Housingcode

February 5, 2021

0.1 # Housing Market

[1]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
from scipy import stats
from geopy.geocoders import Nominatim
import folium

train_data = pd.read_csv('C:
 ↪\\Users\\52551\\Documents\\notas-vs\\final\\datos\\train.csv')
test_data = pd.read_csv('C:
 ↪\\Users\\52551\\Documents\\notas-vs\\final\\datos\\test.csv')

print(train_data.head())
print('Train data shape: '+str(np.shape(train_data)))
print('-----------------------------------------------------------------')
print(test_data.head())
print('Test data shape: '+str(np.shape(test_data)))
```

```
   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 data shape: (1460, 81)
----------------------------------------------------------------
     Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
0  1461          20       RH         80.0    11622   Pave   NaN      Reg
1  1462          20       RL         81.0    14267   Pave   NaN      IR1
2  1463          60       RL         74.0    13830   Pave   NaN      IR1
3  1464          60       RL         78.0     9978   Pave   NaN      IR1
4  1465         120       RL         43.0     5005   Pave   NaN      IR1

  LandContour Utilities  ... ScreenPorch PoolArea PoolQC  Fence MiscFeature  \
0         Lvl    AllPub  ...         120        0    NaN  MnPrv         NaN
1         Lvl    AllPub  ...           0        0    NaN    NaN        Gar2
2         Lvl    AllPub  ...           0        0    NaN  MnPrv         NaN
3         Lvl    AllPub  ...           0        0    NaN    NaN         NaN
4         HLS    AllPub  ...         144        0    NaN    NaN         NaN

  MiscVal MoSold  YrSold  SaleType  SaleCondition
0       0      6    2010        WD         Normal
1   12500      6    2010        WD         Normal
2       0      3    2010        WD         Normal
3       0      6    2010        WD         Normal
4       0      1    2010        WD         Normal

[5 rows x 80 columns]
Test data shape: (1459, 80)
```

0.2 Exploratory analysis

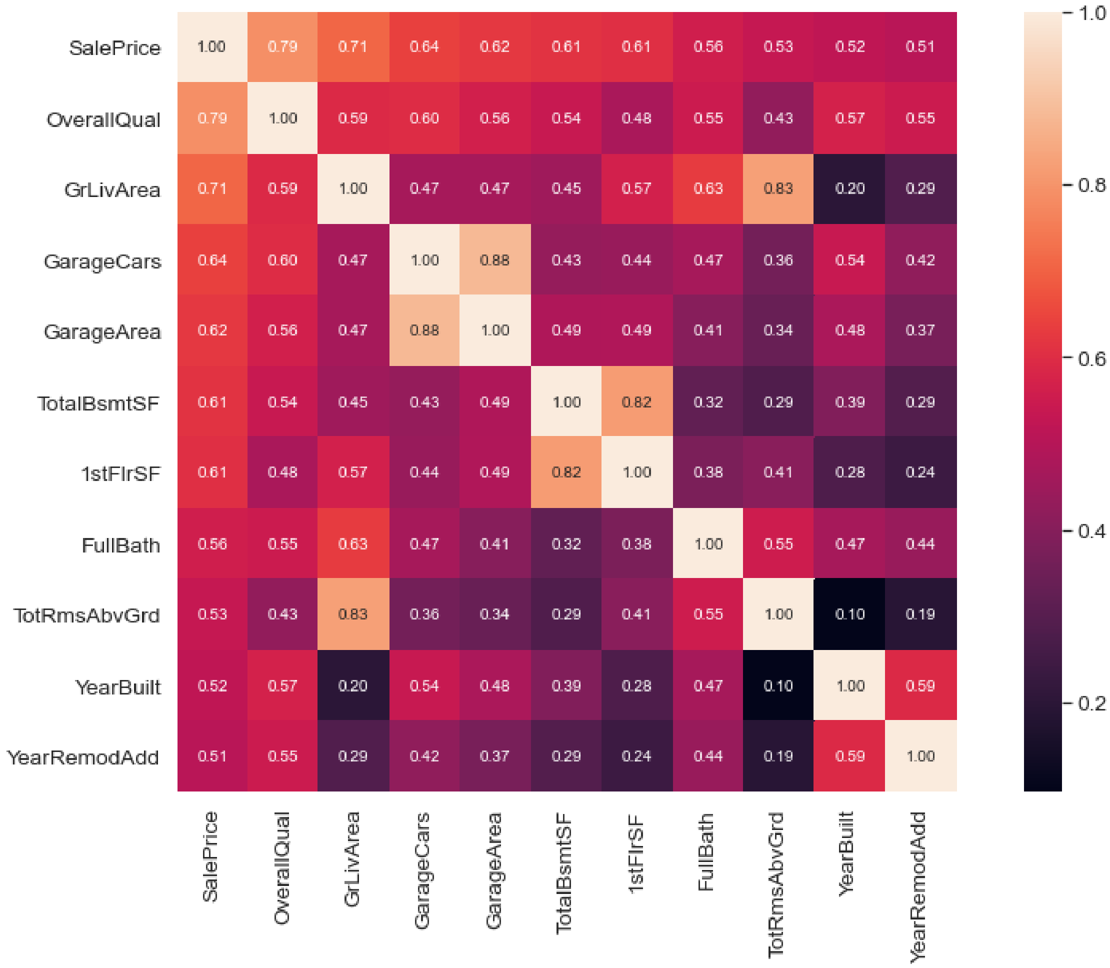
[2]:
```
# Descriptive analitics
train_data.describe()
```

[2]:
```
                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
50%     730.500000   50.000000   69.000000    9478.500000    6.000000
75%    1095.250000   70.000000   80.000000   11601.500000    7.000000
max    1460.000000  190.000000  313.000000  215245.000000   10.000000

       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  …
50%       5.000000  1973.000000   1994.000000     0.000000   383.500000  …
75%       6.000000  2000.000000   2004.000000   166.000000   712.250000  …
max       9.000000  2010.000000   2010.000000  1600.000000  5644.000000  …

