1.Problem Analysis

In the estate market, each house has a selling price based on its different features and conditions. Based on the given train dataset, test dataset and description of each features, this project aims to predict the provided house sale price on the market.

2.Get Data and Data cleaning

Import the pandas library and load train.csv file into Python. A quick review of the input data is as follows: there are 81 different features in column and 1460 observations samples in row:

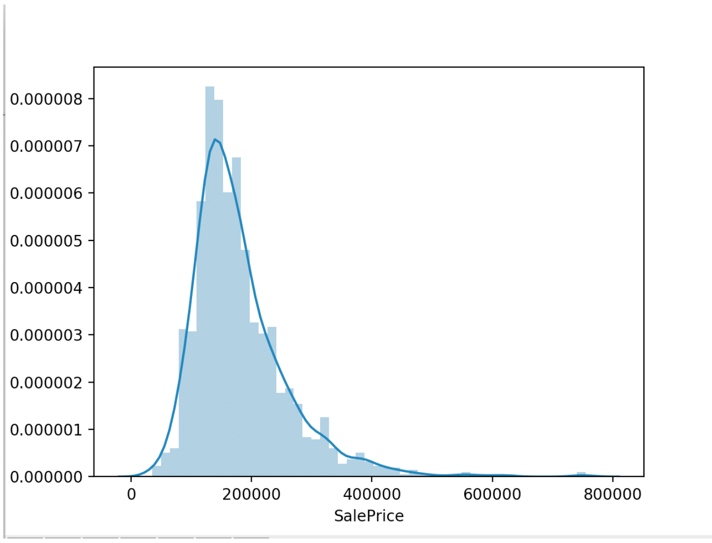
```
>>> df_train
        Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape LandContour Utilities  ... PoolArea PoolQC  Fence MiscFeature MiscVal MoSold YrSold  SaleType  SaleCondition  SalePric
e
0        1          60       RL         65.0     8450   Pave   NaN      Reg         Lvl    AllPub  ...        0    NaN    NaN         NaN       0      2   2008        WD         Normal     20850
0
1        2          20       RL         80.0     9600   Pave   NaN      Reg         Lvl    AllPub  ...        0    NaN    NaN         NaN       0      5   2007        WD         Normal     18150
0
2        3          60       RL         68.0    11250   Pave   NaN      IR1         Lvl    AllPub  ...        0    NaN    NaN         NaN       0      9   2008        WD         Normal     22350
0
3        4          70       RL         60.0     9550   Pave   NaN      IR1         Lvl    AllPub  ...        0    NaN    NaN         NaN       0      2   2006        WD        Abnorml     14000
0
4        5          60       RL         84.0    14260   Pave   NaN      IR1         Lvl    AllPub  ...        0    NaN    NaN         NaN       0     12   2008        WD         Normal     25000
0
...    ...         ...      ...          ...      ...    ...   ...      ...         ...       ...  ...      ...    ...    ...         ...     ...    ...    ...       ...            ...        ..
.
1455  1456          60       RL         62.0     7917   Pave   NaN      Reg         Lvl    AllPub  ...        0    NaN    NaN         NaN       0      8   2007        WD         Normal     17500
0
1456  1457          20       RL         85.0    13175   Pave   NaN      Reg         Lvl    AllPub  ...        0    NaN  MnPrv         NaN       0      2   2010        WD         Normal     21000
0
1457  1458          70       RL         66.0     9042   Pave   NaN      Reg         Lvl    AllPub  ...        0    NaN  GdPrv        Shed    2500      5   2010        WD         Normal     26650
0
1458  1459          20       RL         68.0     9717   Pave   NaN      Reg         Lvl    AllPub  ...        0    NaN    NaN         NaN       0      4   2010        WD         Normal     14212
5
1459  1460          20       RL         75.0     9937   Pave   NaN      Reg         Lvl    AllPub  ...        0    NaN    NaN         NaN       0      6   2008        WD         Normal     14750
0

[1460 rows x 81 columns]
```

All column headers in the train dataset can be observed as below:

```
>>> df_train=pd.read_csv('train.csv')
>>> print(df_train.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')
>>>
```

