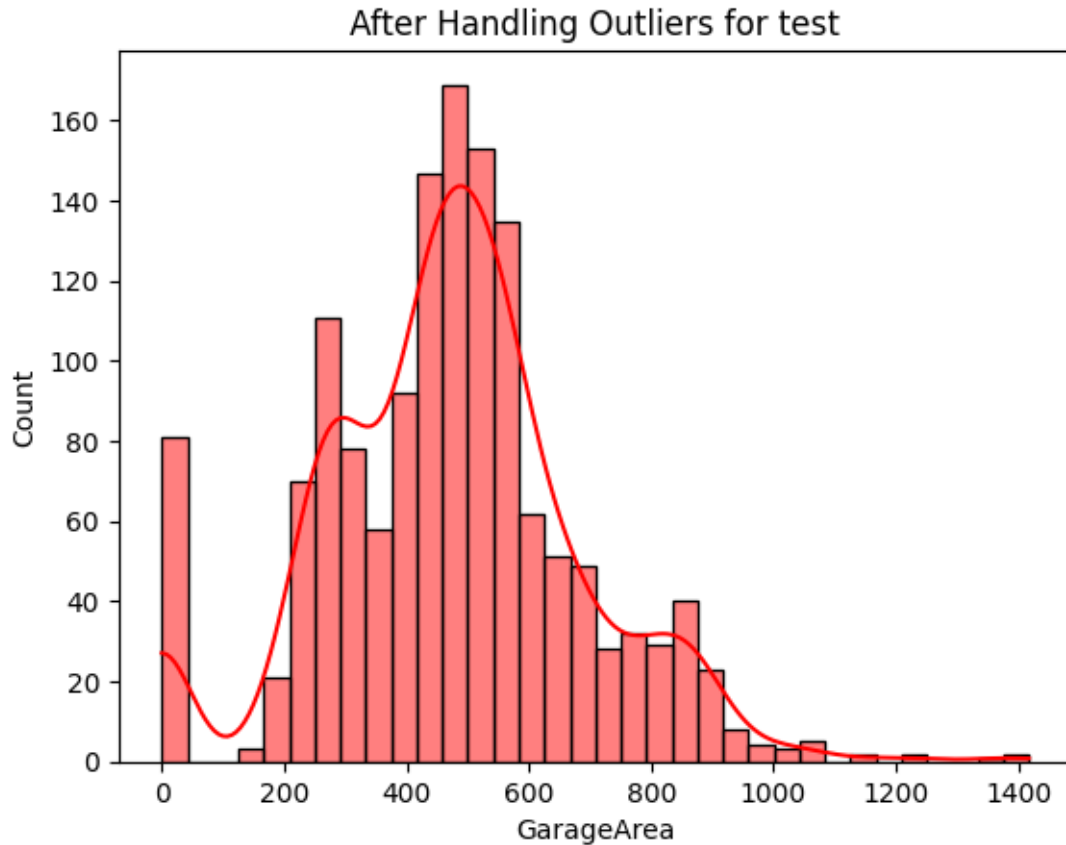
Outlier-free histogram for GarageArea for train



Histogram without outliers for GrLivArea for test



Histogram without outliers for GarageArea for test



Using Categorical Data for Analysis. Neighborhood for train

Data processing

One-hot encoding

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	\
0	1	60	RL	65.0	8450	Pave	Reg	Lvl	
1	2	20	RL	80.0	9600	Pave	Reg	Lvl	
2	3	60	RL	68.0	11250	Pave	IR1	Lvl	
3	4	70	RL	60.0	9550	Pave	IR1	Lvl	
4	5	60	RL	84.0	14260	Pave	IR1	Lvl	

	Utilities	LotConfig	LandSlope	Condition1	Condition2	BldgType	HouseStyle	\
0	AllPub	Inside	Gtl	Norm	Norm	1Fam	2Story	
1	AllPub	FR2	Gtl	Feedr	Norm	1Fam	1Story	
2	AllPub	Inside	Gtl	Norm	Norm	1Fam	2Story	
3	AllPub	Corner	Gtl	Norm	Norm	1Fam	2Story	
4	AllPub	FR2	Gtl	Norm	Norm	1Fam	2Story	

	OverallQual	OverallCond	YearBuilt	YearRemodAdd	RoofStyle	RoofMatl	\
0	7	5	2003	2003	Gable	CompShg	
1	6	8	1976	1976	Gable	CompShg	