        WoodDeckSF  OpenPorchSF  EnclosedPorch    3SsnPorch  ScreenPorch  \
count  1460.000000  1460.000000    1460.000000  1460.000000  1460.000000
mean     94.244521    46.660274      21.954110     3.409589    15.060959
std     125.338794    66.256028      61.119149    29.317331    55.757415
min       0.000000     0.000000       0.000000     0.000000     0.000000
25%       0.000000     0.000000       0.000000     0.000000     0.000000
50%       0.000000    25.000000       0.000000     0.000000     0.000000
75%     168.000000    68.000000       0.000000     0.000000     0.000000
max     857.000000   547.000000     552.000000   508.000000   480.000000

          PoolArea       MiscVal       MoSold       YrSold      SalePrice
count  1460.000000   1460.000000  1460.000000  1460.000000    1460.000000
mean      2.758904     43.489041     6.321918  2007.815753  180921.195890
std      40.177307    496.123024     2.703626     1.328095   79442.502883
min       0.000000      0.000000     1.000000  2006.000000   34900.000000
25%       0.000000      0.000000     5.000000  2007.000000  129975.000000
50%       0.000000      0.000000     6.000000  2008.000000  163000.000000
75%       0.000000      0.000000     8.000000  2009.000000  214000.000000
max     738.000000  15500.000000    12.000000  2010.000000  755000.000000

[8 rows x 38 columns]
```

[3]:
```python
#saleprice correlation matrix
k = 11 #number of variables for heatmap
corrmat = train_data.corr()
cols = corrmat.nlargest(k, 'SalePrice')['SalePrice'].index
cm = np.corrcoef(train_data[cols].values.T)
sns.set(font_scale=1.25)
f, ax = plt.subplots(figsize=(15, 9))
```

```
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f',␣
↪annot_kws={'size': 10}, yticklabels=cols.values, xticklabels=cols.values)
plt.show()
```

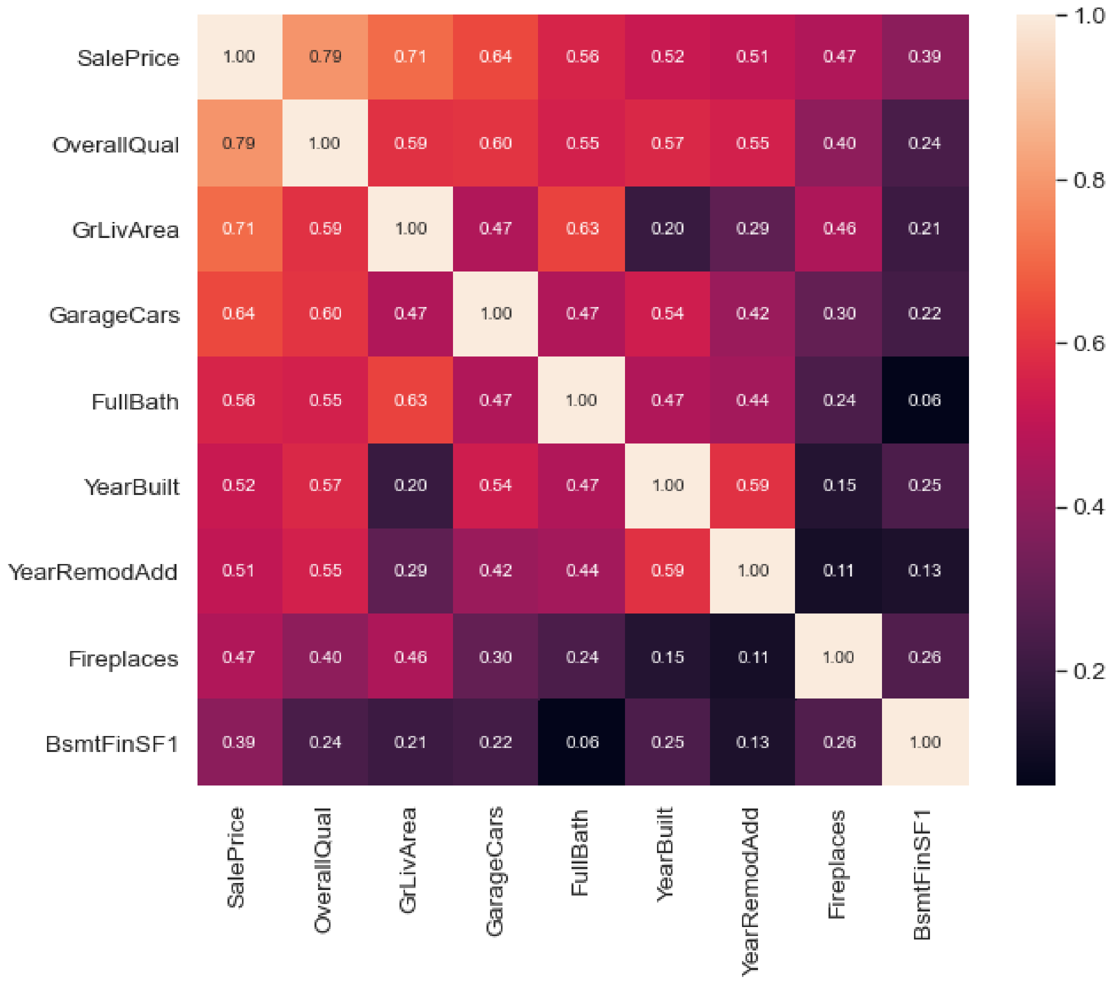


[4]:
```
#Droping the correlated independent variables
train_data.
↪drop(['GarageArea','TotalBsmtSF','TotRmsAbvGrd','1stFlrSF','GarageYrBlt','MasVnrArea'],axis
print('Train data shape: '+str(np.shape(train_data)))
```

