```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats

train = pd.read_csv('/content/train (1).csv')

train.head()

   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape
\
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


  LandContour Utilities  ... PoolArea PoolQC Fence MiscFeature MiscVal
MoSold  \
0         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
2
1         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
5
2         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
9
3         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
2
4         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
12

  YrSold  SaleType  SaleCondition  SalePrice
0   2008        WD         Normal     208500
1   2007        WD         Normal     181500
2   2008        WD         Normal     223500
3   2006        WD        Abnorml     140000
4   2008        WD         Normal     250000

[5 rows x 81 columns]

train.shape

(1460, 81)

train.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
 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
```

```
train.isnull().sum()
```

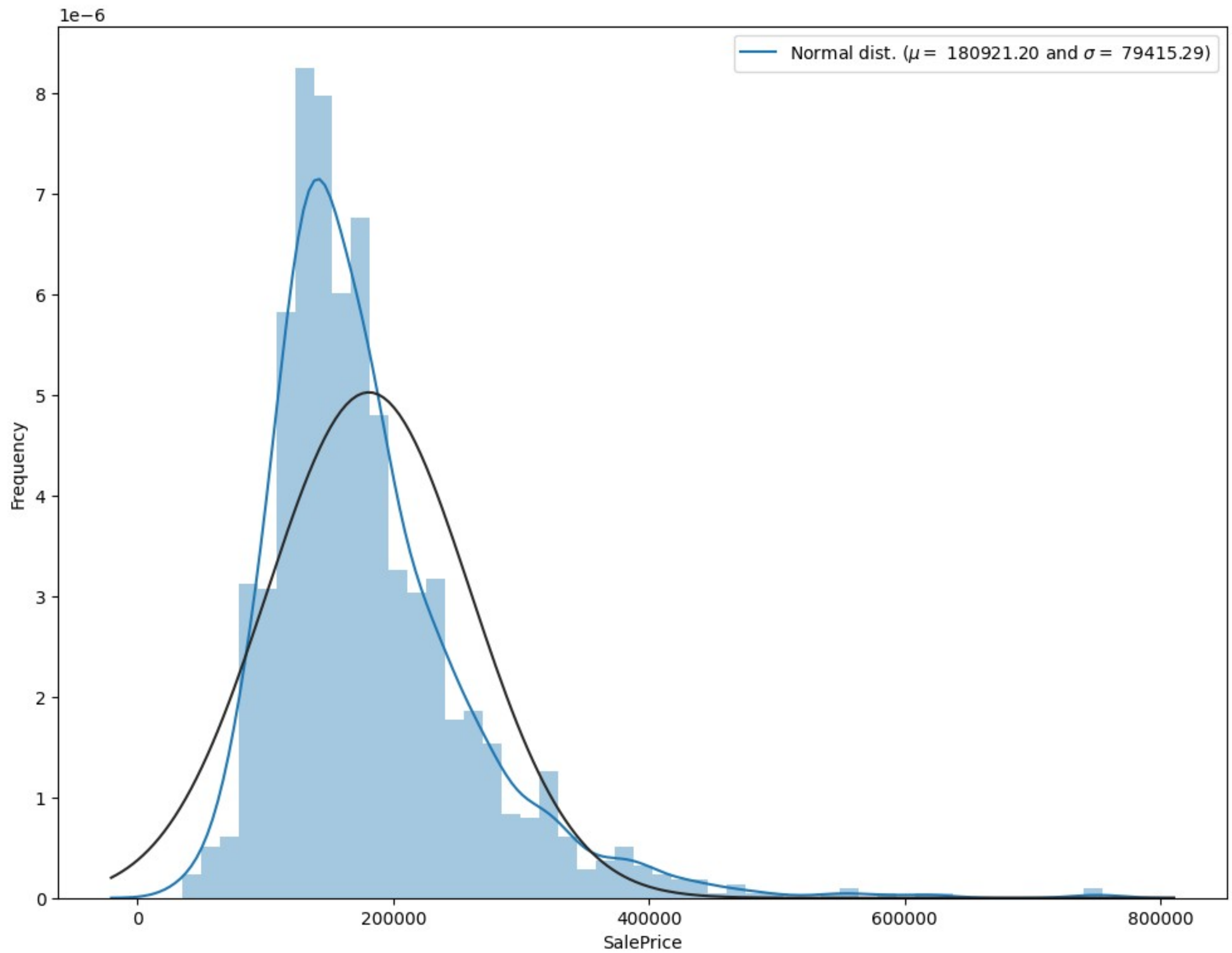
```
Id                 0
MSSubClass         0
MSZoning           0
LotFrontage      259
LotArea            0
                ...
MoSold             0
YrSold             0
SaleType           0
```

```
SaleCondition      0
SalePrice          0
Length: 81, dtype: int64

drop_columns = ['Alley','PoolQC','MiscFeature','Fence']
```

Distribution of Target Variable

A "dist plot" typically refers to a distribution plot, which is a graphical representation of the distribution of a dataset. It helps you understand the underlying probability distribution of the data, providing insights into the central tendency, spread, and shape of the data.

```
plt.subplots(figsize=(12,9))
sns.distplot(train['SalePrice'], fit=stats.norm)
(mu, sigma) = stats.norm.fit(train['SalePrice'])

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

<ipython-input-9-7421450389f9>:2: 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(train['SalePrice'], fit=stats.norm)

Text(0, 0.5, 'Frequency')
```

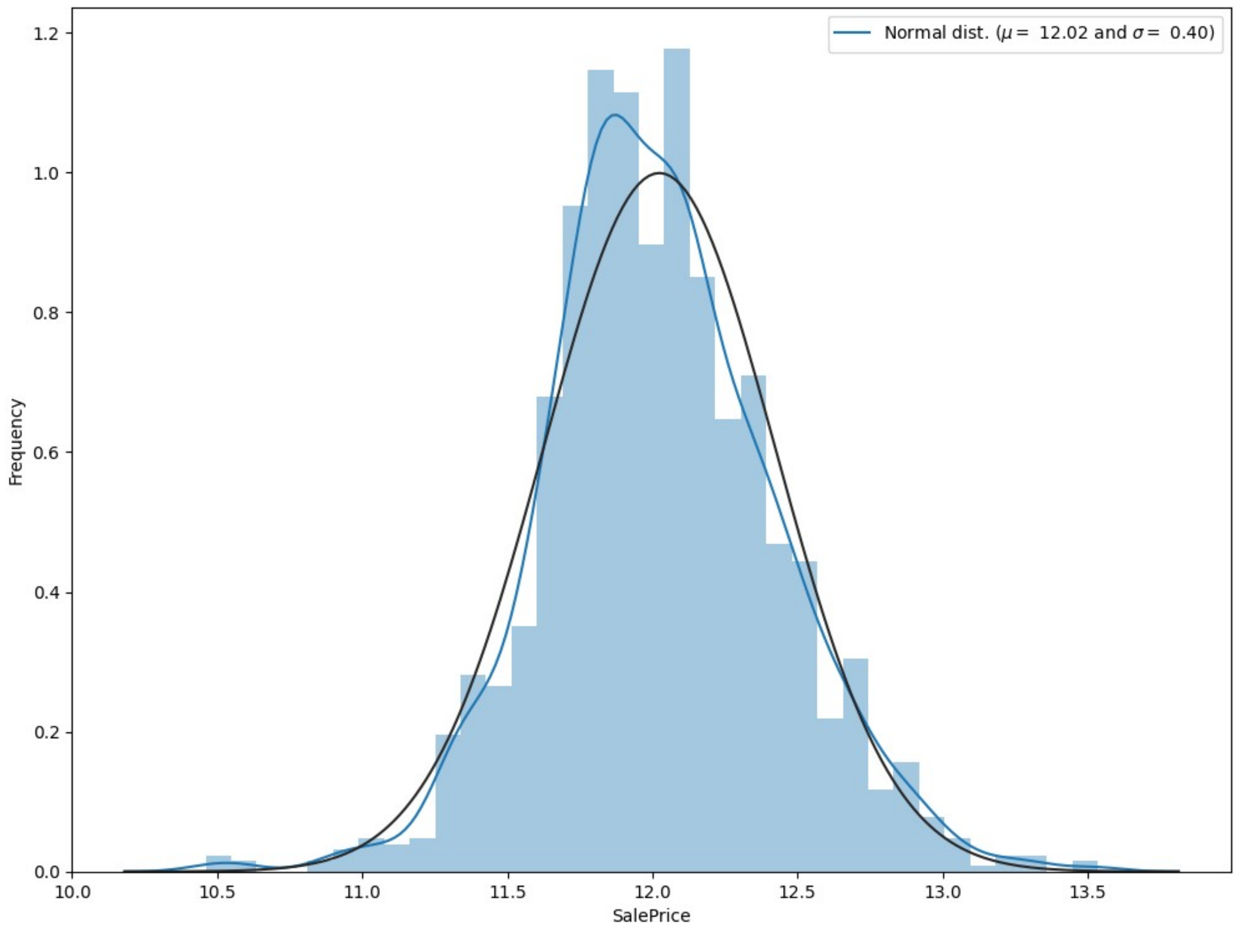


This target varibale is right skewed. Now, we need to tranform this variable and make it normal distribution.

```
# we use log function which is in numpy
train['SalePrice'] = np.log1p(train['SalePrice'])

# Check again for more normal distribution
plt.subplots(figsize=(12,9))
sns.distplot(train['SalePrice'], fit=stats.norm)

# Get the fitted parameters used by the funstion
(mu, sigma) = stats.norm.fit(train['SalePrice'])

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

```
<ipython-input-10-99566006fe8b>:6: 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(train['SalePrice'], fit=stats.norm)

Text(0, 0.5, 'Frequency')
```

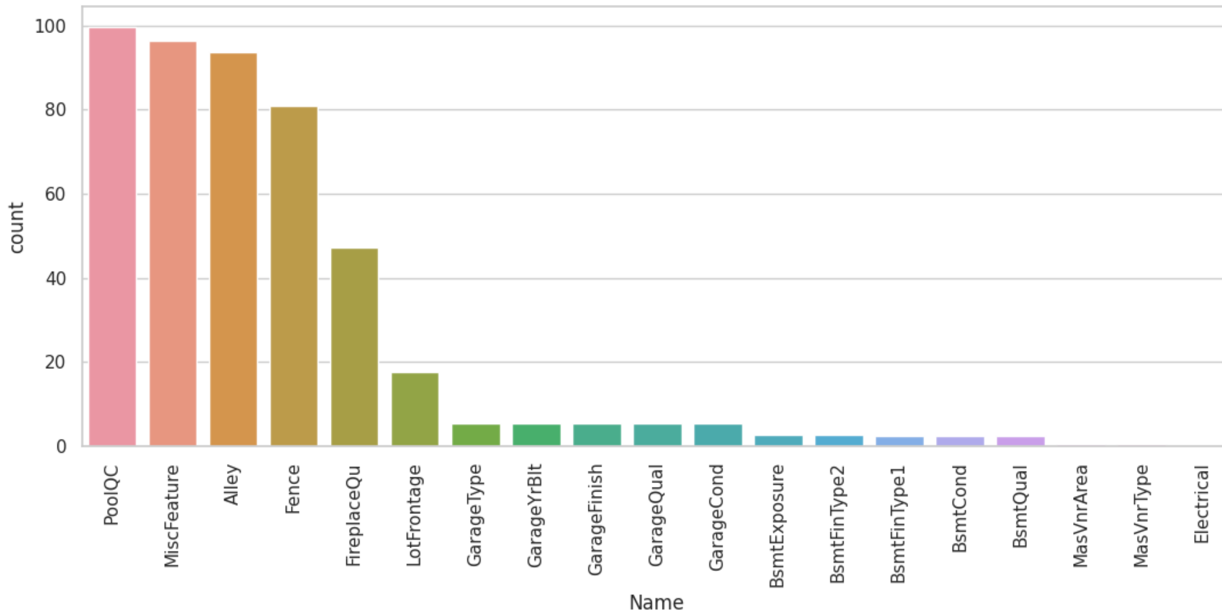


Handling Missing Values

```
Isnull = train.isnull().sum()/len(train)*100
Isnull = Isnull[Isnull>0]
Isnull.sort_values(inplace=True, ascending=False)
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
```

```
# Convert into dataframe
Isnull = Isnull.to_frame()
Isnull.columns = ['count']
Isnull.index.names = ['Name']
#print(Isnull)
Isnull['Name'] = Isnull.index

# plot Missing Values
plt.figure(figsize=(13, 5))
sns.set(style='whitegrid')
sns.barplot(x='Name', y='count', data=Isnull)
plt.xticks(rotation = 90)
plt.show()
```




```
train = train.drop(columns= ['PoolQC','MiscFeature','Alley','Fence'])
train

        Id  MSSubClass MSZoning  LotFrontage  LotArea Street LotShape
\
0        1          60       RL         65.0     8450   Pave      Reg

1        2          20       RL         80.0     9600   Pave      Reg

2        3          60       RL         68.0    11250   Pave      IR1

3        4          70       RL         60.0     9550   Pave      IR1

4        5          60       RL         84.0    14260   Pave      IR1

...    ...         ...      ...          ...      ...    ...      ...

1455  1456          60       RL         62.0     7917   Pave      Reg

1456  1457          20       RL         85.0    13175   Pave      Reg

1457  1458          70       RL         66.0     9042   Pave      Reg

1458  1459          20       RL         68.0     9717   Pave      Reg

1459  1460          20       RL         75.0     9937   Pave      Reg


     LandContour Utilities LotConfig  ... EnclosedPorch 3SsnPorch
ScreenPorch  \
0            Lvl    AllPub    Inside  ...             0         0
```

```
0
1             Lvl    AllPub       FR2  ...            0         0
0
2             Lvl    AllPub    Inside  ...            0         0
0
3             Lvl    AllPub    Corner  ...          272         0
0
4             Lvl    AllPub       FR2  ...            0         0
0
...           ...       ...       ...  ...          ...       ...
...
1455          Lvl    AllPub    Inside  ...            0         0
0
1456          Lvl    AllPub    Inside  ...            0         0
0
1457          Lvl    AllPub    Inside  ...            0         0
0
1458          Lvl    AllPub    Inside  ...          112         0
0
1459          Lvl    AllPub    Inside  ...            0         0
0

     PoolArea MiscVal MoSold  YrSold  SaleType  SaleCondition
SalePrice
0           0       0      2    2008        WD         Normal
12.247699
1           0       0      5    2007        WD         Normal
12.109016
2           0       0      9    2008        WD         Normal
12.317171
3           0       0      2    2006        WD        Abnorml
11.849405
4           0       0     12    2008        WD         Normal
12.429220
...       ...     ...    ...     ...       ...            ...        .
..
1455        0       0      8    2007        WD         Normal
12.072547
1456        0       0      2    2010        WD         Normal
12.254868
1457        0    2500      5    2010        WD         Normal
12.493133
1458        0       0      4    2010        WD         Normal
11.864469
1459        0       0      6    2008        WD         Normal
11.901590

[1460 rows x 77 columns]
```

```python
from sklearn.impute import SimpleImputer

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

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

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

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

train
```

```
        Id  MSSubClass MSZoning  LotFrontage  LotArea Street LotShape
\
0        1          60       RL         65.0     8450   Pave      Reg

1        2          20       RL         80.0     9600   Pave      Reg

2        3          60       RL         68.0    11250   Pave      IR1

3        4          70       RL         60.0     9550   Pave      IR1

4        5          60       RL         84.0    14260   Pave      IR1

...    ...         ...      ...          ...      ...    ...      ...

1455  1456          60       RL         62.0     7917   Pave      Reg

1456  1457          20       RL         85.0    13175   Pave      Reg

1457  1458          70       RL         66.0     9042   Pave      Reg

1458  1459          20       RL         68.0     9717   Pave      Reg

1459  1460          20       RL         75.0     9937   Pave      Reg


     LandContour Utilities LotConfig  ... EnclosedPorch 3SsnPorch
ScreenPorch  \
0            Lvl    AllPub    Inside  ...             0         0
0
1            Lvl    AllPub       FR2  ...             0         0
0
2            Lvl    AllPub    Inside  ...             0         0
0
```

```
3            Lvl    AllPub    Corner  ...           272         0
0
4            Lvl    AllPub       FR2  ...             0         0
0
...          ...       ...       ...  ...           ...       ...
...
1455         Lvl    AllPub    Inside  ...             0         0
0
1456         Lvl    AllPub    Inside  ...             0         0
0
1457         Lvl    AllPub    Inside  ...             0         0
0
1458         Lvl    AllPub    Inside  ...           112         0
0
1459         Lvl    AllPub    Inside  ...             0         0
0

     PoolArea MiscVal MoSold  YrSold  SaleType  SaleCondition
SalePrice
0           0       0      2    2008        WD         Normal
12.247699
1           0       0      5    2007        WD         Normal
12.109016
2           0       0      9    2008        WD         Normal
12.317171
3           0       0      2    2006        WD        Abnorml
11.849405
4           0       0     12    2008        WD         Normal
12.429220
...       ...     ...    ...     ...       ...            ...          .
..
1455        0       0      8    2007        WD         Normal
12.072547
1456        0       0      2    2010        WD         Normal
12.254868
1457        0    2500      5    2010        WD         Normal
12.493133
1458        0       0      4    2010        WD         Normal
11.864469
1459        0       0      6    2008        WD         Normal
11.901590

[1460 rows x 77 columns]
```

```
train.isnull().sum()
```

```
Id                 0
MSSubClass         0
MSZoning           0
LotFrontage        0
```

```
LotArea            0
                  ..
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 77, dtype: int64
```

```
# Separate variable into new dataframe from original dataframe which
has only numerical values
# there is 38 numerical attributes from 81 attributes

train_corr = train.select_dtypes(include=[np.number])

train_corr.shape
```

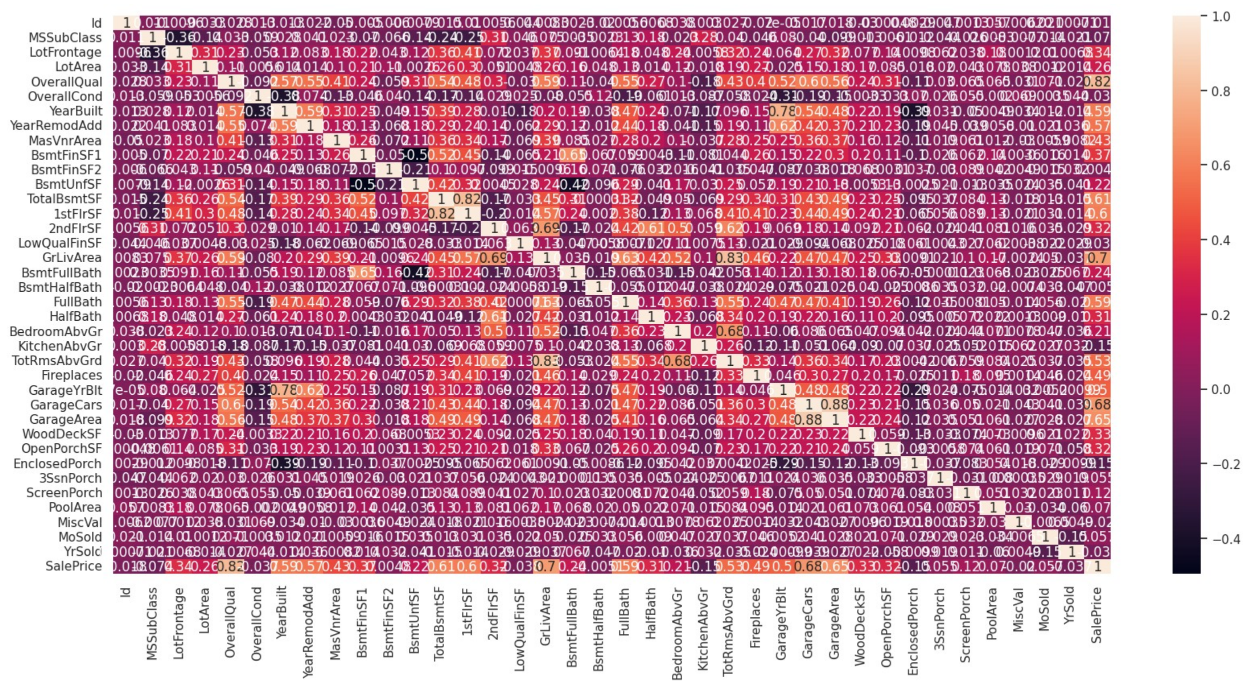
```
(1460, 38)
```

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

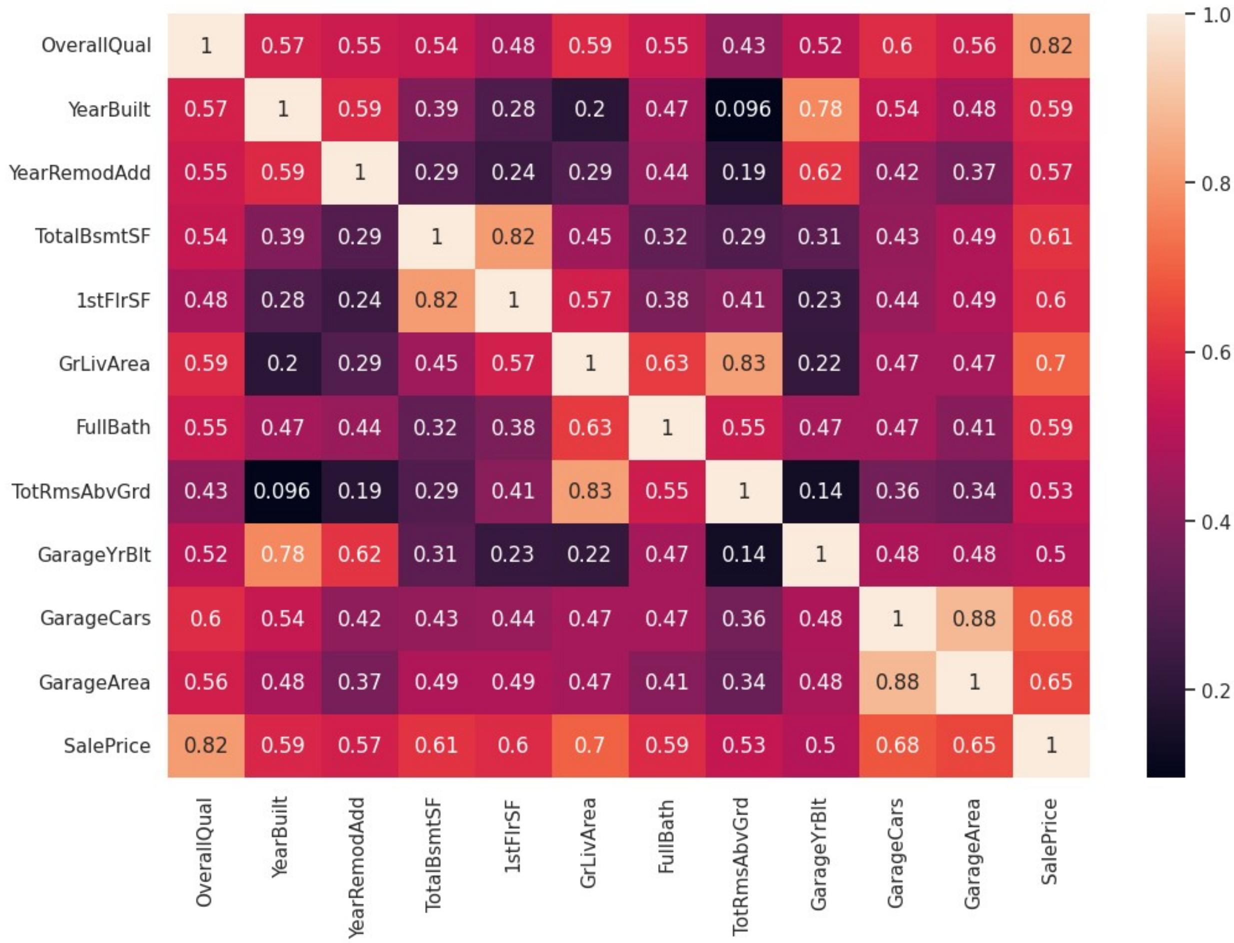
```
      MSSubClass  LotFrontage  LotArea  OverallQual  OverallCond
YearBuilt  \
0             60         65.0     8450            7            5
2003
1             20         80.0     9600            6            8
1976
2             60         68.0    11250            7            5
2001
3       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

      YearRemodAdd  MasVnrArea  BsmtFinSF1  BsmtFinSF2  ...
WoodDeckSF  \
0             2003       196.0         706           0  ...
0
1             1976         0.0         978           0  ...
```

```
298
2             2002       162.0          486           0  ...
0
3             1970         0.0          216           0  ...
0
4             2000       350.0          655           0  ...
192
...            ...         ...          ...         ...  ...          ..
.
1455          2000         0.0            0           0  ...
0
1456          1988       119.0          790         163  ...
349
1457          2006         0.0          275           0  ...
0
1458          1996         0.0           49        1029  ...
366
1459          1965         0.0          830         290  ...
736

      OpenPorchSF  EnclosedPorch  3SsnPorch  ScreenPorch  PoolArea
MiscVal  \
0              61              0          0            0         0
0
1               0              0          0            0         0
0
2              42              0          0            0         0
0
3              35            272          0            0         0
0
4              84              0          0            0         0
0
...           ...            ...        ...          ...       ...
...
1455           40              0          0            0         0
0
1456            0              0          0            0         0
0
1457           60              0          0            0         0
2500
1458            0            112          0            0         0
0
1459           68              0          0            0         0
0

      MoSold  YrSold  SalePrice
0          2    2008  12.247699
1          5    2007  12.109016
2          9    2008  12.317171
```

```
3           2    2006  11.849405
4          12    2008  12.429220
...       ...     ...        ...
1455        8    2007  12.072547
1456        2    2010  12.254868
1457        5    2010  12.493133
1458        4    2010  11.864469
1459        6    2008  11.901590

[1460 rows x 37 columns]

# Coralation plot
corr = train_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=(12, 8))
top_corr = train[top_feature].corr()
sns.heatmap(top_corr, annot=True)
plt.show()
```




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

```
Find most important features relative to target
```

```
<ipython-input-21-785333892f23>:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  corr = train.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
```

```
#train['MicFeature'] = train['MiscFeature'].fillna('None')
#train['Alley'] = train['Alley'].fillna('None')
#train['Fence'] = train['Fence'].fillna('None')
train['FireplaceQu'] = train['FireplaceQu'].fillna('None')

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

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

#BsmtFinType2, BsmtExposure, BsmtFinType1, BsmtCond, BsmtQual these
are relacing with None
for col in ('BsmtFinType2', 'BsmtExposure', 'BsmtFinType1',
'BsmtCond', 'BsmtQual'):
```

```
  train[col] = train[col].fillna('None')

train['Electrical'] =
train['Electrical'].fillna(train['Electrical']).mode()[0]
train['MasVnrArea'] = train['MasVnrArea'].fillna(int(0))
train['MasVnrType'] = train['MasVnrType'].fillna('None')
train['LotFrontage'] =
train['LotFrontage'].fillna(train['LotFrontage'].mean())
#train = train.drop('PoolQC', axis = 1)

# Extracting categorical columns:
catFeatures = [col for col in train.columns if col in
train.select_dtypes(include = object).columns]

from sklearn.preprocessing import LabelEncoder

# Encoding Categorical Data
labelEncode = LabelEncoder()

# Iterating Over each categorical features:
for col in catFeatures:
  # storing its numerical value:
  train[col] = labelEncode.fit_transform(train[col])

y = train['SalePrice']
#Take their values in X and y
X = train.drop('SalePrice', axis = 1).values
y = y.values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=7)
```

Models

LinearRegression Model --> Accuracy = 89.66

RandomForestRegressor Model --> Accuracy = 89.64

GradientBoostingRegressor Model --> Accuracy = 91.93

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
# Fit the model
model.fit(X_train,y_train)

# Prediction
print("Predict value " + str(model.predict([X_test[142]])))
print("Real value" + str(y_test[142]))
```

```
Predict value [11.66265169]
Real value11.767187766223199

# Score?Accuracy
print("Accuracy -->", model.score(X_test, y_test)*100)

Accuracy --> 89.66933781714985
```

RandomForestRegressor Model --> Accuracy = 89.64

```
# Train the model
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=1000)
# Fit
model.fit(X_train, y_train)

# Prediction
print("Predict value " + str(model.predict([X_test[142]])))
print("Real value " + str(y_test[142]))

print("Accuracy --> ", model.score(X_test, y_test)*100)

Predict value [11.72839851]
Real value 11.767187766223199
Accuracy -->  89.64105055311022
```

GradientBoostingRegressor Model --> Accuracy = 91.93

```
# Train the model
from sklearn.ensemble import GradientBoostingRegressor
GBR = GradientBoostingRegressor(n_estimators=100, max_depth=4)

# Fit
GBR.fit(X_train, y_train)

# Prediction
print("Predict value " + str(model.predict([X_test[142]])))
print("Real value " + str(y_test[142]))

print("Accuracy --> ", GBR.score(X_test, y_test)*100)

Predict value [11.71870829]
Real value 11.767187766223199
Accuracy -->  91.9380681872603
```

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