2	7	5	2001	2002	Gable	CompShg
3	7	5	1915	1970	Gable	CompShg
4	8	5	2000	2000	Gable	CompShg

	Exterior1st	Exterior2nd	MasVnrArea	ExterQual	ExterCond	Foundation	BsmtQual	\
0	VinylSd	VinylSd	196.0	Gd	TA	PConc	Gd	
1	MetalSd	MetalSd	0.0	TA	TA	CBlock	Gd	
2	VinylSd	VinylSd	162.0	Gd	TA	PConc	Gd	
3	Wd Sdng	Wd Shng	0.0	TA	TA	BrkTil	TA	
4	VinylSd	VinylSd	350.0	Gd	TA	PConc	Gd	

	BsmtCond	BsmtExposure	BsmtFinType1	BsmtFinSF1	BsmtFinType2	BsmtFinSF2	\
0	TA	No	GLQ	706	Unf	0	
1	TA	Gd	ALQ	978	Unf	0	
2	TA	Mn	GLQ	486	Unf	0	
3	Gd	No	ALQ	216	Unf	0	
4	TA	Av	GLQ	655	Unf	0	

	BsmtUnfSF	TotalBsmtSF	Heating	HeatingQC	CentralAir	Electrical	1stFlrSF	\
0	150	856	GasA	Ex	Y	SBrkr	856	
1	284	1262	GasA	Ex	Y	SBrkr	1262	
2	434	920	GasA	Ex	Y	SBrkr	920	
3	540	756	GasA	Gd	Y	SBrkr	961	
4	490	1145	GasA	Ex	Y	SBrkr	1145	

	2ndFlrSF	LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath	FullBath	\
0	854	0	1710	1	0	2	
1	0	0	1262	0	1	2	
2	866	0	1786	1	0	2	
3	756	0	1717	1	0	1	
4	1053	0	2198	1	0	2	

	HalfBath	BedroomAbvGr	KitchenAbvGr	KitchenQual	TotRmsAbvGrd	Functional	\
0	1	3	1	Gd	8	Typ	
1	0	3	1	TA	6	Typ	
2	1	3	1	Gd	6	Typ	
3	0	3	1	Gd	7	Typ	
4	1	4	1	Gd	9	Typ	

	Fireplaces	GarageType	GarageYrBlt	GarageFinish	GarageCars	GarageArea	\
0	0	Attchd	2003.0	RFn	2	548	
1	1	Attchd	1976.0	RFn	2	460	
2	1	Attchd	2001.0	RFn	2	608	
3	1	Detchd	1998.0	Unf	3	642	
4	1	Attchd	2000.0	RFn	3	836	

	GarageQual	GarageCond	PavedDrive	WoodDeckSF	OpenPorchSF	EnclosedPorch	\
0	TA	TA	Y	0	61	0	

1	TA	TA	Y	298	0	0
2	TA	TA	Y	0	42	0
3	TA	TA	Y	0	35	272
4	TA	TA	Y	192	84	0

	3SsnPorch	ScreenPorch	PoolArea	MiscVal	MoSold	YrSold	SaleType	\
0	0	0	0	0	2	2008	WD	
1	0	0	0	0	5	2007	WD	
2	0	0	0	0	9	2008	WD	
3	0	0	0	0	2	2006	WD	
4	0	0	0	0	12	2008	WD	

	SaleCondition	SalePrice	SalePrice_log	Neighborhood_blueste	\
0	Normal	208500	12.247699	False	
1	Normal	181500	12.109016	False	
2	Normal	223500	12.317171	False	
3	Abnorml	140000	11.849405	False	
4	Normal	250000	12.429220	False	

	Neighborhood_brdale	Neighborhood_brkside	Neighborhood_clearcr	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	False	False	

	Neighborhood_collgcr	Neighborhood_crawfor	Neighborhood_edwards	\
0	True	False	False	
1	False	False	False	
2	True	False	False	
3	False	True	False	
4	False	False	False	

	Neighborhood_gilbert	Neighborhood_idotrr	Neighborhood_meadowv	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	False	False	

	Neighborhood_mitchel	Neighborhood_names	Neighborhood_noridge	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	False	True	

	Neighborhood_npkvill	Neighborhood_nridght	Neighborhood_nwames	\
--	----------------------	----------------------	---------------------	---

