CS521 SC2 Information Structures with Python

(2020 Summer 1)

Final Project

Haili Huang

BUID：61556665

Problem descriptions

Housing is the most basic need of every family. Finding assets with a reasonable price always combine with mass work. We are trying to build suitable model with Boston bousing database to quickly and precisely predict prices for buyers and sellers. This model can provide seller and buyers with appropriate reference for selling price information. The goal is to promote fair market transaction and help residents reduce the difficulties encountered in buying and selling property.

File included

This project contains the following files:

House Prediction Data.CSV

data_description.text

Analysis process

Data description:

```
#observation
data
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	Mis
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2914	2915	160	RM	21.0	1936	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
2915	2916	160	RM	21.0	1894	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
2916	2917	20	RL	160.0	20000	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
2917	2918	85	RL	62.0	10441	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	Shed	
2918	2919	60	RL	74.0	9627	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	

2919 rows × 81 columns

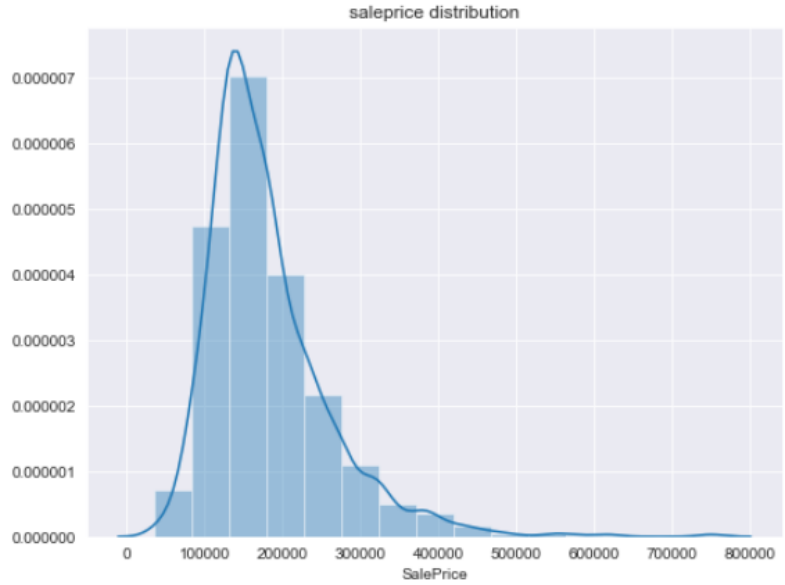
The total database is very detailed, nearly 3K samples have 81 variables available for research. I classified the data into several categories: exterior, internal, utility, overall, others. I think there are representative variables as independent variables in each category.

Our main research direction is price, so the samples without prices in the database are relatively useless for us; I use drop.na to delete the samples which sale price is NA. There are 1460 sample lest in our dataset.

```
data2=data.dropna(subset=['SalePrice'])#Drop NA in saleprice
data2.shape
```

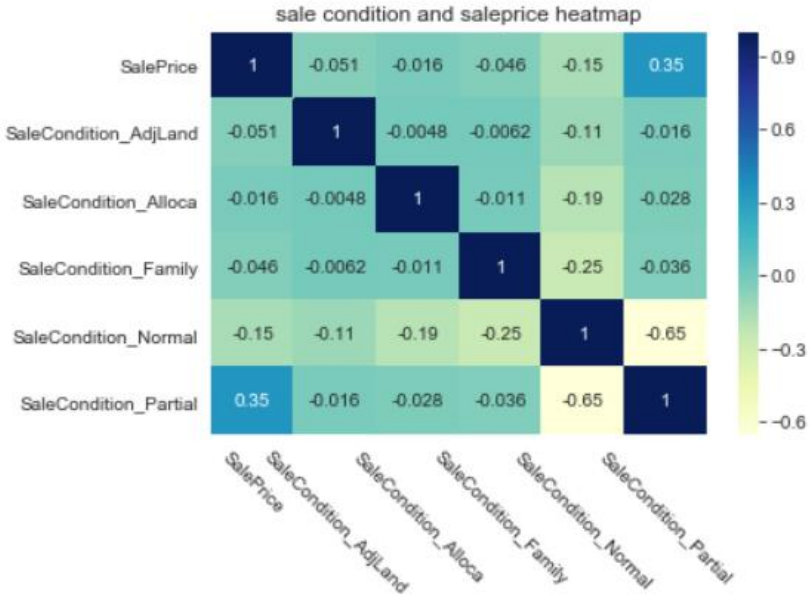
```
(1460, 81)
```

Visualization：




```
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
```

In our housing data, the price of most properties is between 10M-20M. The number of houses below 1M is also relatively large. The number of expensive houses above 3M is relatively small. The average value of prices in all samples is 180921, with standardization of 79442.



According to the above figure, the correlation between sales price and sale condition is general. It is worth noting that **partial** [1]has a higher positive correlation with prices than other

[1] Partial： Home was not completed when last assessed (associated with New Homes)

conditions. This is very interesting. Usually “unfinished”, in other words renovated or newly built houses, so the price will be higher.

I also used other categorical variables to do correlation analysis, the research found that the best and worst cases of the condition class have a higher correlation to the price, and the median value has almost no effect.

The above figure is very interesting. Through eye observation, we can see that there is a certain linear relationship between overall condition and sale price, but the highest price does not appear in houses with 9 points condition. They are actually housing with 5-6 points. This shows that there are other variables that affect prices.

Linear regression model and test:

The independent variable is total living square feet, where I use the sum of the area of the first floor and the second floor, and overall quality.