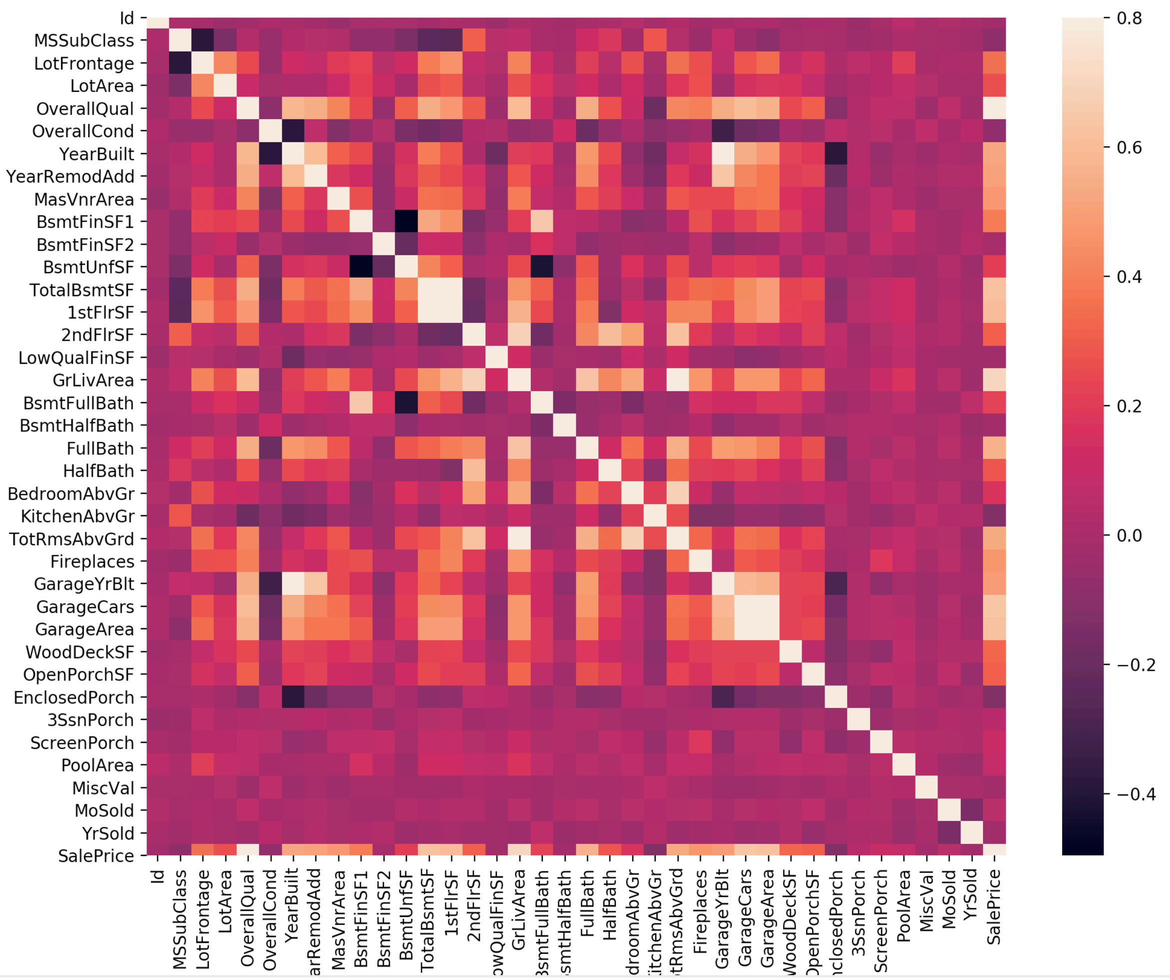
A descriptive statistic summary of SalePrice and distribution plot in histogram are as below:

```
>>> df_train['SalePrice'].describe()
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
>>>
```



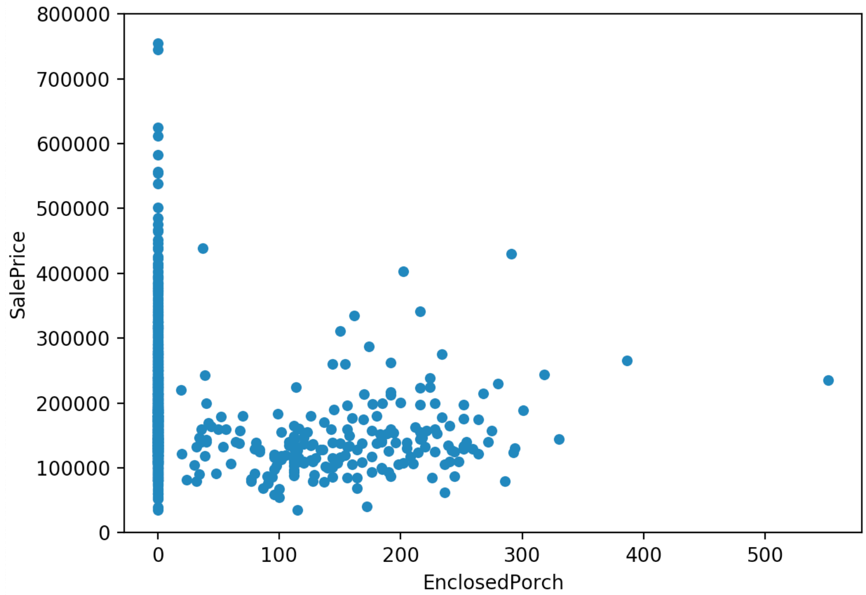
From the statistics and SalePrice distribution perspective, the mean and max value of SalePrice is about $180921 and $755000, and most of the SalePrice values are between $100000 to $400000.

Through correlation matrix, the correlation between each two variables can be observed:



Based on the above Heatmap figure, a low correlation can be observed between feature EnclosedPorch and SalePrice. A further scatter plot is applied as below which further shows that there is no clear linear correlation between this two feature. EnclosedPorch will be kept until further confirmation.

This strategy can be applied to check whether a feature is highly related with SalePrice or not.



Duplicated observation in sample dataset can be showed by using duplicated() function. The output will be Empty DataFrame when there is no duplicated rows in the dataset.

```
>>> df_train[df_train.duplicated()==True]
Empty DataFrame
Columns: [Id, MSSubClass, MSZoning, LotFrontage, LotArea, Street, Alley, LotShape, LandContour, Utilities, LotConfig, LandSlope, Neighborhood, Condition1, Condition2, BldgType, HouseStyle, Overa
llQual, OverallCond, YearBuilt, YearRemodAdd, RoofStyle, RoofMatl, Exterior1st, Exterior2nd, MasVnrType, MasVnrArea, ExterQual, ExterCond, Foundation, BsmtQual, BsmtCond, BsmtExposure, BsmtFinTy
pe1, 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, Paved
Drive, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, PoolQC, Fence, MiscFeature, MiscVal, MoSold, YrSold, SaleType, SaleCondition, SalePrice]
Index: []
>>>
```

There are mainly 3 data types in the train dataset, which are int64, object and float64.

The detailed info can be observed as below snapshots:

```
>>> res = df_train.dtypes
>>> print(res[res == np.dtype('int64')])
Id               int64
MSSubClass       int64
LotArea          int64
OverallQual      int64
OverallCond      int64
YearBuilt        int64
YearRemodAdd     int64
BsmtFinSF1       int64
BsmtFinSF2       int64
BsmtUnfSF        int64
TotalBsmtSF      int64
1stFlrSF         int64
2ndFlrSF         int64
LowQualFinSF     int64
GrLivArea        int64
BsmtFullBath     int64
BsmtHalfBath     int64
FullBath         int64
HalfBath         int64
BedroomAbvGr     int64
KitchenAbvGr     int64
TotRmsAbvGrd     int64
Fireplaces       int64
GarageCars       int64
GarageArea       int64
WoodDeckSF       int64
OpenPorchSF      int64
EnclosedPorch    int64
3SsnPorch        int64
ScreenPorch      int64
PoolArea         int64
MiscVal          int64
MoSold           int64
YrSold           int64
SalePrice        int64
dtype: object
>>>
```

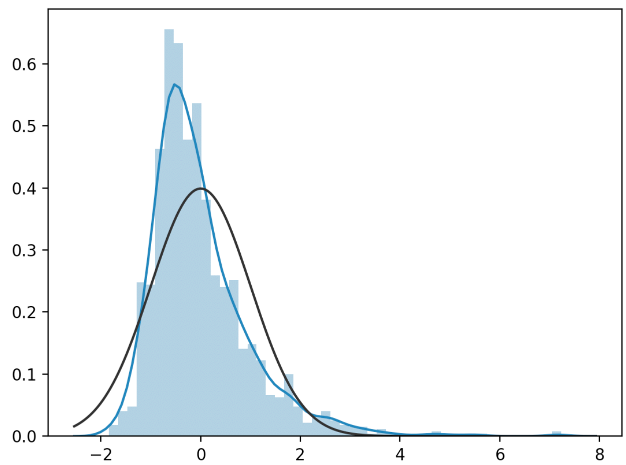
```
>>> print(res[res == np.dtype('float64')])
LotFrontage    float64
MasVnrArea     float64
GarageYrBlt    float64
dtype: object
>>>
```

```
>>> print(res[res==np.dtype('object')])
MSZoning         object
Street           object
Alley            object
LotShape         object
LandContour      object
Utilities        object
LotConfig        object
LandSlope        object
Neighborhood     object
Condition1       object
Condition2       object
BldgType         object
HouseStyle       object
RoofStyle        object
RoofMatl         object
Exterior1st      object
Exterior2nd      object
MasVnrType       object
ExterQual        object
ExterCond        object
Foundation       object
BsmtQual         object
BsmtCond         object
BsmtExposure     object
BsmtFinType1     object
BsmtFinType2     object
Heating          object
HeatingQC        object
CentralAir       object
Electrical       object
KitchenQual      object
Functional       object
FireplaceQu      object
GarageType       object
GarageFinish     object
GarageQual       object
GarageCond       object
PavedDrive       object
PoolQC           object
Fence            object
MiscFeature      object
SaleType         object
SaleCondition    object
dtype: object
>>>
```

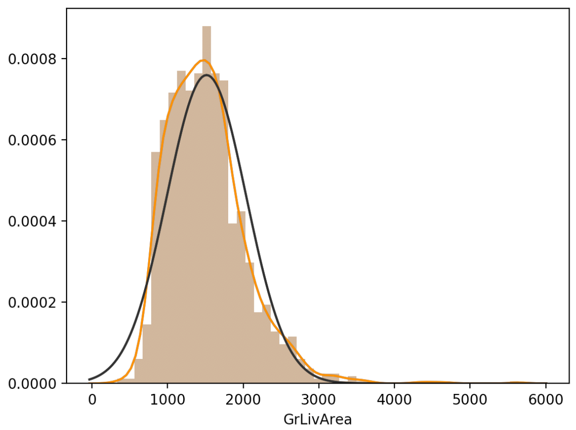
The unique values of each variable can also be observed through unique() function as below example:

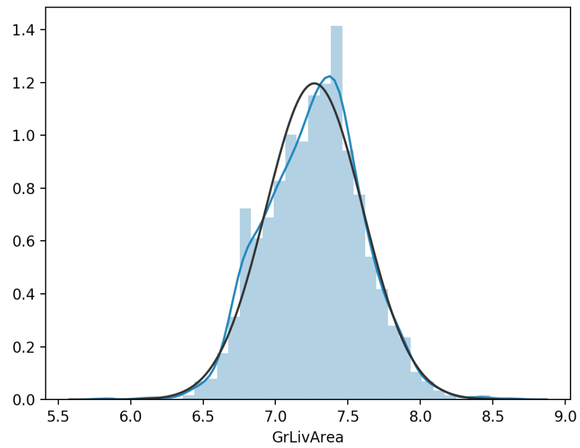
```
>>> print(df_train["LotConfig"].unique())
['Inside' 'FR2' 'Corner' 'CulDSac' 'FR3']
>>>
```

For numerical variable, feature scaling can be applied to fit and transform data into normal distribution .



Data transformation: the distribution of GrLivArea column can be observed with fitting normalisation format as the left figure. GrLivArea column can also be calculated through log() function and a new distribution picture can be plotted as the right figure.





About missing data processing:

Sum all null values of each column and calculate the percentage of missing values for each column, then the first 20 highest percentages of

missing values for columns can be found out and further processing can be applied based on different situations, for example, for any column whose missing percentage is grater than 15% will be dropped.

```
>>> total = df_train.isnull().sum().sort_values(ascending=False)
>>> total
PoolQC          1453
MiscFeature     1406
Alley           1369
Fence           1179
FireplaceQu      690
                ...
CentralAir         0
SaleCondition      0
Heating            0
TotalBsmtSF        0
Id                 0
Length: 81, dtype: int64
>>>
```

```
>>> missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
>>> missing_data.head(20)
              Total   Percent
PoolQC         1453  0.995205
MiscFeature    1406  0.963014
Alley          1369  0.937671
Fence          1179  0.807534
FireplaceQu     690  0.472603
LotFrontage     259  0.177397
GarageCond       81  0.055479
GarageType       81  0.055479
GarageYrBlt      81  0.055479
GarageFinish     81  0.055479
GarageQual       81  0.055479
BsmtExposure     38  0.026027
BsmtFinType2     38  0.026027
BsmtFinType1     37  0.025342
BsmtCond         37  0.025342
BsmtQual         37  0.025342
MasVnrArea        8  0.005479
MasVnrType        8  0.005479
Electrical        1  0.000685
Utilities         0  0.000000
>>>
```

```
>>> percent = (df_train.isnull().sum()/df_train.isnull().count()).sort_values(ascending=False)
>>> percent
PoolQC          0.995205
MiscFeature     0.963014
Alley           0.937671
Fence           0.807534
FireplaceQu     0.472603
                  ...
CentralAir      0.000000
SaleCondition   0.000000
Heating         0.000000
TotalBsmtSF     0.000000
Id              0.000000
Length: 81, dtype: float64
>>>
```

Missing data can also be imputed: The method is that set the values of a highly correlated column to replace the missing values.

There are 259 values are missing before the imputation, and after processing the missing value number is 0.

```
>>>
>>> print(df_train["LotFrontage"].isnull().sum())
259
>>> cond = df_train['LotFrontage'].isnull()
>>> df_train["LotFrontage"][cond]=df_train["SqrtLotArea"][cond]
__main__:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
>>> print(df_train["LotFrontage"].isnull().sum())
0
>>>
```

Another way is to mark the missing value of the column as “Missing”:

```
>>> mis=df_train['GarageType'].isnull()
>>>
>>> df_train["GarageType"][mis]="Missing"
>>> df_train["GarageType"].unique()
array(['Attchd', 'Detchd', 'BuiltIn', 'CarPort', 'Missing', 'Basment',
       '2Types'], dtype=object)
>>>
```

Drop the columns whose data value missing percentage is greater than 15%, drop the row whose “Electrical” is null. After dropping, the data frame is changed into 1459 rows * 76 columns from original 1460 rows * 81 columns.

```
>>> df_train = df_train.drop((missing_data[missing_data['Percent'] > 0.15]).index,1)
>>> df_train = df_train.drop(df_train.loc[df_train['Electrical'].isnull()].index)
>>> df_train
        Id  MSSubClass MSZoning  LotArea Street LotShape LandContour Utilities LotConfig  ... ScreenPorch PoolArea MiscVal MoSold YrSold SaleType  SaleCondition  SalePrice  SqrtLotArea
0        1          60       RL     8450   Pave      Reg         Lvl    AllPub    Inside  ...           0        0       0      2   2008       WD         Normal     208500    91.923882
1        2          20       RL     9600   Pave      Reg         Lvl    AllPub       FR2  ...           0        0       0      5   2007       WD         Normal     181500    97.979590
2        3          60       RL    11250   Pave      IR1         Lvl    AllPub    Inside  ...           0        0       0      9   2008       WD         Normal     223500   106.066017
3        4          70       RL     9550   Pave      IR1         Lvl    AllPub    Corner  ...           0        0       0      2   2006       WD        Abnorml     140000    97.724101
4        5          60       RL    14260   Pave      IR1         Lvl    AllPub       FR2  ...           0        0       0     12   2008       WD         Normal     250000   119.415242
...    ...         ...      ...      ...    ...      ...         ...       ...       ...  ...         ...      ...     ...    ...    ...      ...            ...        ...          ...
1455  1456          60       RL     7917   Pave      Reg         Lvl    AllPub    Inside  ...           0        0       0      8   2007       WD         Normal     175000    88.977525
1456  1457          20       RL    13175   Pave      Reg         Lvl    AllPub    Inside  ...           0        0       0      2   2010       WD         Normal     210000   114.782403
1457  1458          70       RL     9042   Pave      Reg         Lvl    AllPub    Inside  ...           0        0    2500      5   2010       WD         Normal     266500    95.089432
1458  1459          20       RL     9717   Pave      Reg         Lvl    AllPub    Inside  ...           0        0       0      4   2010       WD         Normal     142125    98.574845
1459  1460          20       RL     9937   Pave      Reg         Lvl    AllPub    Inside  ...           0        0       0      6   2008       WD         Normal     147500    99.684502

[1459 rows x 76 columns]
```

Check correlation between two variables through corr() function: LotFrontage and LotArea

```
>>> df_train['LotFrontage'].corr(df_train['LotArea'])
0.42609501877180833
>>>
```

Another sqrt() function can be further applied to reduce the numeric effect.

```
>>> df_train['SqrtLotArea']=np.sqrt(df_train['LotArea'])
>>> df_train['LotFrontage'].corr(df_train['SqrtLotArea'])
0.6020022167939361
>>>
```

For outliers processing:

There are some outliers for several columns, these outliers can be removed directly to reduce the effects of outliers in final prediction processing. The train data is changed into 1444 * 76 from 1459 * 76.