```
Train data shape: (1460, 75)
```

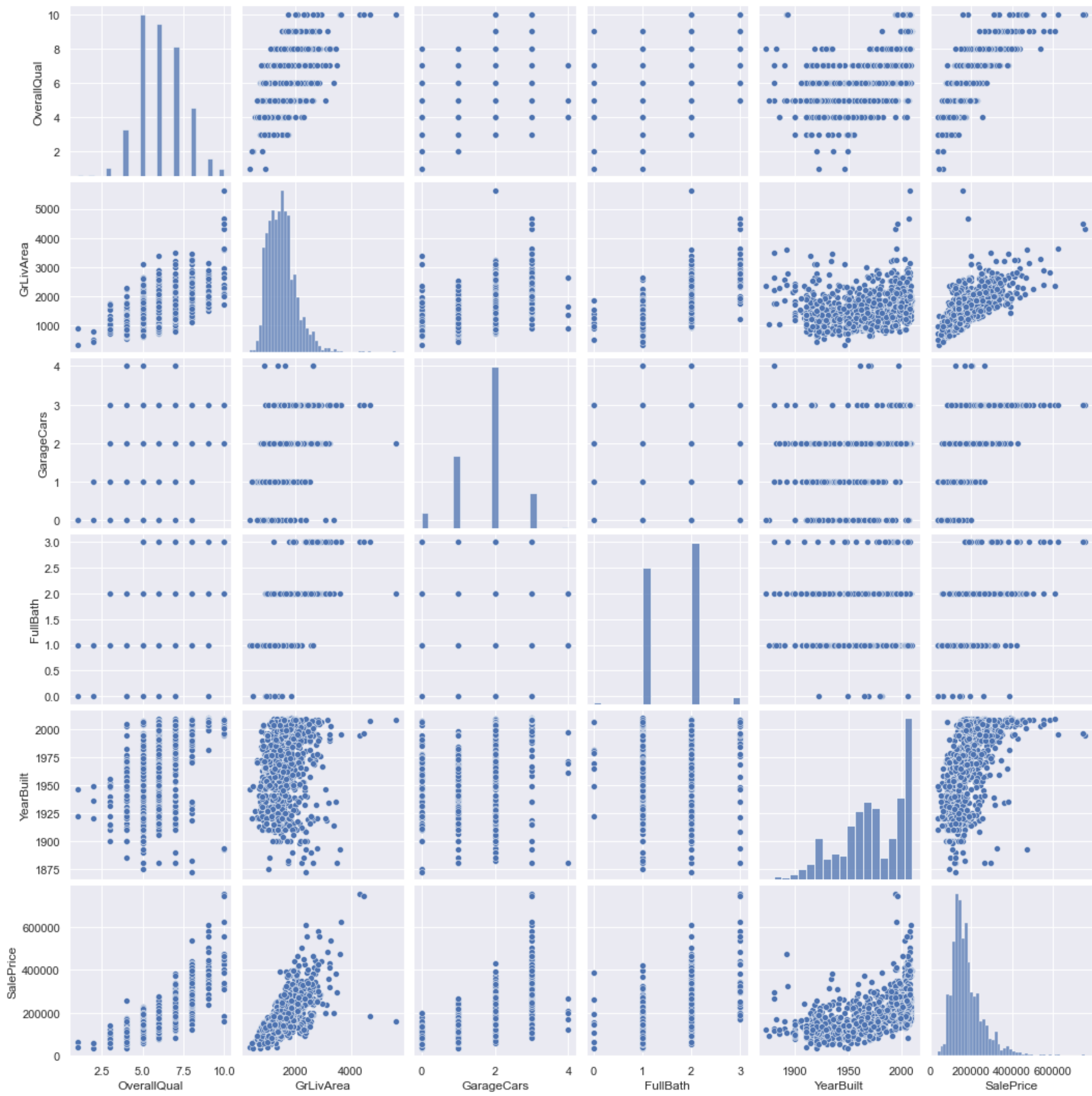
[5]:
```
#saleprice correlation matrix
k = 9 #number of variables for heatmap
corrmat = train_data.corr()
cols = corrmat.nlargest(k, 'SalePrice')['SalePrice'].index
cm = np.corrcoef(train_data[cols].values.T)
sns.set(font_scale=1.25)
```