```
#separate select data to training and test set
from sklearn.model_selection import train_test_split
X = SLR[['OverallQual','totalSF']]
y = SLR['SalePrice']
X_train, X_test, y_train, y_test =train_test_split(X,y,test_size=0.4, random_state=666)
```

Then I use sklearn to do data partition. I split my data to 60% training set and 40% test set. I built model with training set and test it on testing set. Below is the model result.

```
#build simple linear regression with trainnig set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
model=regressor.fit(X_train, y_train)
coef_SLR=pd.DataFrame(regressor.coef_, X.columns, columns=['Coefficient'])
coef_SLR
```

	Coefficient
OverallQual	32925.612857
totalSF	60.283570

The coefficient of Overall quality is 32925, Total SF is 60.

```
#tested on the testing data
from sklearn.metrics import mean_squared_error, r2_score

predictions = model.predict(X_test)
print('score', model.score(X_test,y_test))
print('r2 score', r2_score(y_test,predictions))
print('Average Sqaured Error', mean_squared_error(y_test, predictions))
```

```
score 0.6783986018032607
r2 score 0.6783986018032607
Average Sqaured Error 1824408939.2166831
```

The test result shows:

Score: 0.68

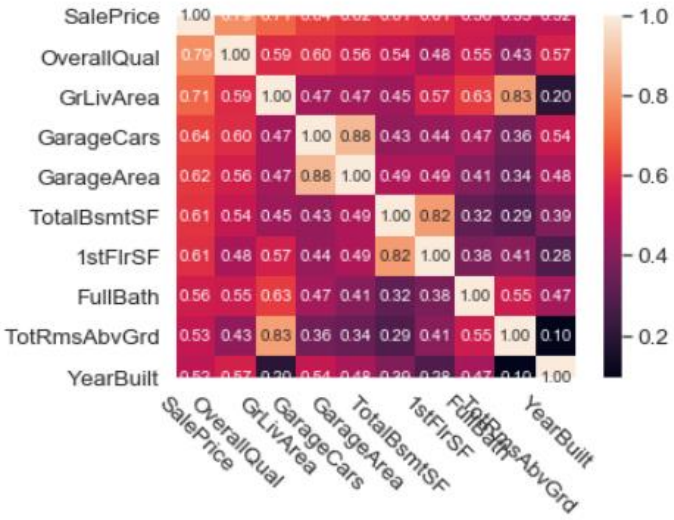
R2 score: 0.67

Average Squared error: 1824408939

Extension

Correlation & use more variables:

From the results of the first model, the effect of our model is not ideal. I think we have missed a lot of important independent variables, so we use correlation methods to filter.



	Most Correlated Features
0	SalePrice
1	OverallQual
2	GrLivArea
3	GarageCars
4	GarageArea
5	TotalBsmtSF
6	1stFlrSF
7	FullBath
8	TotRmsAbvGrd
9	YearBuilt

The 10 features with the strongest correlation are selected from 80 variables. Base on this table above, I save a lot of time to select variables. I use 6 features as independent variables to build a new regression model.

'SalePrice','OverallQual','GrLivArea','GarageCars','FullBath','TotRmsAbvGrd','1stFlrSF'

	Coefficient
OverallQual	26124.720363
GrLivArea	47.566530
GarageCars	16018.858972
FullBath	3316.129318
TotRmsAbvGrd	-2209.658554
1stFlrSF	35.014717

Then I test new model with testing set, the result shows below. This result shows that our model is more efficient on new data

```
score 0.7233816620334339
r2 score 0.7233816620334339
Average Sqaured Error 1569225044.938196
```

A simple app to let user predict price:

In this simple application, users only need to enter their expected characteristics, and the computer automatically calculates the expected price

```
#APP to predict
OverallQuality= int(input("Which Quality you want to livein choose:1 -10 "))
GrLivArea= int(input("Above ground living area you expected: in square feet"))
GarageCars = int(input("GarageCars: Size of garage in car capacity: choose:0-5 "))
FullBath=int(input("Full bathrooms above grade: choose:0-5 "))
TotRmsAbvGrd=int(input("Total rooms above grade (does not include bathrooms):choose 2-15"))
firstFlrSF= int(input("First Floor square feet you expected:"))
price=26124.720363*OverallQuality+47.566530*GrLivArea+16018.858972*GarageCars+3316.129318*FullBath-TotRmsAbvGrd*2209.658554+1

print("According to your choice, the model predicts the price is : ",price)
```

```
Which Quality you want to livein choose:1 -10 5
Above ground living area you expected: in square feet200
GarageCars: Size of garage in car capacity: choose:0-5 1
Full bathrooms above grade: choose:0-5 2
Total rooms above grade (does not include bathrooms):choose 2-152
First Floor square feet you expected:200
According to your choice, the model predicts the price is :  165371.651715
```

Conclusions and what I learnt

We have huge and detailed data, but not all data is helpful. In this project, I feel that understanding the data is the key to ensuring the quality of the model. There are many things that we easily miss out on the huge data.

This project allows me to connect the skills I learned this summer and apply them in a research environment. Not only require me to be proficient in using python skills, it also gives me a lot of space for thinking.