2 Programming Report

Problem 1

Description: We want to use appropriate attributes in “SqFt, Bedrooms, Bathrooms, Neighborhood” to predict
the attributes “Price”.

Step1: use panda to load the csv file.

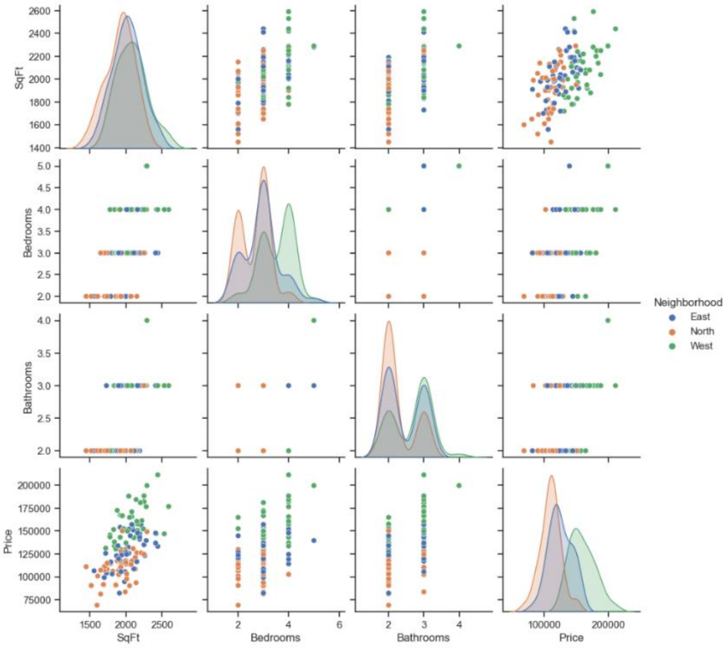
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 128 entries, 0 to 127
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   SqFt          128 non-null    int64
 1   Bedrooms      128 non-null    int64
 2   Bathrooms     128 non-null    int64
 3   Neighborhood  128 non-null    category
 4   Price         128 non-null    int64
dtypes: category(1), int64(4)
memory usage: 4.4 KB
```

	SqFt	Bedrooms	Bathrooms	Price
count	128.000000	128.000000	128.000000	128.000000
mean	2000.937500	3.023438	2.445312	130427.343750
std	211.572431	0.725951	0.514492	26868.770371
min	1450.000000	2.000000	2.000000	69100.000000
25%	1880.000000	3.000000	2.000000	111325.000000
50%	2000.000000	3.000000	2.000000	125950.000000
75%	2140.000000	3.000000	3.000000	148250.000000
max	2590.000000	5.000000	4.000000	211200.000000

there are 128 data samples and 3 feature column, and I guess SqFt is the most important feature contributing to Pirce according to my experience. And also neighborhood is transformed to category to help machine learn it.

Step2: use seaborn library to visualize the dataset.

pairplot:



Just focusing on the last column, SqFt, Bedrooms, Bathrooms and Neighborhood are all correlated highly to Price. And as SqFt, Bedrooms, Bathrooms increases, Price increases. At last, the preference of neighborhood is West > East > North.

Heatmap:

From the last colum of the heatmap, it is clear that SqFt, Bedrooms, Bathrooms is of almost the same correlation.

Step 3: Use sklearn library to process the category variable and use train_test_split in sklearn.model selection to randomly split the data.

Step4:

```
MSE_train: 201089508.69690457
MSE_test: 248597410.68059832
```

Problem 2

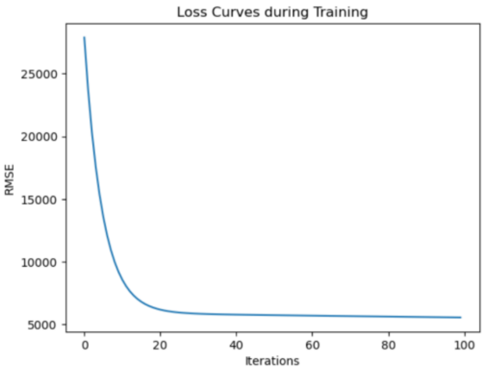
Description: Use gradient descent to train a linear regression model

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 442 entries, 0 to 441
Data columns (total 11 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   age     442 non-null    float64
 1   sex     442 non-null    float64
 2   bmi     442 non-null    float64
 3   bp      442 non-null    float64
 4   s1      442 non-null    float64
 5   s2      442 non-null    float64
 6   s3      442 non-null    float64
 7   s4      442 non-null    float64
 8   s5      442 non-null    float64
 9   s6      442 non-null    float64
 10  target  442 non-null    float64
dtypes: float64(11)
memory usage: 38.1 KB
```

There are 442 samples and 10 features to predict target.

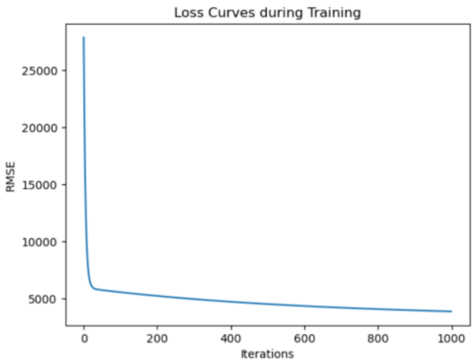
Stopping criterion: I choose when the iteration times is reached because it helps to find the relationship between loss and iteration.

```
step_size: 0.1
iter_step: 100
```