```
f, ax = plt.subplots(figsize=(10, 8))
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f',␣
↪annot_kws={'size': 10}, yticklabels=cols.values, xticklabels=cols.values)
plt.show()
```



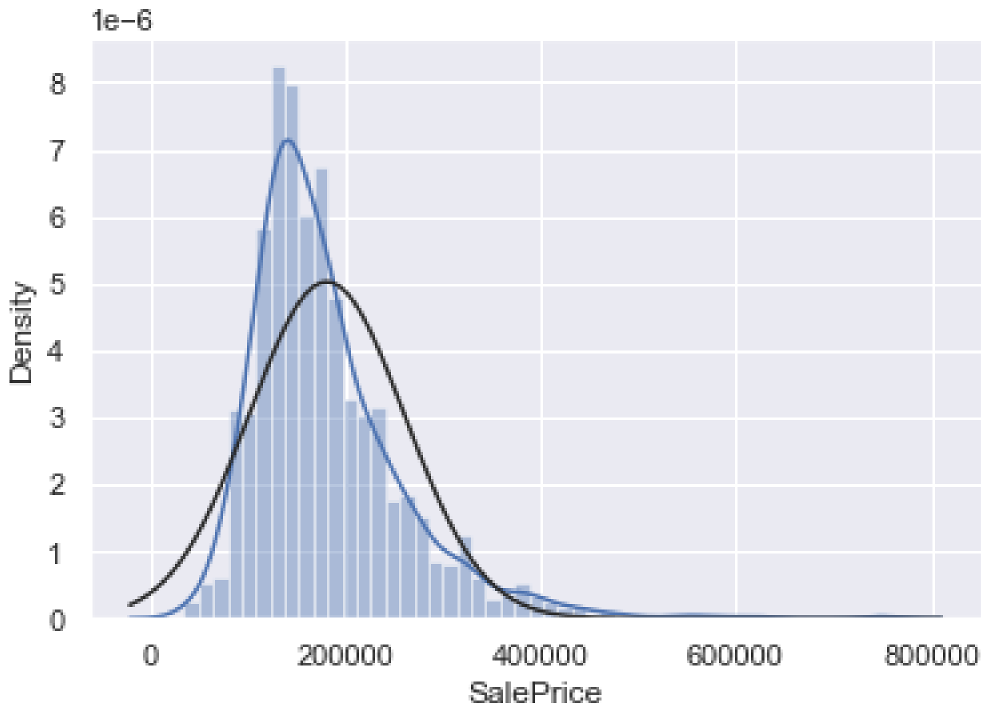
[6]:

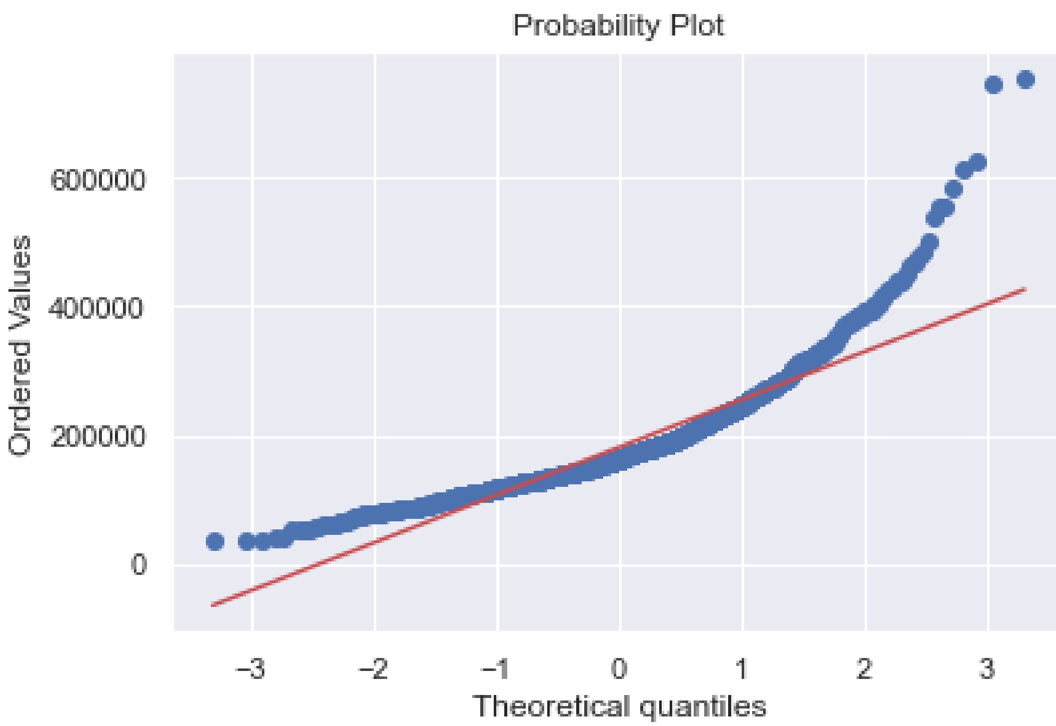
```
#scatterplot
sns.set()
cols =␣
↪['OverallQual','GrLivArea','GarageCars','FullBath','YearBuilt','SalePrice']
sns.pairplot(train_data[cols], size = 2.5)
plt.show()
```



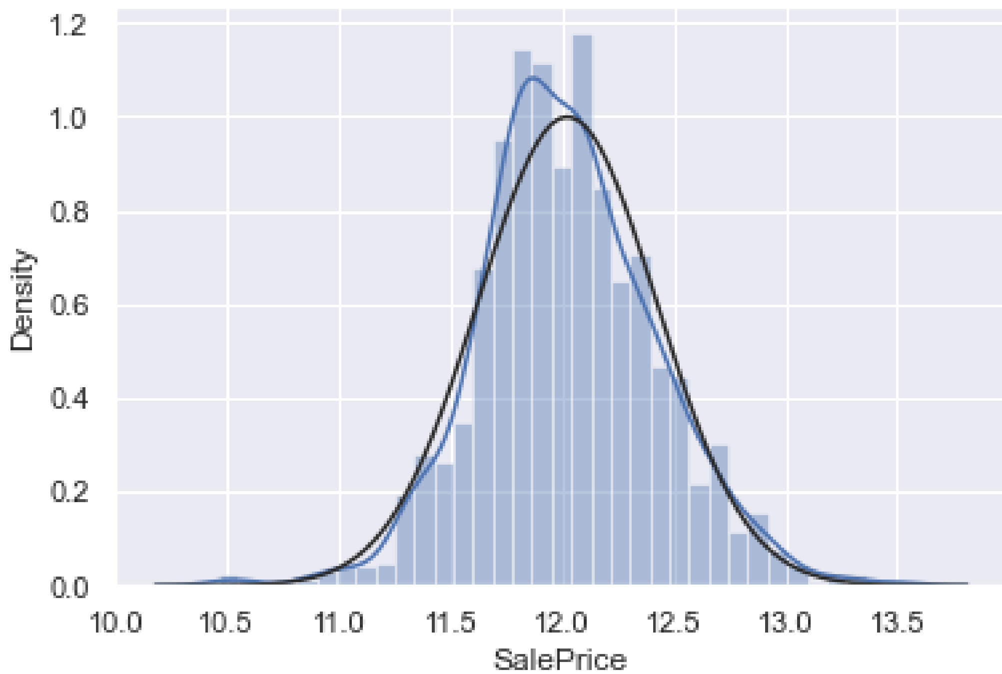
0.3 Normalizing Data

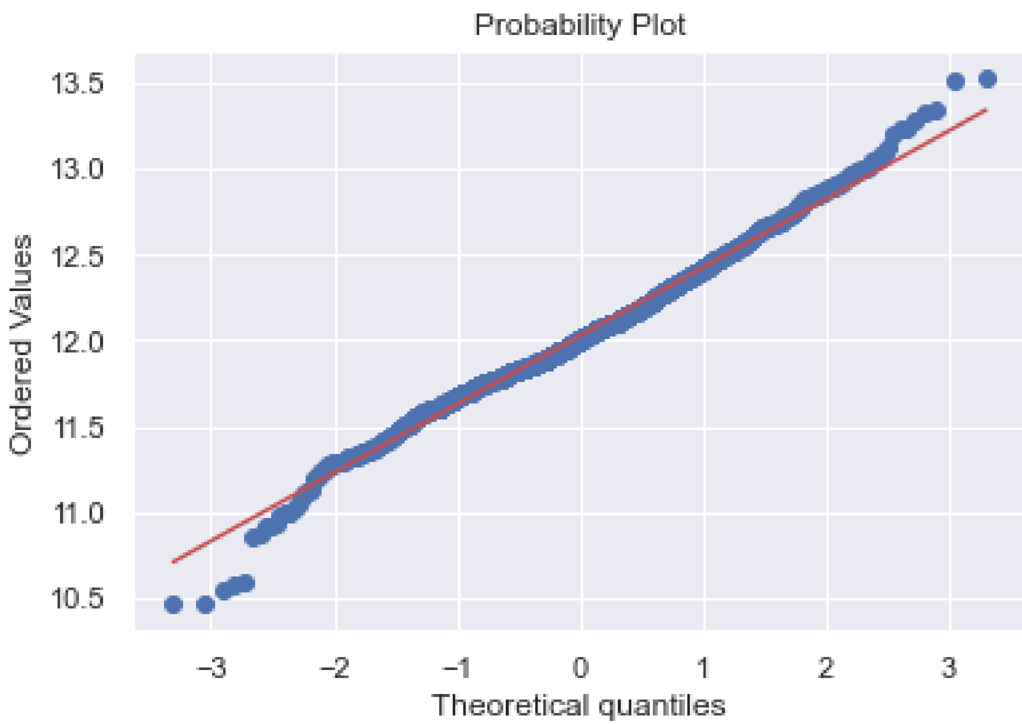
