https://colab.research.google.com/drive/1IrVXgOEsdxez5sF393BGoXLxD_vQ8buX?usp=sharing

	sqmtrs	nrooms	view	crime_rate	price
0	251	5	west	low	925701.721399
1	211	3	west	high	622237.482636
2	128	5	east	low	694998.182376
3	178	3	east	high	564689.015926
4	231	3	west	low	811222.970379
5	253	5	north	high	766250.032506
6	101	1	north	low	512749.401548
7	242	1	north	high	637010.760148
8	174	5	west	high	638136.374869
9	328	2	south	high	787704.988273

It is 5 simple datasets. There are not too many features.

```
Data dimensionality is:
(4800, 5)
means
sqmtrs        225.033542
nrooms          2.983958
price      725756.960758
dtype: float64
```

```
] print("std")
train_df.std(axis=0)
```

```
std
sqmtrs        71.851436
nrooms         1.421251
price     151041.121658
dtype: float64
```

	sqmtrs	nrooms	price
count	4800.000000	4800.000000	4.800000e+03
mean	225.033542	2.983958	7.257570e+05
std	71.851436	1.421251	1.510411e+05
min	100.000000	1.000000	3.564985e+05
25%	163.000000	2.000000	6.179536e+05
50%	226.000000	3.000000	7.292999e+05
75%	287.000000	4.000000	8.389284e+05
max	349.000000	5.000000	1.076067e+06

I observe data set via mean and standard derivation also I checked NA rows however all data is fulfilled. I need to play around with some features because their datatypes are not suitable for model building. For example, I need to change directions.

	sqmtrs	nrooms	view	crime_rate	price
0	251	5	0	0	925701.721399
1	211	3	0	1	622237.482636
2	128	5	1	0	694998.182376
3	178	3	1	1	564689.015926
4	231	3	0	0	811222.970379
5	253	5	3	1	766250.032506
6	101	1	3	0	512749.401548
7	242	1	3	1	637010.760148
8	174	5	0	1	638136.374869
9	328	2	2	1	787704.988273

I got this done right away. I replaced the crime rate and building directions with numbers. I didn't do this with "001" and similar 3-bit numbers because I was going to scale it between 0-1 anyway. Then I scaled it and printed the results, now a model is suitable for application.

I removed the view column as label to use in the model.

```
# train one-hidden layered neural networks
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
from statistics import mean

# define your model architecture
layer=[25,50,100]
def model(a,b):
  acc=[]
  for k in layer:

    knn=KNeighborsClassifier(n_neighbors=k)
    scores=cross_val_score(knn, a,b,cv=5,scoring='accuracy')
    acc.append(scores.mean())
  print(acc)

model(train,label)
```

```
[0.2634259259259259, 0.2743055555555555, 0.2777777777777778]
```

I defined the sklearn model with the given k values (25,50,100). It gave the best result k=100.