0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False

	Neighborhood_oldtown	Neighborhood_sawyer	Neighborhood_sawyerw \
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False

	Neighborhood_somerst	Neighborhood_stonebr	Neighborhood_swisu \
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False

	Neighborhood_timber	Neighborhood_veenker
0	False	False
1	False	True
2	False	False
3	False	False
4	False	False

Using Categorical Data for Analysis. Neighborhood for test

One-hot encoding

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape \
0	1461	20	RH	65.0	11622	Pave	Reg
1	1462	20	RL	80.0	14267	Pave	IR1
2	1463	60	RL	68.0	13830	Pave	IR1
3	1464	60	RL	60.0	9978	Pave	IR1
4	1465	120	RL	84.0	5005	Pave	IR1

	LandContour	Utilities	LotConfig	LandSlope	Condition1	Condition2	BldgType \
0	Lvl	AllPub	Inside	Gtl	Feedr	Norm	1Fam
1	Lvl	AllPub	Corner	Gtl	Norm	Norm	1Fam
2	Lvl	AllPub	Inside	Gtl	Norm	Norm	1Fam
3	Lvl	AllPub	Inside	Gtl	Norm	Norm	1Fam
4	HLS	AllPub	Inside	Gtl	Norm	Norm	TwnhsE

	HouseStyle	OverallQual	OverallCond	YearBuilt	YearRemodAdd	RoofStyle \
0	1Story	5	6	1961	1961	Gable
1	1Story	6	6	1958	1958	Hip
2	2Story	5	5	1997	1998	Gable
3	2Story	6	6	1998	1998	Gable
4	1Story	8	5	1992	1992	Gable

	RoofMatl	Exterior1st	Exterior2nd	MasVnrArea	ExterQual	ExterCond	Foundation	\
0	CompShg	VinylSd	VinylSd	196.0	TA	TA	CBlock	
1	CompShg	Wd Sdng	Wd Sdng	0.0	TA	TA	CBlock	
2	CompShg	VinylSd	VinylSd	162.0	TA	TA	PConc	
3	CompShg	VinylSd	VinylSd	0.0	TA	TA	PConc	
4	CompShg	HdBoard	HdBoard	350.0	Gd	TA	PConc	

	BsmtQual	BsmtCond	BsmtExposure	BsmtFinType1	BsmtFinSF1	BsmtFinType2	\
0	TA	TA	No	Rec	706	LwQ	
1	TA	TA	No	ALQ	978	Unf	
2	Gd	TA	No	GLQ	486	Unf	
3	TA	TA	No	GLQ	216	Unf	
4	Gd	TA	No	ALQ	655	Unf	

	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF	Heating	HeatingQC	CentralAir	Electrical	\
0	0	150	856	GasA	TA	Y	SBrkr	
1	0	284	1262	GasA	TA	Y	SBrkr	
2	0	434	920	GasA	Gd	Y	SBrkr	
3	0	540	756	GasA	Ex	Y	SBrkr	
4	0	490	1145	GasA	Ex	Y	SBrkr	

	1stFlrSF	2ndFlrSF	LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath	\
0	896	0	0	896.0	1	0	
1	1329	0	0	952.5	0	1	
2	928	701	0	952.5	1	0	
3	926	678	0	952.5	1	0	
4	1280	0	0	952.5	1	0	

	FullBath	HalfBath	BedroomAbvGr	KitchenAbvGr	KitchenQual	TotRmsAbvGrd	\
0	1	0	2	1	TA	5	
1	1	1	3	1	Gd	6	
2	2	1	3	1	TA	6	
3	2	1	3	1	Gd	7	
4	2	0	2	1	Gd	5	

	Functional	Fireplaces	GarageType	GarageYrBlt	GarageFinish	GarageCars	\
0	Typ	0	Attchd	2003.0	Unf	2	
1	Typ	0	Attchd	1976.0	Unf	2	
2	Typ	1	Attchd	2001.0	Fin	2	
3	Typ	1	Attchd	1998.0	Fin	3	
4	Typ	0	Attchd	2000.0	RFn	3	

	GarageArea	GarageQual	GarageCond	PavedDrive	WoodDeckSF	OpenPorchSF	\
0	548	TA	TA	Y	140	0	
1	460	TA	TA	Y	393	36	
2	608	TA	TA	Y	212	34	
3	642	TA	TA	Y	360	36	



