```
1 import numpy as np
2 import pandas as pd
3
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6
7 from sklearn.model_selection import train_test_split
8 from sklearn.metrics import accuracy_score
9 from sklearn.preprocessing import LabelEncoder
10 from sklearn.preprocessing import StandardScaler
```

```
1 df = pd.read_csv('/content/House Prices.csv')
2 df
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandCor
0	1	60	RL	65.0	8450	Pave	NaN	Reg	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	
...	...	...	...	...	...	...	...	...	
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	

1460 rows × 81 columns

```
1 df.info()
```

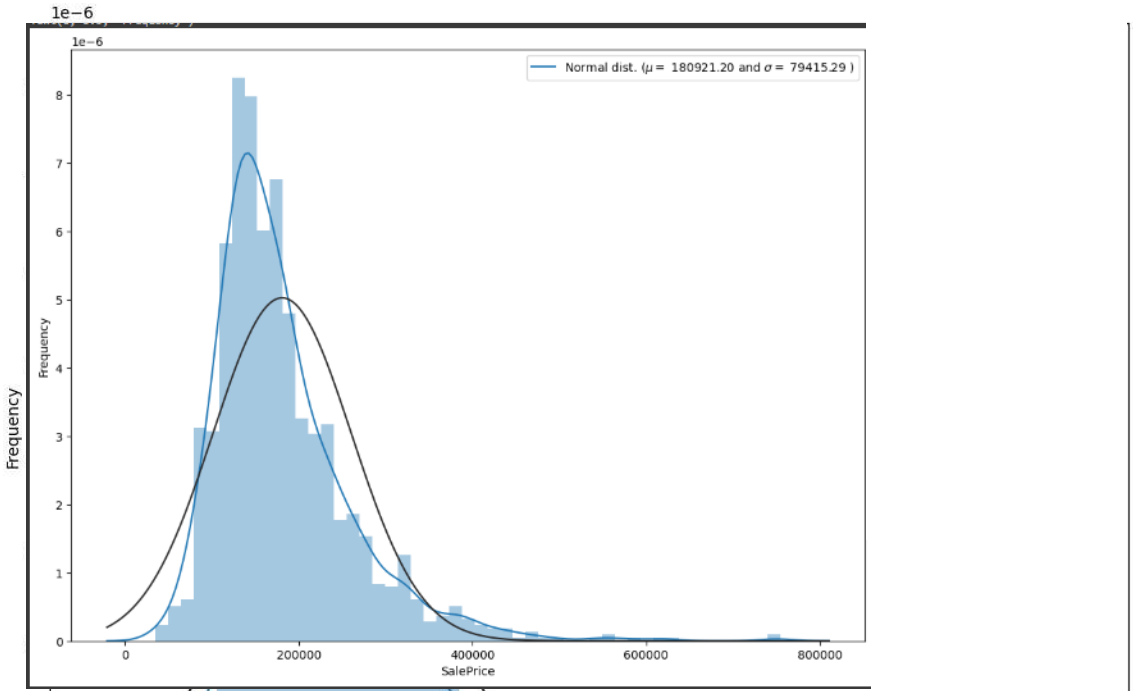
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 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     1452 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
```

```
drop_cols = ['Alley', 'PoolQC', 'Fence', 'MiscFeature', 'FireplaceQu']
```