[7]:
```
#SalePrice Histogram and normal probability plot
sns.distplot(train_data['SalePrice'], fit=norm)
fig = plt.figure()
res = stats.probplot(train_data['SalePrice'], plot=plt)
```


1e−6
8
7
6
5
4
3
2
1
0
Density
0
200000
400000
600000
800000
SalePrice

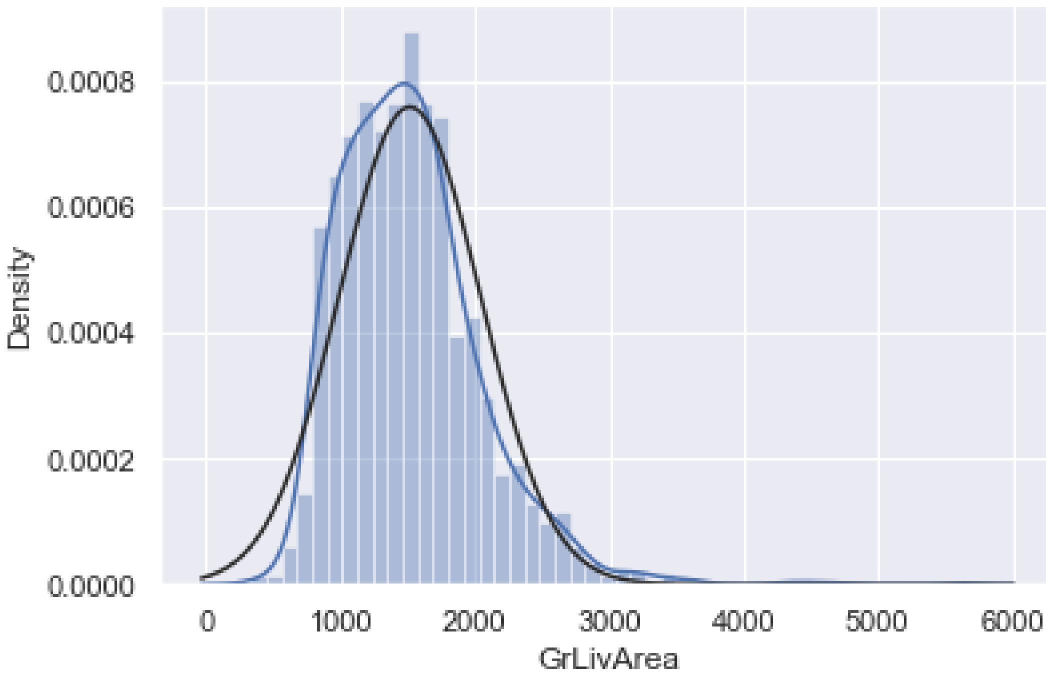

Probability Plot
600000
400000
200000
0
Ordered Values
−3
−2
−1
0
1
2
3
Theoretical quantiles

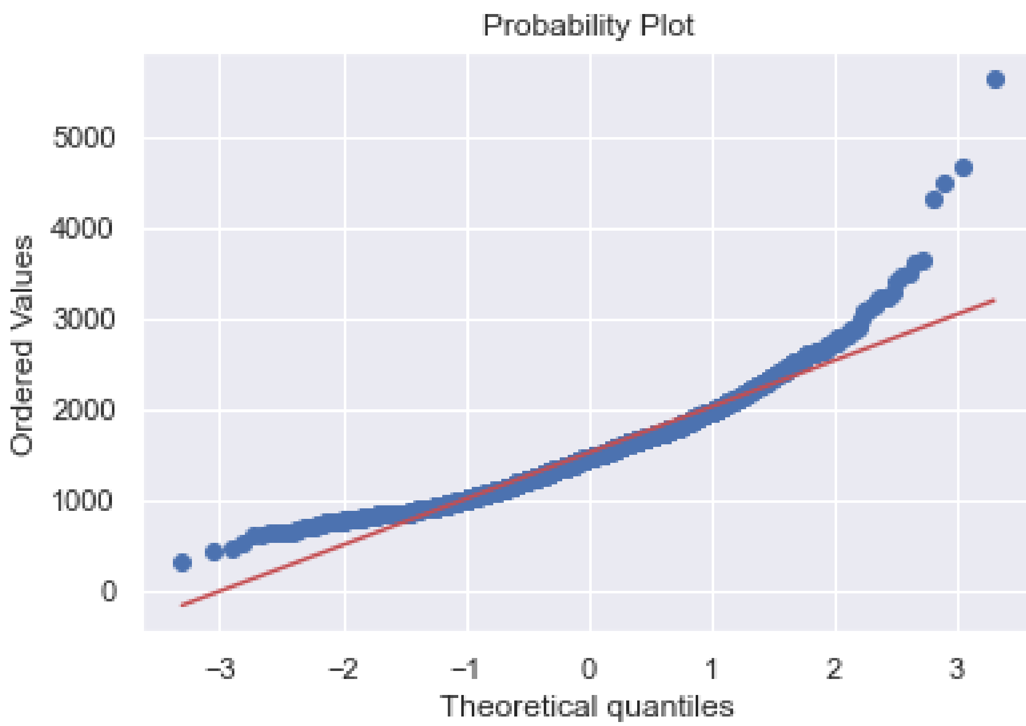
[8]:
```
# Adjusting 'SalePrice' for a Normal distribution
train_data['SalePrice'] = np.log(train_data['SalePrice'])

sns.distplot(train_data['SalePrice'], fit=norm)
fig = plt.figure()
res = stats.probplot(train_data['SalePrice'], plot=plt)
```





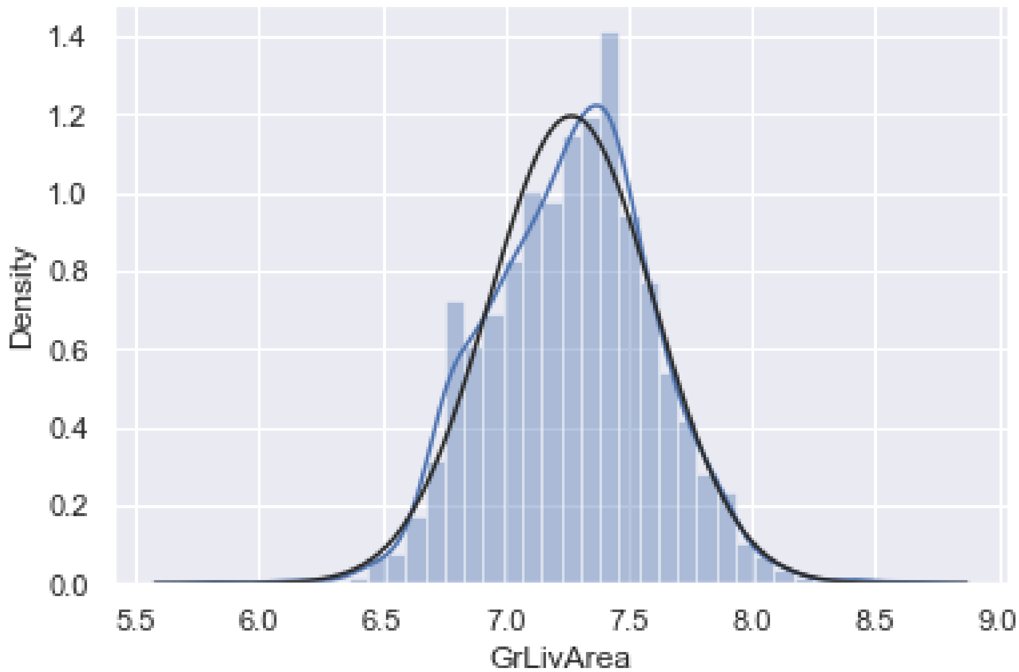
[9]:
```
#GrLivArea Histogram and normal probability plot
sns.distplot(train_data['GrLivArea'], fit=norm)
fig = plt.figure()
res = stats.probplot(train_data['GrLivArea'], plot=plt)
```

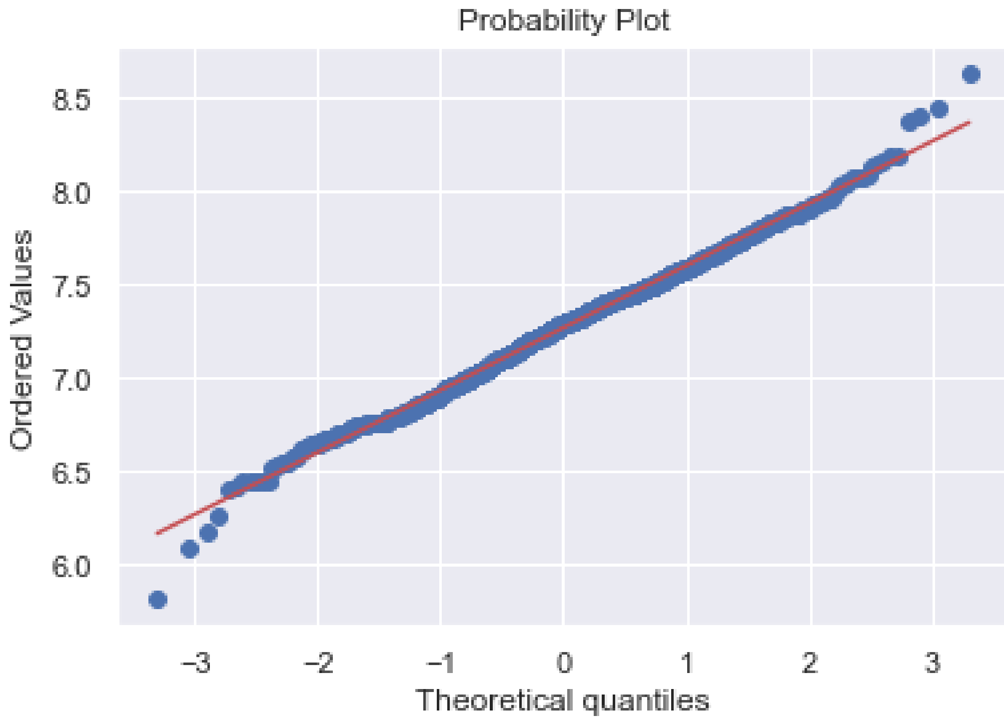

0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
1000
2000
3000
4000
5000
6000
GrLivArea


Probability Plot
5000
4000
3000
2000
1000
0
Ordered Values
−3
−2
−1
0
1
2
3
Theoretical quantiles

[10]:

```
# Adjusting 'GrLivArea' for a Normal distribution
train_data['GrLivArea'] = np.log(train_data['GrLivArea'])

sns.distplot(train_data['GrLivArea'], fit=norm)
fig = plt.figure()
res = stats.probplot(train_data['GrLivArea'], plot=plt)
```





[11]:
```
#Split Data for training and testing
x_train, x_test, y_train, y_test =␣
↪train_test_split(train_data[['OverallQual','GrLivArea','GarageCars','FullBath','YearBuilt']
↪3,random_state=0)
```

[12]:
```
# Kmeans clustering training
x_train_data_clust=train_data.
↪drop(labels=['SalePrice','MSZoning','Street','Alley','LotShape','LandContour','Utilities','
y_train_data_clust = train_data['SalePrice'].values

x_train_data_clust = x_train_data_clust.values[:,1:]
x_train_data_clust = np.nan_to_num(x_train_data_clust)
y_train_data_clust= np.nan_to_num(y_train_data_clust)

x_train_data_clust = StandardScaler().fit_transform(x_train_data_clust)
x_train_data_clust
```

[12]:
```
array([[ 0.07337496,  0.2128772 , -0.20714171, …, -0.08768781,
        -1.5991111 ,  0.13877749],
       [-0.87256276,  0.64574726, -0.09188637, …, -0.08768781,
        -0.48911005, -0.61443862],
       [ 0.07337496,  0.29945121,  0.07347998, …, -0.08768781,
         0.99089135,  0.13877749],
```

```
…,
[ 0.30985939,  0.2417352 , -0.14781027, …,  4.95311151,
 -0.48911005,  1.64520971],
[-0.87256276,  0.29945121, -0.08016039, …, -0.08768781,
 -0.8591104 ,  1.64520971],
[-0.87256276,  0.50145724, -0.05811155, …, -0.08768781,
 -0.1191097 ,  0.13877749]])
```

[13]:
```
#Train Model and Predict
k_means = KMeans(init = "k-means++", n_clusters = 3, n_init = 100)
k_means.fit(x_train_data_clust)
labels = k_means.labels_

print(labels)
```

```
[1 0 1 … 1 0 0]
```

[14]:
```
#adding labels
train_data["Cluster"] = labels
x_train_cluster, x_test_cluster, y_train_cluster, y_test_cluster =␣
↪train_test_split(train_data[['OverallQual','GrLivArea','GarageCars','FullBath','YearBuilt',
↪3,random_state=0)
x_test_cluster
```

[14]:

	OverallQual	GrLivArea	GarageCars	FullBath	YearBuilt	Cluster
529	6	7.830028	2	3	1957	2
491	6	7.363914	1	1	1941	0
459	5	7.092574	1	1	1950	0
279	7	7.611842	2	2	1977	1
655	6	6.995766	1	1	1971	0
…	…	…	…	…	…	…
271	7	7.217443	2	1	1954	0
445	6	7.431892	2	1	1956	0
654	8	7.655864	3	2	1995	1
1280	7	7.360740	2	2	2002	1
898	9	7.768110	3	2	2009	1

[438 rows x 6 columns]

[15]:
```
# Adding the cluster analysis as a predictor variable
lmclust = LinearRegression()
lmclust.fit(x_train_cluster,y_train_cluster)
Yhatclust=lmclust.predict(x_test_cluster)

# Metrics
print('Cluster R^2 = '+str(lmclust.score(x_test_cluster,y_test_cluster)))
print('Cluster RMSE = '+str(mean_squared_error(y_test_cluster␣
↪,Yhatclust,squared=False)))
```

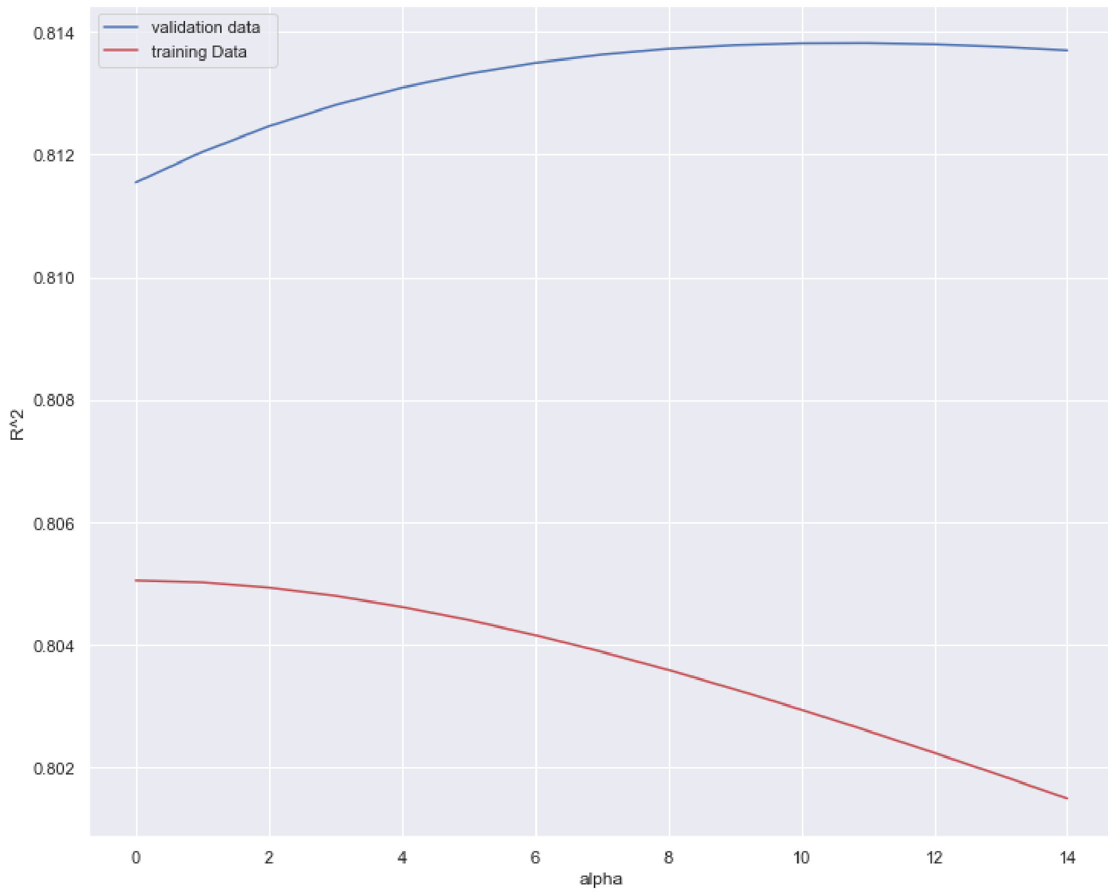
```
print('Range = '+str(y_test_cluster.max()-y_test_cluster.min()))
```

```
Cluster R^2 = 0.8137879307033405
Cluster RMSE = 0.16962263456404542
Range = 2.652571048781981
```

0.4 Ridge Modeling

[16]:
```
#Plotting R^2 for diferente alphas
Rsqu_test = []
Rsqu_train = []
dummy1 = []
Alpha = np.array(range(0,15))
for alpha in Alpha:
    RigeModel = Ridge(alpha=alpha)
    RigeModel.fit(x_train, y_train)
    Rsqu_test.append(RigeModel.score(x_test, y_test))
    Rsqu_train.append(RigeModel.score(x_train, y_train))

width = 12
height = 10
plt.figure(figsize=(width, height))

plt.plot(Alpha,Rsqu_test, label='validation data  ')
plt.plot(Alpha,Rsqu_train, 'r', label='training Data ')
plt.xlabel('alpha')
plt.ylabel('R^2')
plt.legend()
```

[16]:
```
<matplotlib.legend.Legend at 0x26cd0923dc0>
```



[17]:
```
#Ridge model prediction
parameters1= [{'alpha': [0.001,.01,0.1,1, 10, 100, 1000]}]
Grid1 = GridSearchCV(RigeModel, parameters1,cv=4)
Grid1.fit(x_train,y_train)
BestRR=Grid1.best_estimator_

Ridgelm = Ridge(alpha=1)
Ridgelm.fit(x_train, y_train)
YhatRid = Ridgelm.predict(x_test)

print('BestRR = '+str(BestRR))

print('Ridge R^2 = '+str(Ridgelm.score(x_test,y_test)))
print('Ridge RMSE = '+str(mean_squared_error(y_test ,YhatRid,squared=False)))
print('Range = '+str(y_test.max()-y_test.min()))
```

```
BestRR = Ridge(alpha=1)
Ridge R^2 = 0.8120441945914397
Ridge RMSE = 0.17041497822092613
```

Range = 2.652571048781981

0.5 Multiple Linear Regression

[18]:
```
#Linear Regression
lm = LinearRegression()
lm.fit(x_train,y_train)
Yhat=lm.predict(x_test)

# Metrics
print('MLR R^2 = '+str(lm.score(x_test,y_test)))
print('MLR RMSE = '+str(mean_squared_error(y_test ,Yhat,squared=False)))
print('Range = '+str(y_test.max()-y_test.min()))
```

```
MLR R^2 = 0.8115446401767943
MLR RMSE = 0.1706412948895512
Range = 2.652571048781981
```

0.6 Test Data Wrangling and Prediction

[19]:
```
# Test Data Cleaning
test_data['GrLivArea'] = np.log(test_data['GrLivArea'])
x_test_sub =␣
 ↪test_data[['OverallQual','GrLivArea','GarageCars','FullBath','YearBuilt']]
print(x_test_sub.isnull().sum())

x_test_sub['GarageCars'].fillna(x_test_sub['GarageCars'].mean(),inplace=True)
print('----Clean Data------')
print(x_test_sub.isnull().sum())
```

```
OverallQual    0
GrLivArea      0
GarageCars     1
FullBath       0
YearBuilt      0
dtype: int64
----Clean Data------
OverallQual    0
GrLivArea      0
GarageCars     0
FullBath       0
YearBuilt      0
dtype: int64
```

[20]:
```
# Submission

prediction = lm.predict(x_test_sub)
Housing_results = pd.DataFrame(prediction)
```

```
Housing_results.columns=['SalePrice']
Housing_results['SalePrice'] = np.exp(Housing_results['SalePrice'])
print(Housing_results.head())
Housing_results.to_csv(path_or_buf='C:
↪\\Users\\52551\\Documents\\notas-vs\\final\\datos\\Housing_results.csv')
```

```
       SalePrice
0  108047.697615
1  145453.067162
2  170401.173192
3  188858.573263
4  205756.463761
```