HomePriceRegression

September 29, 2020

1 Home Price Regression

This is a kaggle dataset where the goal is to predict home prices. There are 79 explanatory variables which represent most features of residential homes in Ames, Iowa.

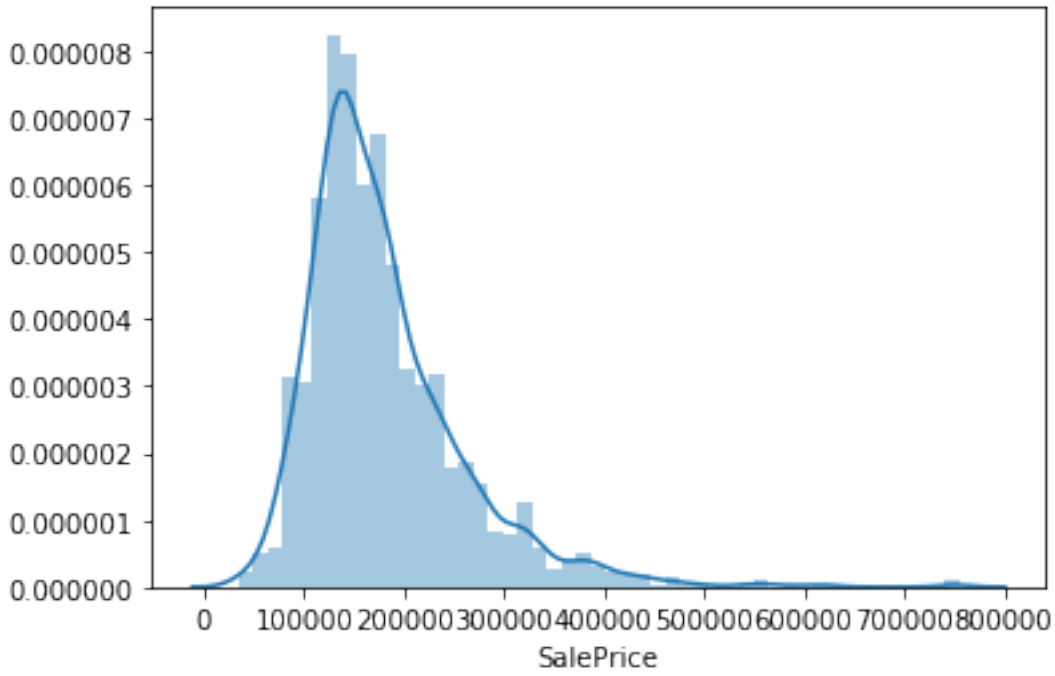
[16]:
```
from pycaret.regression import *
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
from scipy import stats
```

[17]:
```
#Read in Data
data = pd.read_csv('train.csv')
data.head()
#Target Distribution
sns.distplot(data['SalePrice']);

#Descriptive Statistics
data['SalePrice'].describe()

#missing data
total = data.isnull().sum().sort_values(ascending=False)
percent = (data.isnull().sum()/data.isnull().count()).
 ↪sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
```

[17]:

	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
```

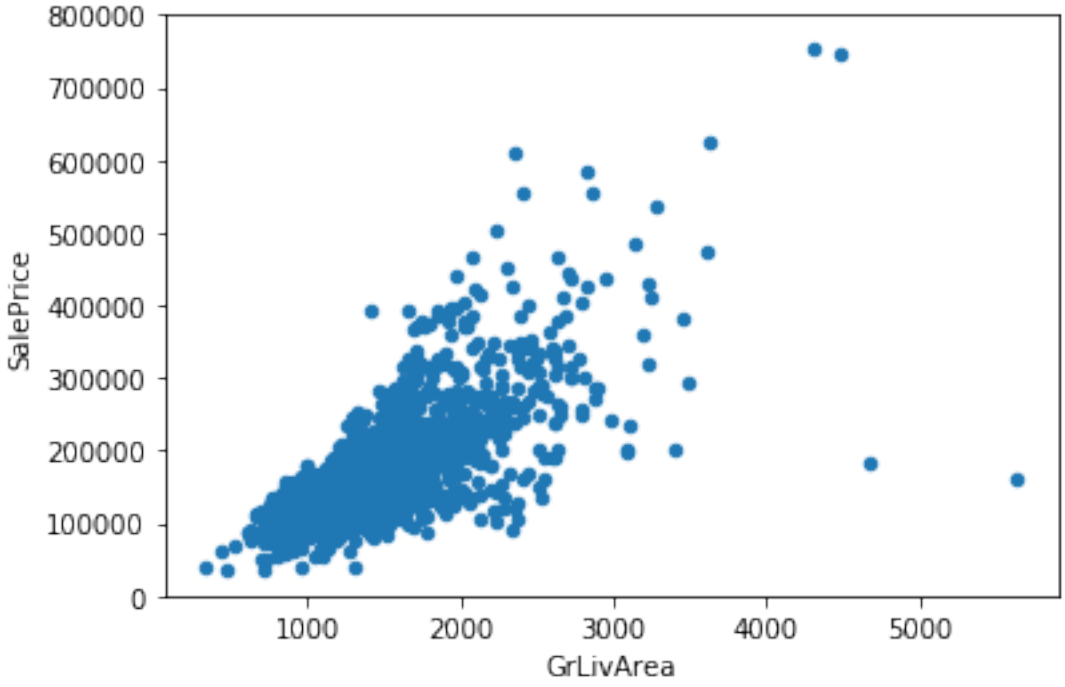


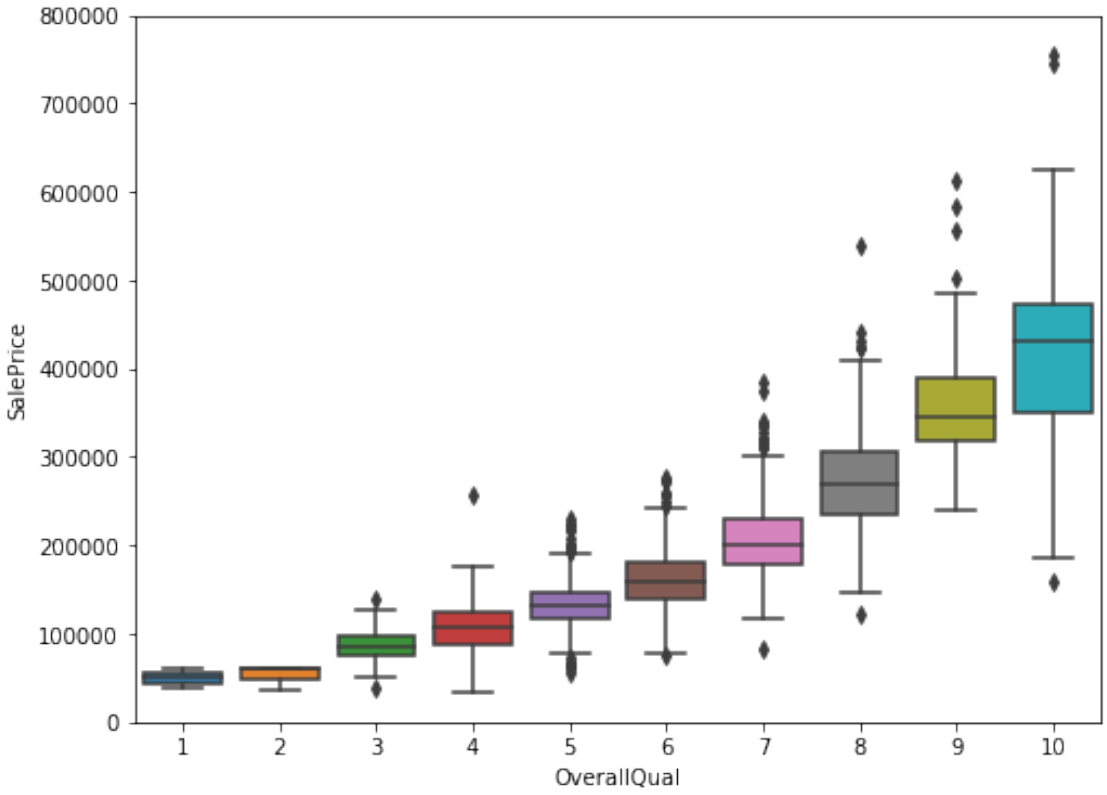
[18]:
```
#Delete missing data
data = data.drop((missing_data[missing_data['Total'] > 1]).index,1)
data = data.drop(data.loc[data['Electrical'].isnull()].index)