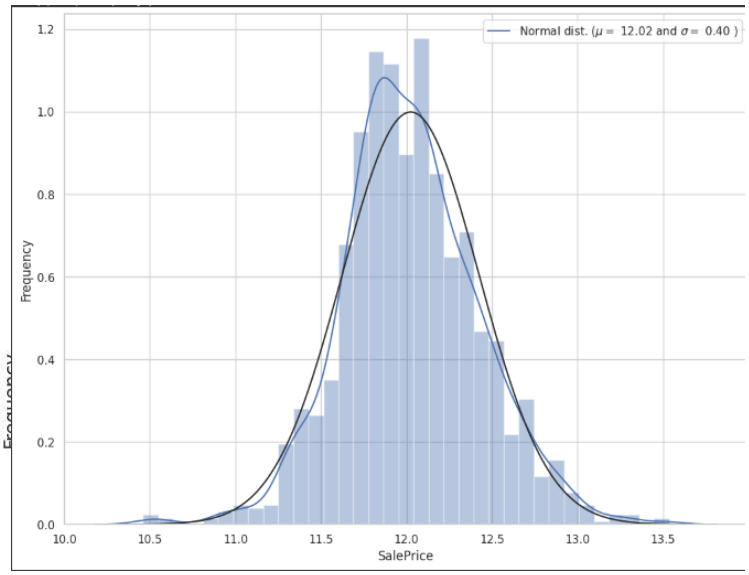
```
from scipy import stats

plt.subplots(figsize = (12, 9))
sns.distplot(df['SalePrice'], fit = stats.norm)

(mu, sigma) = stats.norm.fit(df['SalePrice'])

plt.legend(['Normal dist. ($\mu=$ {:.2f} and $\sigma=$ {:.2f} )'.format(mu, sigma)], loc = 'best')
plt.ylabel('Frequency')
```

```
<ipython-input-5-0da9c2224253>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['SalePrice'], fit = stats.norm)
Text(0, 0.5, 'Frequency')
```




```
1 df['SalePrice'] = np.log1p(df['SalePrice'])
2
3 plt.subplots(figsize = (12, 9))
4 sns.distplot(df['SalePrice'], fit = stats.norm)
5
6 (mu, sigma) = stats.norm.fit(df['SalePrice'])
7
8 plt.legend(['Normal dist. ($\mu=$ {:.2f} and $\sigma=$ {:.2f} )'.format(mu, sigma)], loc = 'best')
9 plt.ylabel('Frequency')
```

```
<ipython-input-26-7b1c475fdc0e>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['SalePrice'], fit = stats.norm)
Text(0, 0.5, 'Frequency')
```

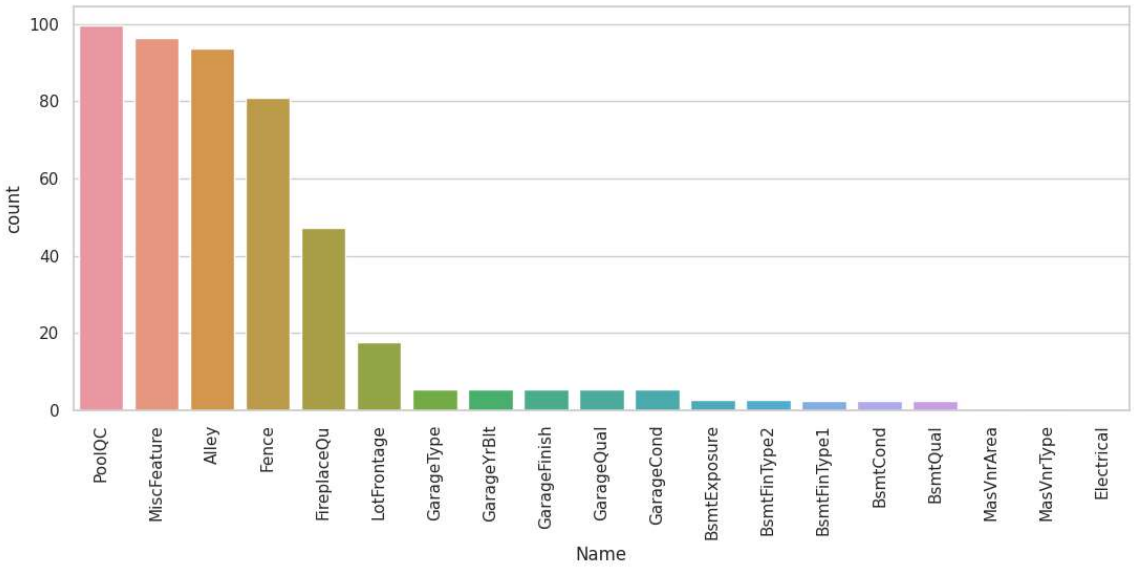



```
1 Isnull = df.isnull().sum()/len(df)*100
2 Isnull = Isnull[Isnull>0]
3 Isnull.sort_values(inplace = True, ascending = False)
4 Isnull
```

```
PoolQC          99.520548
MiscFeature     96.301370
Alley           93.767123
Fence           80.753425
FireplaceQu     47.260274
LotFrontage     17.739726
GarageType       5.547945
GarageYrBlt      5.547945
GarageFinish     5.547945
GarageQual       5.547945
GarageCond       5.547945
BsmtExposure     2.602740
BsmtFinType2     2.602740
BsmtFinType1     2.534247
BsmtCond         2.534247
BsmtQual         2.534247
MasVnrArea       0.547945
MasVnrType       0.547945
Electrical       0.068493
dtype: float64
```

```
1 Isnull = Isnull.to_frame()
2 Isnull.columns = ['count']
3 Isnull.index.names = ['Name']
4
5 Isnull['Name'] = Isnull.index
6
7 plt.figure(figsize = (13, 5))
```

```
sns.set(style = 'whitegrid')
sns.barplot(x = 'Name', y = 'count', data = Isnull)
plt.xticks(rotation = 90)
plt.show()
```




```
df = df.drop(columns = drop_cols)
df
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Util
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	
...	...	...	...	...	...	...	...	...	
1455	1456	60	RL	62.0	7917	Pave	Reg	Lvl	
1456	1457	20	RL	85.0	13175	Pave	Reg	Lvl	
1457	1458	70	RL	66.0	9042	Pave	Reg	Lvl	
1458	1459	20	RL	68.0	9717	Pave	Reg	Lvl	
1459	1460	20	RL	75.0	9937	Pave	Reg	Lvl	

1460 rows × 76 columns

```
from sklearn.impute import SimpleImputer

numeric_columns = df.select_dtypes(include=['float64']).columns
categorical_columns = df.select_dtypes(include=['object']).columns

```

```
numeric_imputer = SimpleImputer(strategy='mean')
categorical_imputer = SimpleImputer(strategy='most_frequent')

# Impute missing values in numeric columns with mean
df[numeric_columns] = numeric_imputer.fit_transform(df[numeric_columns])

# Impute missing values in categorical columns with mode
df[categorical_columns] = categorical_imputer.fit_transform(df[categorical_columns])

df
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContou
0	1	60	RL	65.0	8450	Pave	Grvl	Reg	L
1	2	20	RL	80.0	9600	Pave	Grvl	Reg	L
2	3	60	RL	68.0	11250	Pave	Grvl	IR1	L
3	4	70	RL	60.0	9550	Pave	Grvl	IR1	L
4	5	60	RL	84.0	14260	Pave	Grvl	IR1	L
...	...	...	...	...	...	...	...	...	
1455	1456	60	RL	62.0	7917	Pave	Grvl	Reg	L
1456	1457	20	RL	85.0	13175	Pave	Grvl	Reg	L
1457	1458	70	RL	66.0	9042	Pave	Grvl	Reg	L
1458	1459	20	RL	68.0	9717	Pave	Grvl	Reg	L
1459	1460	20	RL	75.0	9937	Pave	Grvl	Reg	L

1460 rows × 81 columns

```
df.info()
```

```
<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    1460 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          1460 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     1460 non-null   object
 26  MasVnrArea     1460 non-null   float64
 27  ExterQual      1460 non-null   object
 28  ExterCond      1460 non-null   object
 29  Foundation     1460 non-null   object
 30  BsmtQual       1460 non-null   object
 31  BsmtCond       1460 non-null   object
 32  BsmtExposure   1460 non-null   object
 33  BsmtFinType1   1460 non-null   object
 34  BsmtFinSF1     1460 non-null   int64
 35  BsmtFinType2   1460 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     1460 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
```

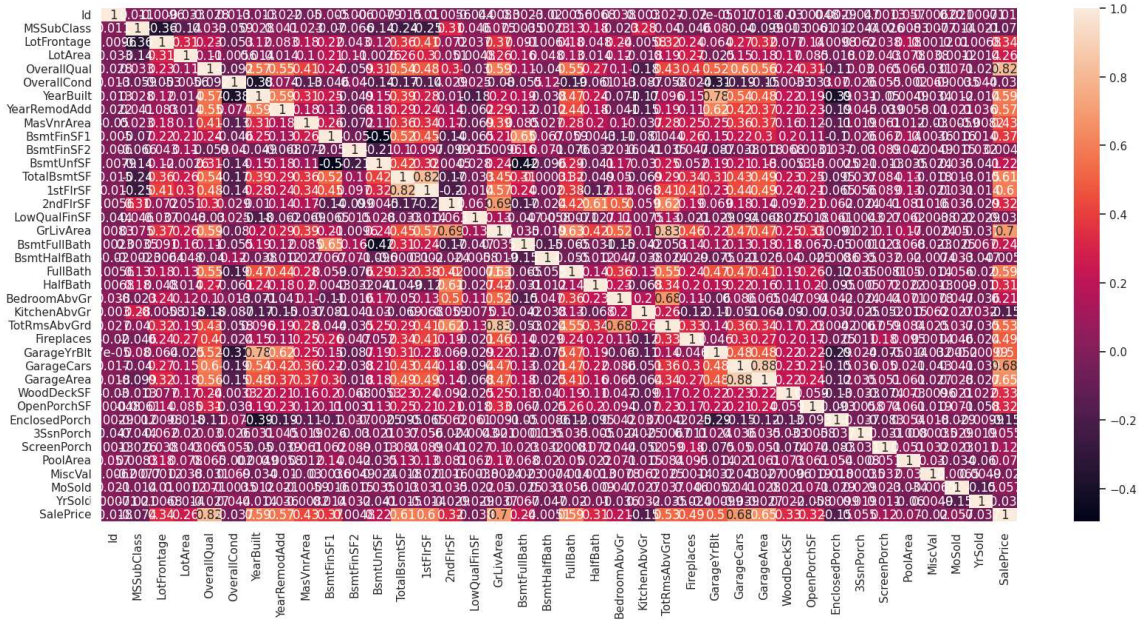
```
df_corr = df.select_dtypes(include = [np.number])
df_corr.shape

df_corr.drop(columns = 'Id')
```

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd
0	60	65.0	8450	7	5	2003	2003
1	20	80.0	9600	6	8	1976	1976
2	60	68.0	11250	7	5	2001	2002
3	70	60.0	9550	7	5	1915	1970
4	60	84.0	14260	8	5	2000	2000
...	...	...	...	...	...	...	...
1455	60	62.0	7917	6	5	1999	2000
1456	20	85.0	13175	6	6	1978	1988
1457	70	66.0	9042	7	9	1941	2006
1458	20	68.0	9717	5	6	1950	1996
1459	20	75.0	9937	5	6	1965	1965

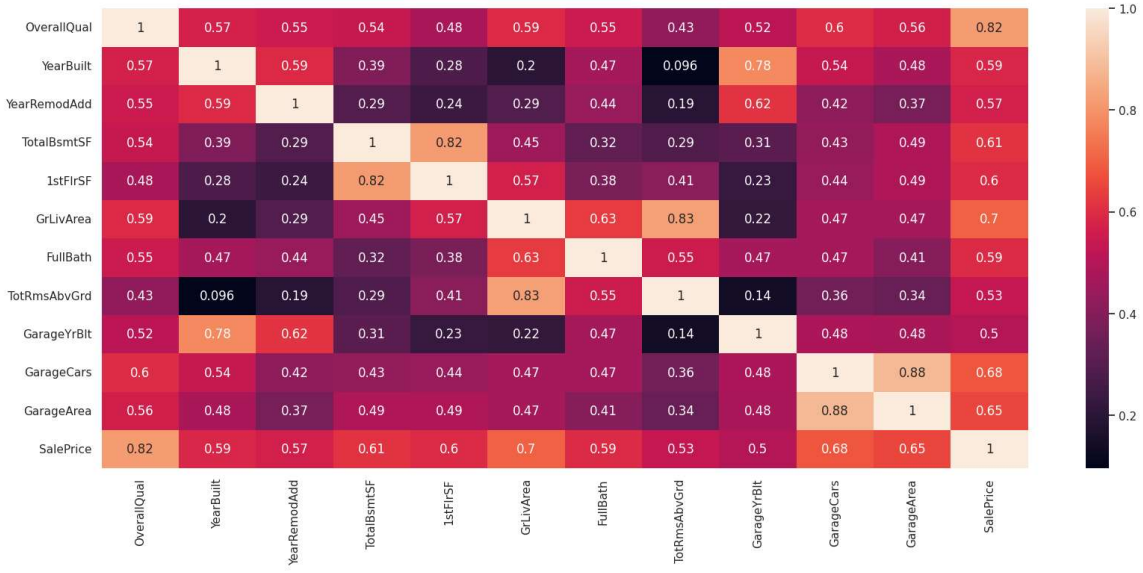
1460 rows × 37 columns

```
corr = df_corr.corr()
plt.subplots(figsize = (20, 9))
sns.heatmap(corr, annot = True)
```

<Axes: >



```
thres = (corr['SalePrice'] > 0.5) | (corr['SalePrice'] < -0.5)
top_feature = corr.index[abs(thres)]

plt.subplots(figsize = (20, 8))
top_corr = df[top_feature].corr()
sns.heatmap(top_corr, annot = True)
plt.show()
```



```
print('Find most important features relative to target')
corr = df.corr()
corr.sort_values(['SalePrice'], ascending = False, inplace = True)
corr.SalePrice
```

```
Find most important features relative to target
<ipython-input-33-aeddc519aba8>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is de
  corr = df.corr()
SalePrice        1.000000
OverallQual      0.817185
GrLivArea        0.700927
GarageCars       0.680625
GarageArea       0.650888
TotalBsmtSF      0.612134
1stFlrSF         0.596981
FullBath         0.594771
YearBuilt        0.586570
YearRemodAdd     0.565608
TotRmsAbvGrd     0.534422
GarageYrBlt      0.500449
Fireplaces       0.489450
MasVnrArea       0.429532
BsmtFinSF1       0.372023
LotFrontage      0.336156
WoodDeckSF       0.334135
OpenPorchSF      0.321053
2ndFlrSF         0.319300
HalfBath         0.313982
LotArea          0.257320
BsmtFullBath     0.236224
BsmtUnfSF        0.221985
BedroomAbvGr     0.209043
ScreenPorch      0.121208
PoolArea         0.069798
MoSold           0.057330
3SsnPorch        0.054900
BsmtFinSF2       0.004832
BsmtHalfBath    -0.005149
Id              -0.017942
MiscVal         -0.020021
OverallCond     -0.036868
YrSold          -0.037263
LowQualFinSF    -0.037963
MSSubClass      -0.073959
KitchenAbvGr    -0.147548
EnclosedPorch   -0.149050
Name: SalePrice, dtype: float64
```

```
df.columns
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
```

```
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

```
df['MiscFeature'] = df['MiscFeature'].fillna('None')
df['Alley'] = df['Alley'].fillna('None')
df['Fence'] = df['Fence'].fillna('None')
df['FireplaceQu'] = df['FireplaceQu'].fillna('None')
```

```
# Garatgetype, GarageFinish, GarageQual and Garagecond these are replacing with none
for col in ['GarageType', 'GarageFinish', 'GarageQual', 'GarageCond']:
  df[col] = df[col].fillna('None')

# GarageYrBlt, GarageArea and GarageCars these are replacing with zero
for col in ['GarageYrBlt', 'GarageArea', 'GarageCars']:
  df[col] = df[col].fillna(int(0))

# # BsmtFinType2, BsmtExposure, BsmtFinType1, BsmtCond, BsmtQual, these are replacing with None
for col in ['BsmtFinType2', 'BsmtExposure', 'BsmtFinType1', 'BsmtCond', 'BsmtQual']:
  df[col] = df[col].fillna('None')

df['Electrical'] = df['Electrical'].fillna(df['Electrical']).mode()[0]
df['MasVnrArea'] = df['MasVnrArea'].fillna(int(0))
df['MasVnrType'] = df['MasVnrType'].fillna('None')
df['LotFrontage'] = df['LotFrontage'].fillna(df['LotFrontage'].mean())
```

```
# df_categ =  df.select_dtypes(include=['object'])
catFeatures = [col for col in df.columns if col in df.select_dtypes(include=['object']).columns]

le = LabelEncoder()

for col in catFeatures:
  df[col] = le.fit_transform(df[col])
```

```
y = df['SalePrice']

X = df.drop('SalePrice', axis = 1).values

y = y.values
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 7)
```

LINEAR REGRESSOR: Accuracy --> 88.99

[ ] ↳ *1 cell hidden*

RANDOM FOREST REGRESSOR: Accuracy --> 89.74

[ ] ↳ *1 cell hidden*

GRADIENT BOOSTING REGRESSOR: (TOP) Accuracy --> 92.11

[ ] ↳ *1 cell hidden*

Summary Report

We have used 3 regression models:

1) `LINEAR REGRESSOR`

2) `RANDOM FOREST REGRESSOR`

3) `GRADIENT BOOSTING REGRESSOR`

but among all, the most accuracy is in the `GradientBoostingRegressor`

✓ 1s completed at 7:33 PM