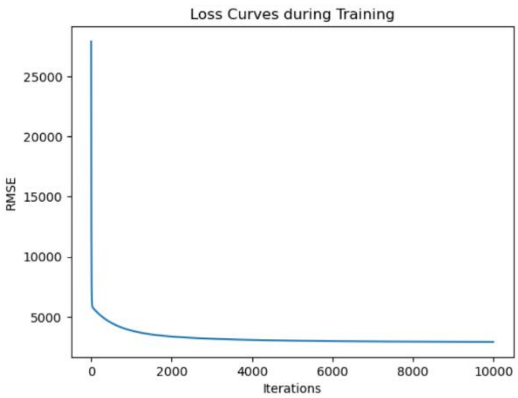

```
MSE_train: 5546.864646667459
MSE_test: 5692.517888348064
```

```
step_size: 0.1
iter_step: 1000
```



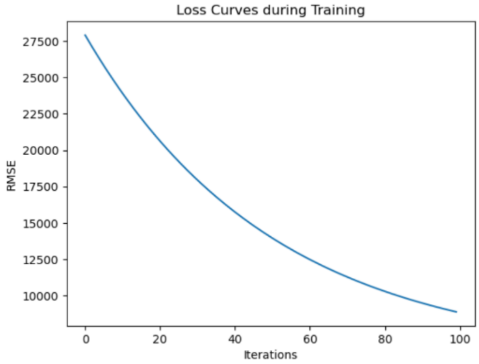

```
MSE_train: 3861.9671931682683
MSE_test: 4271.376747554577
```

```
step_size: 0.1
iter_step: 10000
```



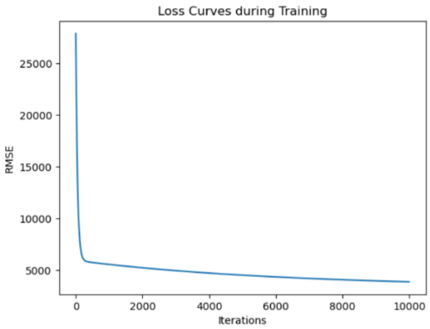

```
MSE_train: 2924.4584061803266
MSE_test: 2994.7085727129615
```

step_size: 0.01
iter_step: 100



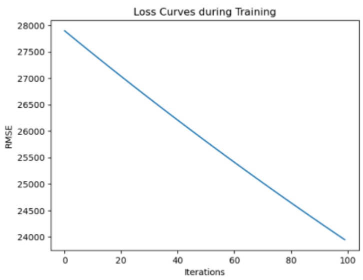
MSE_train: 8833.69602320501
MSE_test: 11026.226586840132

step_size: 0.01
iter_step: 10000



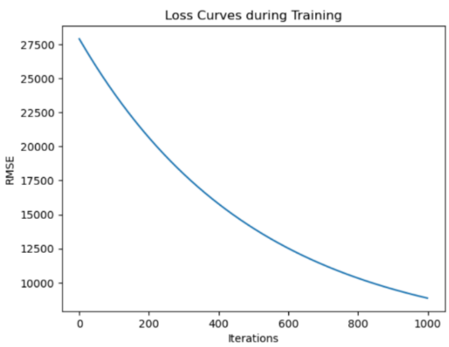
MSE_train: 3862.2433220053304
MSE_test: 4271.560433178849

step_size: 0.001
iter_step: 100



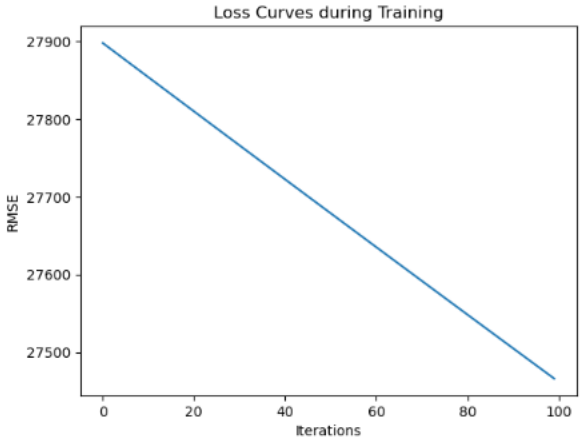
MSE_train: 23909.81557758333
MSE_test: 29205.856619597456

step_size: 0.001
iter_step: 1000



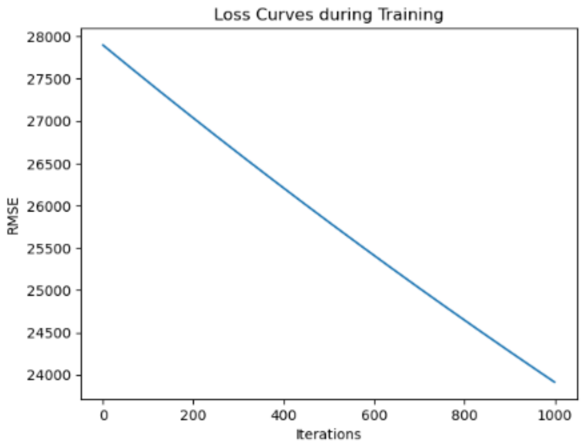
MSE_train: 8860.500356216115
MSE_test: 11062.576980749705

```
step_size: 0.0001
iter_step: 100
```




```
MSE_train: 27462.04538174973
MSE_test: 33250.193370367764
```

```
step_size: 0.0001
iter_step: 1000
```




```
MSE_train: 23911.435241965977
MSE_test: 29207.710836370425
```

Discovery: when step size is relatively large, the loss decreases sharply. When step size is relatively small, it is hard to reach an converge but the accuracy is better. When the number of iteration is large, it actually waste some time since it does not contribute much to the decrease of loss. When the number of iteration is small, it may diverge.