data.isnull().sum().max() #just checking that there's no missing data missing...
```

[18]: 0

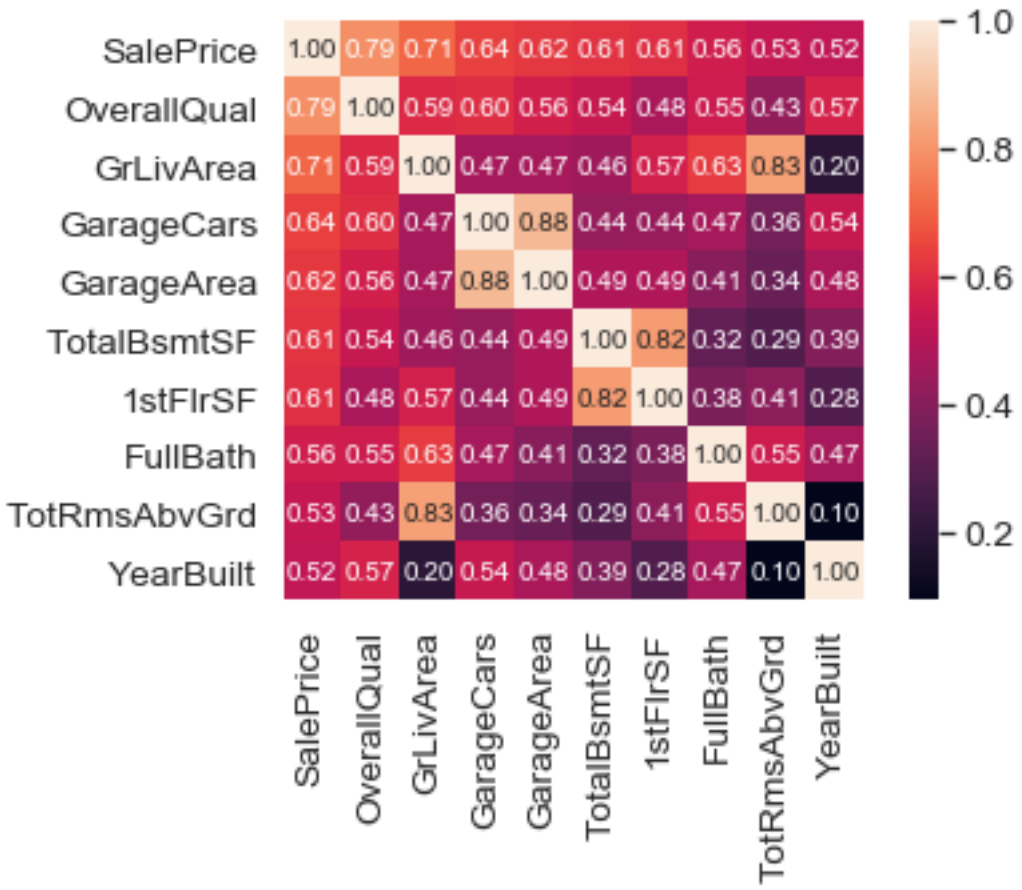
[19]:
```
##Scatter Continuous Relationships
var = 'GrLivArea'
data_con = pd.concat([data['SalePrice'], data[var]], axis=1)
data_con.plot.scatter(x=var, y='SalePrice', ylim=(0,800000));
```

