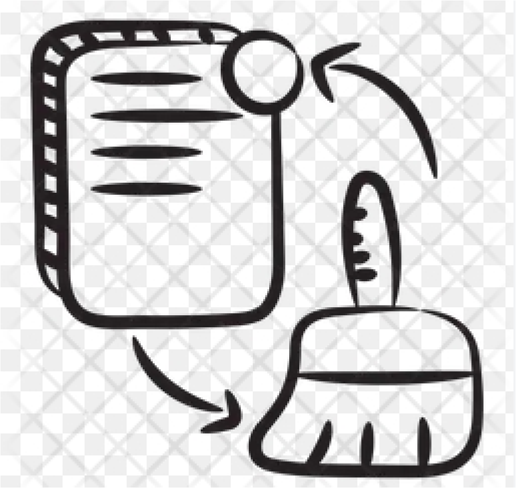
# Surprise Housing Data Analytics - Regularization



# Data Cleaning

Delete unncessary columns

Arrange data in the same data type such as if column has ‘NA’, ‘XXX’ it can be replaced with empty / 0 (if necessary) columns.

# Remove NA from the numeric columns

```
df_train.info()
x_train = x_train.dropna()
x_test = x_test.dropna()

df_train = df_train.dropna()
y_train = df_train.pop('SalePrice')
x_train = df_train
```

# Label Encoding

```python
# Factorize the categorical column to assign integer labels
df['SaleType'], labels = pd.factorize(df['SaleType'])
# Optionally, you can create a dictionary to map the original categories to their integer labels
label_mapping = dict(zip(labels, range(len(labels))))
# Print the mapping of categories to labels
print(label_mapping)

df['SaleCondition'], labels  = pd.factorize(df['SaleCondition'])
label_mapping = dict(zip(labels, range(len(labels))))
# Print the mapping of categories to labels
print(label_mapping)

df['MiscFeature'], labels = pd.factorize(df['MiscFeature'])
label_mapping = dict(zip(labels, range(len(labels))))
# Print the mapping of categories to labels
print(label_mapping)

df['GarageType'], labels = pd.factorize(df['GarageType'])
label_mapping = dict(zip(labels, range(len(labels))))
# Print the mapping of categories to labels
print(label_mapping)
df.info()
```

# Scaling

```
[491]: #Applying the scaling on the test sets
df_test.info()
scaler_vars = ['SaleType','LotArea','LotFrontage','OverallQual','OverallCond','MasVnrArea','TotalBsmtSF','GrLivAr
df_test[scaler_vars] = scaler.transform(df_test[scaler_vars])
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 438 entries, 1436 to 266
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   SaleType      438 non-null    int64
 1   LotArea       438 non-null    int64
 2   LotFrontage   356 non-null    float64
 3   OverallQual   438 non-null    int64
 4   OverallCond   438 non-null    int64
 5   MasVnrArea    434 non-null    float64
 6   TotalBsmtSF   438 non-null    int64
 7   GrLivArea     438 non-null    int64
 8   BsmtUnfSF     438 non-null    int64
```


[522]:
```python
from sklearn import preprocessing

sel_cols = ['LotArea','LotFrontage','OverallQual','OverallCond','MasVnrArea','TotalBsmtSF','GrLivArea','Bs

X = df[sel_cols]

X=X.apply(lambda X: X.fillna(X.median()),axis=0)

# # scale all the columns of the mpg_df. This will produce a numpy array
X_scaled = preprocessing.scale(X)
X_scaled = pd.DataFrame(X_scaled, columns=X.columns)  # ideally the training and test should be

y_scaled = preprocessing.scale(y)
y_scaled = pd.DataFrame(y_scaled, columns=y.columns)  # ideally the training and test should be
```

[523]:
```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_size=0.30, random_state=1)
```

[524]:
```python
from sklearn.linear_model import LinearRegression

regression_model = LinearRegression()
regression_model.fit(X_train, y_train)

for idx, col_name in enumerate(X_train.columns):
```