4	836	TA	TA	Y	0	82	
	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	MiscVal	MoSold	YrSold \
0	0	0	120	0	0	6	2010
1	0	0	0	0	12500	6	2010
2	0	0	0	0	0	3	2010
3	0	0	0	0	0	6	2010
4	0	0	144	0	0	1	2010

	SaleType	SaleCondition	Neighborhood_blueste	Neighborhood_brdale \
0	WD	Normal	False	False
1	WD	Normal	False	False
2	WD	Normal	False	False
3	WD	Normal	False	False
4	WD	Normal	False	False

	Neighborhood_brkside	Neighborhood_clearcr	Neighborhood_collgcr \
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False

	Neighborhood_crawfor	Neighborhood_edwards	Neighborhood_gilbert \
0	False	False	False
1	False	False	False
2	False	False	True
3	False	False	True
4	False	False	False

	Neighborhood_idotrr	Neighborhood_meadowv	Neighborhood_mitchel \
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False

	Neighborhood_names	Neighborhood_noridge	Neighborhood_npkvill \
0	True	False	False
1	True	False	False
2	False	False	False
3	False	False	False
4	False	False	False

	Neighborhood_nridght	Neighborhood_nwames	Neighborhood_oldtown \
0	False	False	False
1	False	False	False
2	False	False	False

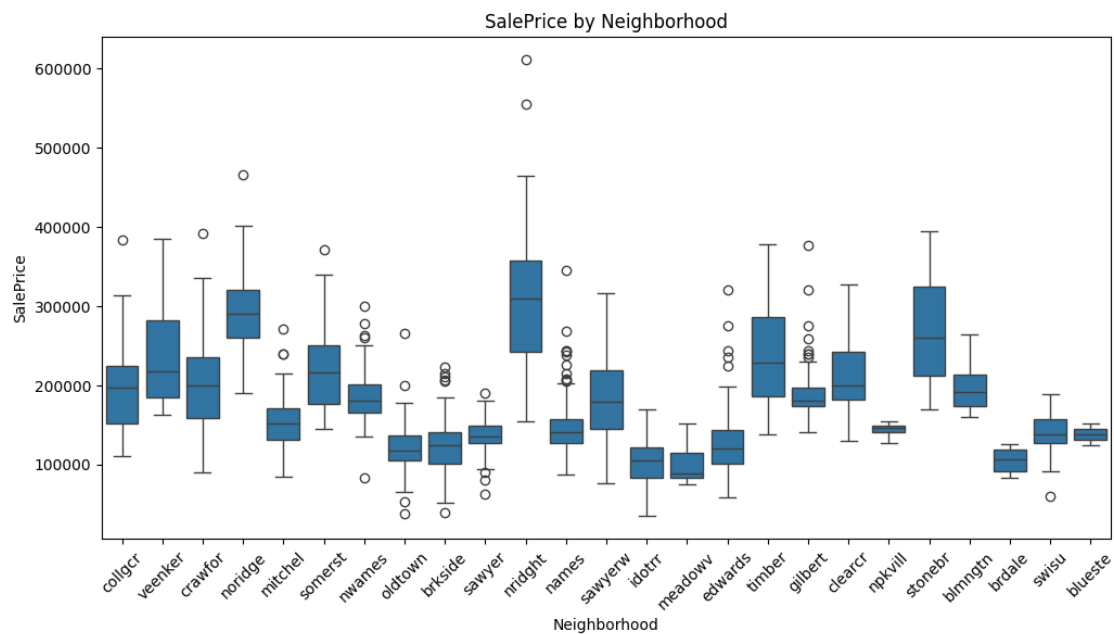
3	False	False	False
4	False	False	False

	Neighborhood_sawyer	Neighborhood_sawyerw	Neighborhood_somerst \
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False

	Neighborhood_stonebr	Neighborhood_swisu	Neighborhood_timber \
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	True	False	False

	Neighborhood_veenker
0	False
1	False
2	False
3	False
4	False

In which area are houses more expensive?