```
##Scatter Categorical Relationships
var = 'OverallQual'
data_cat = pd.concat([data['SalePrice'], data[var]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var, y="SalePrice", data=data_cat)
fig.axis(ymin=0, ymax=800000);
```

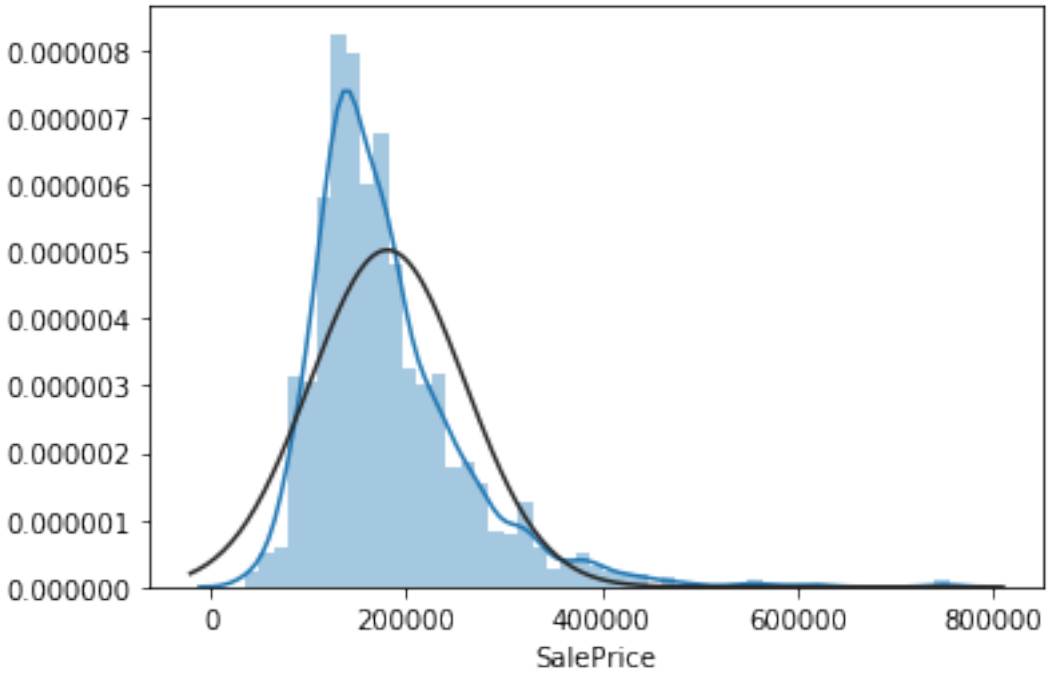


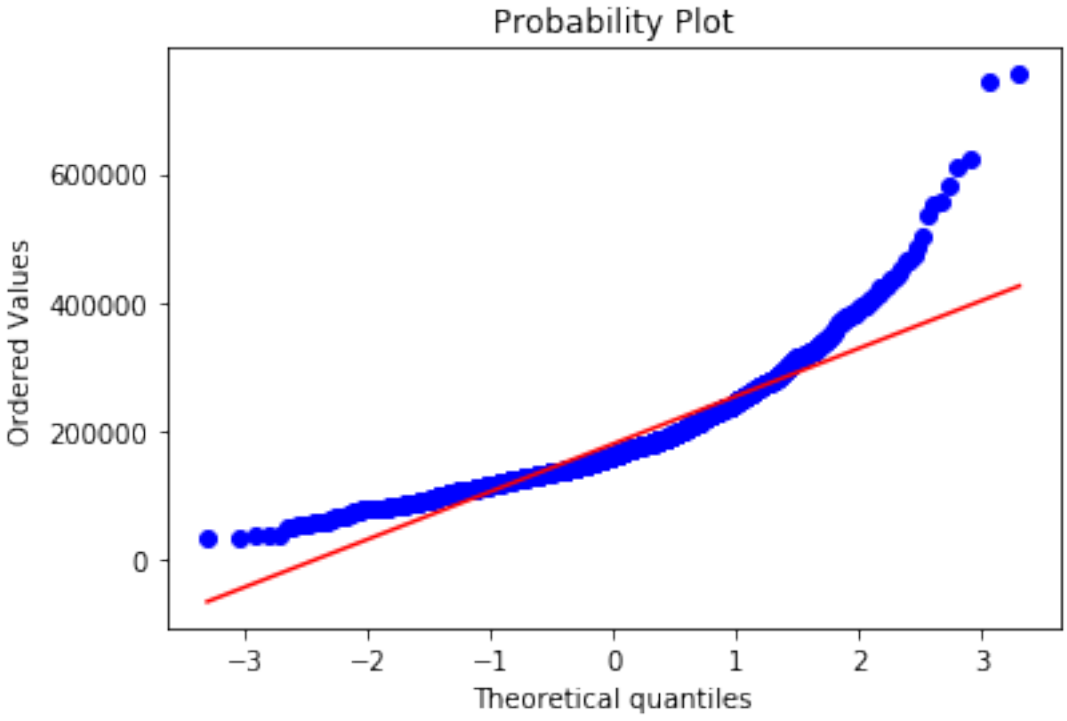


[22]:
```
#saleprice correlation matrix