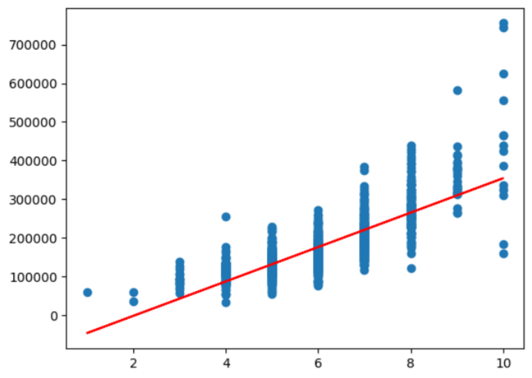
# Housing Data Features

| | |
|---|---|
| **MSSubClass** | Identifies the type of dwelling involved in the sale |
| **LotShape** | General shape of property |
| **OverallQual** | Rates the overall material and finish of the house |
| **OverallCond** | Rates the overall condition of the house |
| **TotalBsmtSF** | Total square feet of basement area |
| **GarageQual** | Garage quality |
| **SaleType** | Type of sale |
| **SaleCondition** | Condition of sale |
| **SalePrice** | Sale Price of the property |

# Residual Analysis

```
[460]: #plt.plot(x_train, 1080.7400 + 169.3134 * x_train, 'r')
       y_train_pred = lr_model.predict(x_train_sm)
```

```
[461]: plt.scatter(x_train,y_train)
       plt.plot(x_train, y_train_pred, 'r')
       plt.show()
```

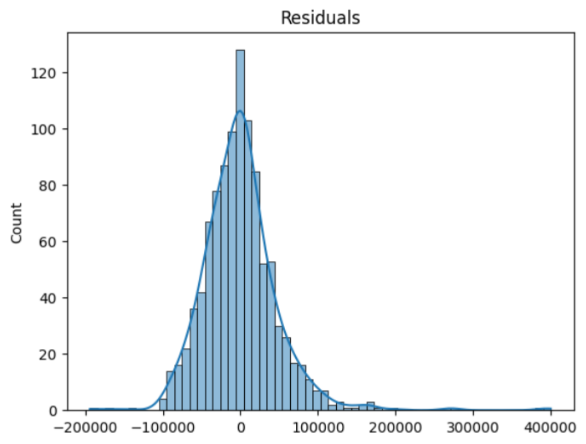



```
[462]: #Residual Analysis
       # f(R) = y_train, y_train_pred
       res = y_train - y_train_pred
       res
```

```
[462]: 318     39060.245700
       239    -63472.053285
       986    -59472.053285
       1416    34963.348746
       390    -13004.352269
                   ...
       802    -31939.754300
       53      75124.843669
       350      8185.843669
       79     -22004.352269
       792     48850.245700
       Length: 1021, dtype: float64
```

```
       plt.title("Residuals")
```

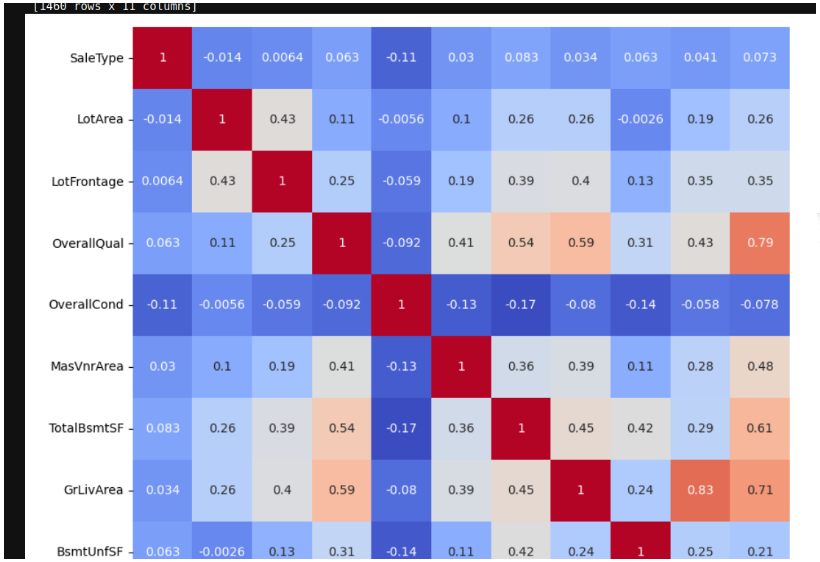
```
[463]: Text(0.5, 1.0, 'Residuals')
```