Creating a new df for the model for train

OverallQual	GrLivArea	GarageArea	Neighborhood_blueste \
-------------	-----------	------------	------------------------

0	7	1710	548	False
1	6	1262	460	False
2	7	1786	608	False
3	7	1717	642	False
4	8	2198	836	False

	Neighborhood_brdale	Neighborhood_brkside	Neighborhood_clearcr	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	False	False	

	Neighborhood_collgcr	Neighborhood_crawfor	Neighborhood_edwards	\
0	True	False	False	
1	False	False	False	
2	True	False	False	
3	False	True	False	
4	False	False	False	

	Neighborhood_gilbert	Neighborhood_idotrr	Neighborhood_meadowv	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	False	False	

	Neighborhood_mitchel	Neighborhood_names	Neighborhood_noridge	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	False	True	

	Neighborhood_npkvill	Neighborhood_nridght	Neighborhood_nwames	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	False	False	

	Neighborhood_oldtown	Neighborhood_sawyer	Neighborhood_sawyerw	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	False	False	

	Neighborhood_somerst	Neighborhood_stonebr	Neighborhood_swisu \
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False

	Neighborhood_timber	Neighborhood_veenker
0	False	False
1	False	True
2	False	False
3	False	False
4	False	False

Creating a new df for the model for test

	OverallQual	GrLivArea	GarageArea	Neighborhood_blueste \
0	5	896.0	548	False
1	6	952.5	460	False
2	5	952.5	608	False
3	6	952.5	642	False
4	8	952.5	836	False

	Neighborhood_brdale	Neighborhood_brkside	Neighborhood_clearcr \
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False

	Neighborhood_collgcr	Neighborhood_crawfor	Neighborhood_edwards \
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False

	Neighborhood_gilbert	Neighborhood_idotrr	Neighborhood_meadowv \
0	False	False	False
1	False	False	False
2	True	False	False
3	True	False	False
4	False	False	False

	Neighborhood_mitchel	Neighborhood_names	Neighborhood_noridge \
0	False	True	False
1	False	True	False
2	False	False	False
3	False	False	False
4	False	False	False

	Neighborhood_npkvill	Neighborhood_nridght	Neighborhood_nwames	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	False	False	

	Neighborhood_oldtown	Neighborhood_sawyer	Neighborhood_sawyerw	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	False	False	

	Neighborhood_somerst	Neighborhood_stonebr	Neighborhood_swisu	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	True	False	

	Neighborhood_timber	Neighborhood_veenker
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False

Column alignment

	OverallQual	GrLivArea	GarageArea	Neighborhood_blueste	\
0	5	896.0	548	False	
1	6	952.5	460	False	
2	5	952.5	608	False	
3	6	952.5	642	False	
4	8	952.5	836	False	

	Neighborhood_brdale	Neighborhood_brkside	Neighborhood_clearcr	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	False	False	

	Neighborhood_collgcr	Neighborhood_crawfor	Neighborhood_edwards	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	

4	False	False	False
	Neighborhood_gilbert	Neighborhood_idotrr	Neighborhood_meadowv \
0	False	False	False
1	False	False	False
2	True	False	False
3	True	False	False
4	False	False	False
	Neighborhood_mitchel	Neighborhood_names	Neighborhood_noridge \
0	False	True	False
1	False	True	False
2	False	False	False
3	False	False	False
4	False	False	False
	Neighborhood_npkvill	Neighborhood_nridght	Neighborhood_nwames \
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False
	Neighborhood_oldtown	Neighborhood_sawyer	Neighborhood_sawyerw \
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False
	Neighborhood_somerst	Neighborhood_stonebr	Neighborhood_swisu \
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	True	False
	Neighborhood_timber	Neighborhood_veenker	
0	False	False	
1	False	False	
2	False	False	
3	False	False	
4	False	False	