k = 10 #number of variables for heatmap
corrmat = data.corr()
cols = corrmat.nlargest(k, 'SalePrice')['SalePrice'].index
cm = np.corrcoef(data[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f',␣
 ↪annot_kws={'size': 10}, yticklabels=cols.values, xticklabels=cols.values)
plt.show()
```

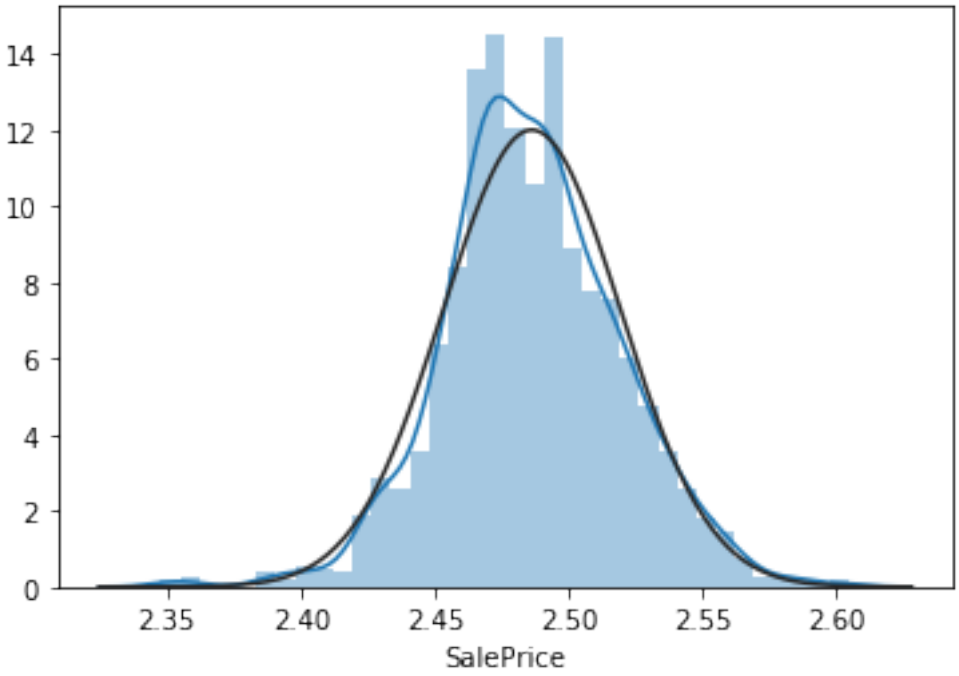


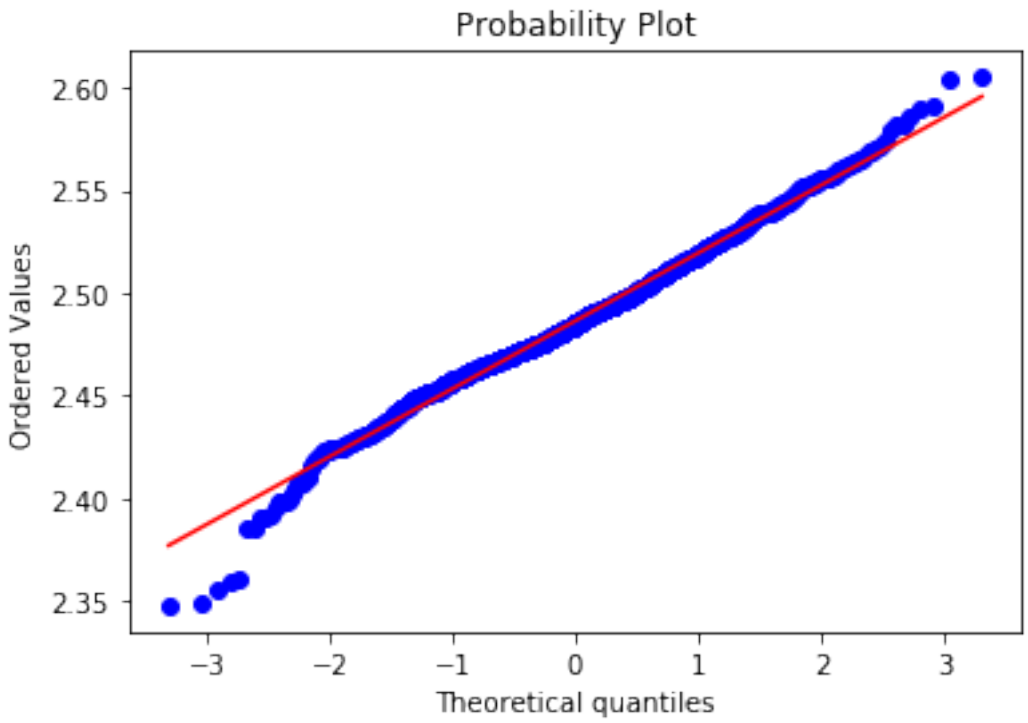
[13]:
```
#histogram and normal probability plot
sns.distplot(data['SalePrice'], fit=norm);
fig = plt.figure()
res = stats.probplot(data['SalePrice'], plot=plt)
```


0.000008
0.000007
0.000006
0.000005
0.000004
0.000003
0.000002
0.000001
0.000000
0
200000
400000
600000
800000
SalePrice


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

[15]:
```
#applying log transformation to normalize
data['SalePrice'] = np.log(data['SalePrice'])


#transformed histogram and normal probability plot
sns.distplot(data['SalePrice'], fit=norm);
fig = plt.figure()
res = stats.probplot(data['SalePrice'], plot=plt)
```





[27]:
```
#Regression Setup

ctl = setup(data=data, target = 'SalePrice',use_gpu = True, normalize=True )
```

```
Setup Succesfully Completed.
```

```
<pandas.io.formats.style.Styler at 0x19901d58548>
```

[28]:
```
compare_models(fold=10)
```

```
<pandas.io.formats.style.Styler at 0x19901cc2908>
```

[28]:
```
<catboost.core.CatBoostRegressor at 0x19900c85488>
```

[29]:
```
#Create CatBoost model
CBR = create_model('catboost')
tuned_CBR = tune_model(CBR)
```

```
<pandas.io.formats.style.Styler at 0x19901c7b9c8>
```

[50]:
```
#Load Test data, predict, and create proper dataframe for kaggle submisison

test = pd.read_csv('test.csv')
pred = predict_model(CBR, data= test)
pd.DataFrame(pred)
pred = pred.rename(columns={'Label':'SalePrice'})
pred = pred[['Id','SalePrice']]
pred.head()
```

[50]:

index	Id	SalePrice
0	1461	130835.3618
1	1462	176170.9827
2	1463	193195.8016
3	1464	196266.4088
4	1465	185225.1019

[51]:
```
#Export file to CVS
pred.to_csv('Kaggle_Final.csv', index=False, header=True)
```

Overall, the model created ended up giving me a 14% Root Mean Log Squared Error. Not state of the art, but decent results for only 2 submissions. If I were to submit another round, I would try to blend the top 3-5 models and see if that would provide better results.