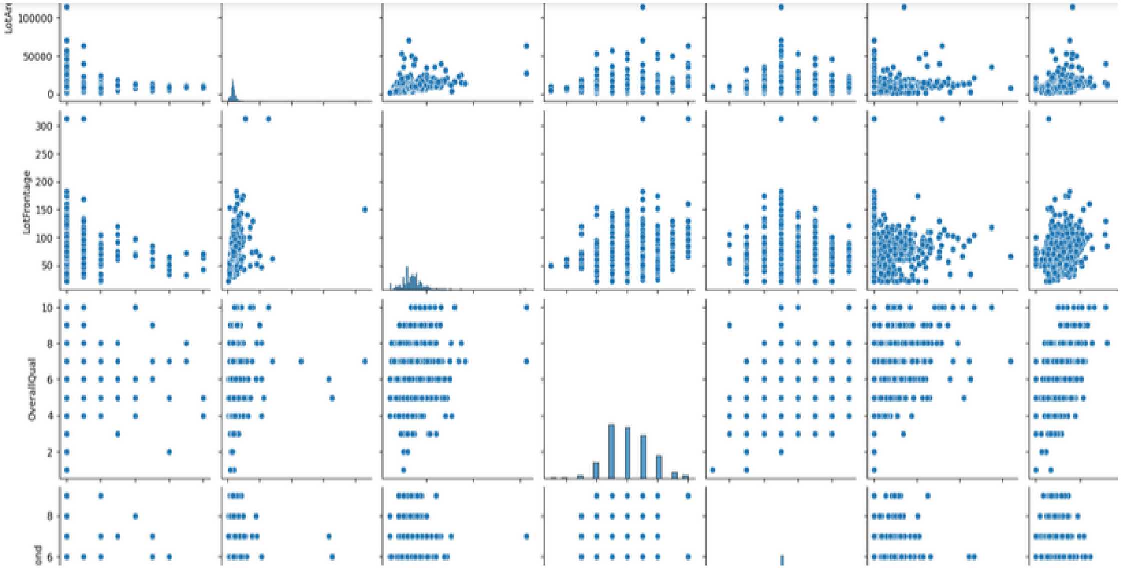


# Multiple Linear Regression

```
#['LotArea','LotFrontage','OverallQual','OverallCond','MasVnrArea','TotalBsmtSF','GrLivArea','BsmtUnfSF','To

#Multiple Linear regression using RFE and (p-value, VIF)
#Data selection . Data already changed and dummies are created, and mappig already done
df1 = df[['SaleType','LotArea','LotFrontage','OverallQual','OverallCond','MasVnrArea','TotalBsmtSF','GrLivAr
print(df1)

plt.figure(figsize=(12, 10))
sns.heatmap(df1.corr(),annot=True, cmap='coolwarm')
sns.pairplot(df1)
plt.show()

# plt.figure(figsize=(20, 12))
# plt.subplot(1,3,1)
# sns.boxplot(x = 'OverallCond', y = 'Saleprice', data = df1)
# plt.subplot(1,3,2)
# sns.boxplot(x = 'TotalBsmtSF', y = 'Saleprice', data = df1)
# plt.subplot(1,3,3)
# sns.boxplot(x = 'TotRmsAbvGrd', y = 'Saleprice', data = df1)
# # Training and test data
# from sklearn.model_selection import train_test_split

# n=10
```

# Correlation & Pairplot



# Training & Test Data

```
[475]: # Training and test data
from sklearn.model_selection import train_test_split

n=10
np.random.seed(0)
df_train, df_test = train_test_split(df1, train_size = 0.7, test_size = 0.3, random_state = 100)

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
#scaler_vars = ['SaleType','LotArea','OverallQual','SalePrice']
scaler_vars = ['SaleType','LotArea','LotFrontage','OverallQual','OverallCond','MasVnrArea','TotalBsmtSF','GrLivArea','BsmtUnfSF
df_train[scaler_vars] = scaler.fit_transform(df_train[scaler_vars])
df_train.head()
```

[475]:

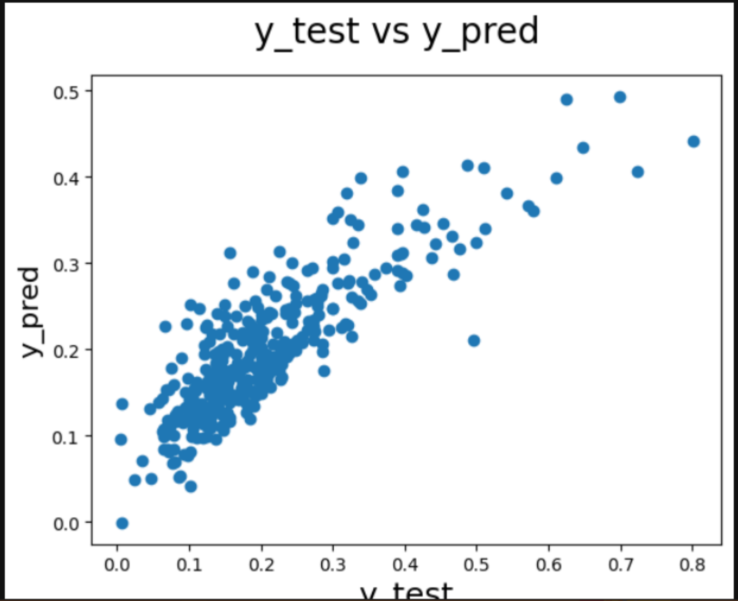
| | SaleType | LotArea | LotFrontage | OverallQual | OverallCond | MasVnrArea | TotalBsmtSF | GrLivArea | BsmtUnfSF | TotRmsAbvGrd | SalePrice |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **210** | 0.0 | 0.019306 | 0.157534 | 0.444444 | 0.625 | 0.00 | 0.141408 | 0.081860 | 0.169521 | 0.181818 | 0.087627 |
| **318** | 0.0 | 0.039403 | 0.236301 | 0.666667 | 0.500 | 0.16 | 0.220458 | 0.424289 | 0.154110 | 0.545455 | 0.312595 |
| **239** | 0.0 | 0.033981 | 0.106164 | 0.555556 | 0.375 | 0.00 | 0.120295 | 0.201576 | 0.274401 | 0.363636 | 0.108457 |
| **986** | 0.0 | 0.017931 | 0.130137 | 0.555556 | 0.875 | 0.00 | 0.079378 | 0.230015 | 0.207620 | 0.181818 | 0.114012 |
| **1416** | 0.0 | 0.046139 | 0.133562 | 0.333333 | 0.625 | 0.00 | 0.127169 | 0.355880 | 0.332620 | 0.727273 | 0.121650 |

```
[485… x_train_new = x_train_rfe.drop(['OverallQual'], axis = 1)
# Adding a constant variable
import statsmodels.api as sm
x_train_lm = sm.add_constant(x_train_new)
lm = sm.OLS(y_train,x_train_lm).fit()
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              SalePrice   R-squared:                       0.601
Model:                            OLS   Adj. R-squared:                  0.597
Method:                 Least Squares   F-statistic:                     139.0
Date:                Wed, 01 May 2024   Prob (F-statistic):          3.84e-159
Time:                        16:28:27   Log-Likelihood:                 1023.8
No. Observations:                 841   AIC:                            -2028.
Df Residuals:                     831   BIC:                            -1980.
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.0063      0.014      0.434      0.665      -0.022       0.035
SaleType         0.0146      0.022      0.678      0.498      -0.028       0.057
LotArea          0.1083      0.069      1.567      0.118      -0.027       0.244
LotFrontage     -0.0464      0.037     -1.255      0.210      -0.119       0.026
OverallCond      0.0077      0.019      0.399      0.690      -0.030       0.046
MasVnrArea       0.1604      0.024      6.797      0.000       0.114       0.207
TotalBsmtSF      0.4985      0.046     10.917      0.000       0.409       0.588
GrLivArea        0.5728      0.048     11.941      0.000       0.479       0.667
```

# Test vs Prediction

```
[493]: #Model Evaluation
       # Plotting y_test and y_pred to understand the spread.
       fig = plt.figure()
       plt.scatter(y_test,y_pred)
       fig.suptitle('y_test vs y_pred', fontsize=20)
       plt.xlabel('y_test', fontsize=18)
       plt.ylabel('y_pred', fontsize=16)

[493]: Text(0, 0.5, 'y_pred')
```



# Recursive Feature Elimination (RFE)

```
memory usage: 95.7 KB
```

[479]:

```python
# Importing RFE and LinearRegression
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression

lm = LinearRegression()
lm.fit(x_train, y_train)

rfe = RFE(lm, n_features_to_select=10)             # running RFE
rfe.fit(x_train, y_train)
```

[479]:

```
▸ RFE
▸ estimator: LinearRegression
  ▸ LinearRegression
```

[480]:

```python
list(zip(x_train.columns,rfe.support_,rfe.ranking_))
```

[480]:

```
[('SaleType', True, 1),
 ('LotArea', True, 1),
 ('LotFrontage', True, 1),
 ('OverallQual', True, 1),
```

# Target

# R2, MSE, RMSE projection

```
y_pred = regression_model.predict(X_train)

r2_score(y_train, y_pred)

# Make predictions on training data
y_pred_train = regression_model.predict(X_train)

# Evaluate the model on training data
r2_train = r2_score(y_train, y_pred_train)
print("R2 Score on Training Data:", r2_train)

rss_train = np.sum(np.square(y_train - y_pred_train))
print("Residual Sum of Squares on Training Data:", rss_train)

mse_train = mean_squared_error(y_train, y_pred_train)
print("Mean Squared Error on Training Data:", mse_train)

rmse_train = mse_train ** 0.5
print("Root Mean Squared Error on Training Data:", rmse_train)

intercept = regression_model.intercept_[0]

print("The intercept for our model is {}".format(intercept))
```

```
R2 Score on Training Data: 0.7305678786021826
Residual Sum of Squares on Training Data: SalePrice    259.724138
dtype: float64
Mean Squared Error on Training Data: 0.2541332075464024
Root Mean Squared Error on Training Data: 0.5041162639177618
The intercept for our model is -0.009340034834956563
```

```
526]: ridge = Ridge(alpha=0.5) #coefficients are prevented to become too big by this alpha value
ridge.fit(X_train,y_train)
for i,col in enumerate(X_train.columns):
```

# Ridge

```
The intercept for our model is -0.009340034834956563
```

```
526…
ridge = Ridge(alpha=0.5)
ridge.fit(X_train,y_train)
for i,col in enumerate(X_train.columns):
    print ("Ridge model coefficients for {} is {}:".format(col,ridge.coef_[0][i]))
```

```
Ridge model coefficients for LotArea is 0.07786746246708784:
Ridge model coefficients for LotFrontage is -0.013170844550486758:
Ridge model coefficients for OverallQual is 0.49602706113954564:
Ridge model coefficients for OverallCond is 0.026967495170341112:
Ridge model coefficients for MasVnrArea is 0.09575602928922848:
Ridge model coefficients for TotalBsmtSF is 0.19152817341305087:
Ridge model coefficients for GrLivArea is 0.19326512506907143:
Ridge model coefficients for BsmtUnfSF is -0.09397672684822486:
Ridge model coefficients for TotRmsAbvGrd is 0.07712244830265556:
Ridge model coefficients for SaleType is 0.01502812161433921:
```

# Lasso

```
Ridge model coefficients for SaleType is 0.01502812161433921:
```

```
[527]: lasso = Lasso(alpha=0.5)
       lasso.fit(X_train,y_train)
       for i,col in enumerate(X_train):
           print ("Lasso model coefficients for {} is {}:".format(col,lasso.coef_[i]))
```

```
Lasso model coefficients for LotArea is 0.0:
Lasso model coefficients for LotFrontage is 0.0:
Lasso model coefficients for OverallQual is 0.223298981755747:
Lasso model coefficients for OverallCond is -0.0:
Lasso model coefficients for MasVnrArea is 0.0:
Lasso model coefficients for TotalBsmtSF is 0.0:
Lasso model coefficients for GrLivArea is 0.04098267082887665:
Lasso model coefficients for BsmtUnfSF is 0.0:
Lasso model coefficients for TotRmsAbvGrd is 0.0:
Lasso model coefficients for SaleType is 0.0:
```

# Score comparison

Regression

```
Lasso model coefficients for SaleType is 0.0:

[528]: print(regression_model.score(X_train, y_train))
       print(regression_model.score(X_test, y_test))

0.7305678786021826
0.8222557634804333
```

Ridge

```
]: print(ridge.score(X_train, y_train))
   print(ridge.score(X_test, y_test))

0.7305677835613592
0.8222184761073315
```

Lasso

```
]: print(lasso.score(X_train, y_train))
   print(lasso.score(X_test, y_test))

0.34427877798440887
0.3577970304469219
```