Model training

Linear

RandomForestRegressor

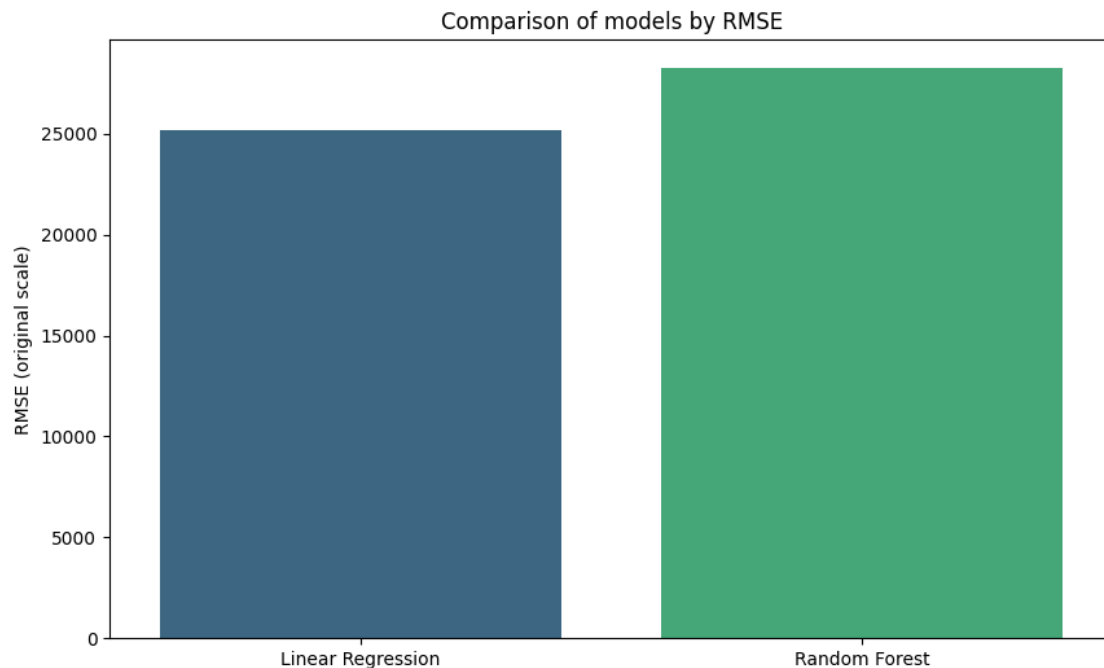
Metrics for linear regression. Log

RMSE 0.1463222517662175

```

R2 0.8545311112781473
Metrics for RandomForestRegressor. Log
RMSE 0.16610166259201772
R2 0.8125448707213268
Metrics for linear regression.
RMSE 25192.78468715876
R2 0.8642775976515636
Metrics for RandomForestRegressor.
RMSE 28244.19693371522
R2 0.8294083969402922
Cross-validation. Mean. Standard deviation. Linear model
Average RMSE 0.8232581658688632 Standard Deviation 0.009078693545882305
Cross-validation. Average. Standard Deviation. Random Forest
Average RMSE: -0.7900592851372817 Standard Deviation 0.0051711900287019375
Comparison of models

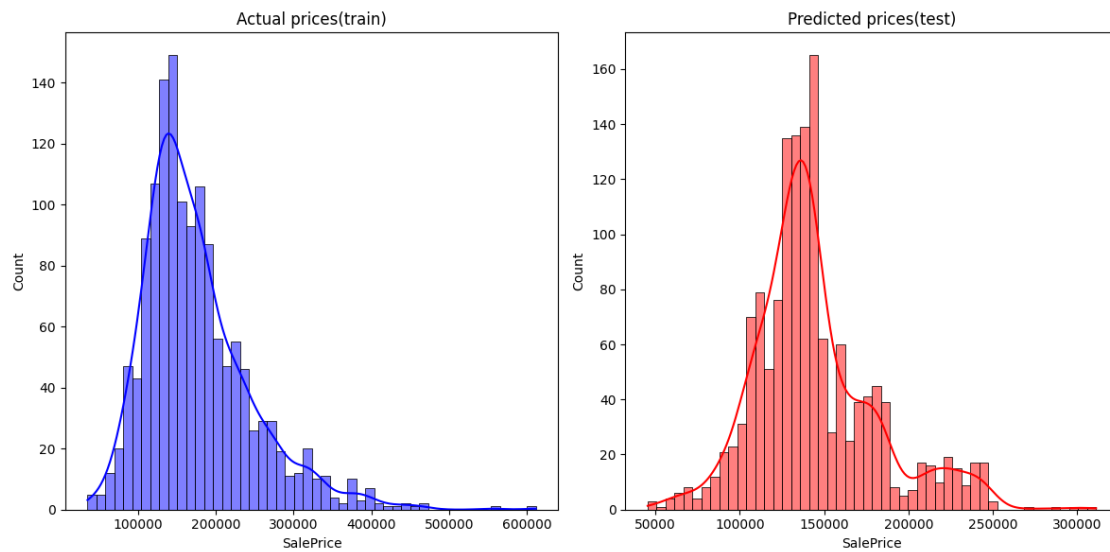
```



```

Train the best model on all data
Predictions for test data
Creating a file
Visualization of Predictions

```



[ ]: