1.Dataset preprocessing and interpretation

1.1Before the data processing, we can see there are some "?" and some discrete values in the data like below Fig.1

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.c
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0	4356	40	Unitec
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18	Unitec
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0	4356	40	Unitec

Fig.1

After we remove the "?" in the data, the result turn to that in Fig. 2 and the number of the whole data set reduce from **32561** to **30162**.

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.c
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18	Unitec
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900	40	Unitec
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900	40	Unitec

Fig.2

1.2 Explore the number of unique value in the categorical features and labels and then do the transforming work.

Feature	workclass	education	marital.status	occupation	Relationship
Num	7	16	7	14	6
Feature	race	sex	native.country	income	
Num	5	2	41	2	

First, I do the transforming work with dictionary projection which is mapping each unique value in the categorical features into the numerical number.
Then, since for features workclass(Fig.3), occupation(Fig.4), native.country(Fig.5) have too many unique features. After I plot the distribution of these three features below

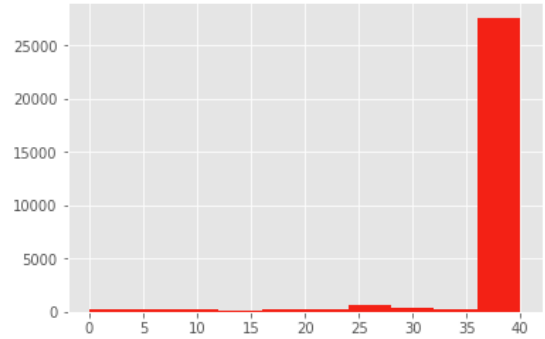

Fig.3

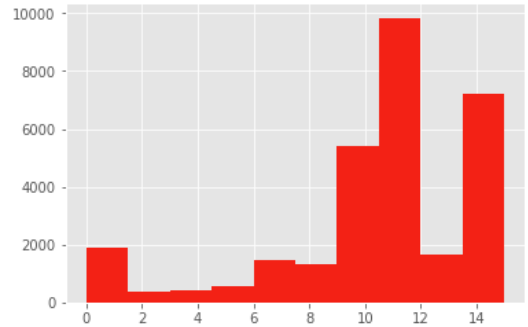

Fig.4

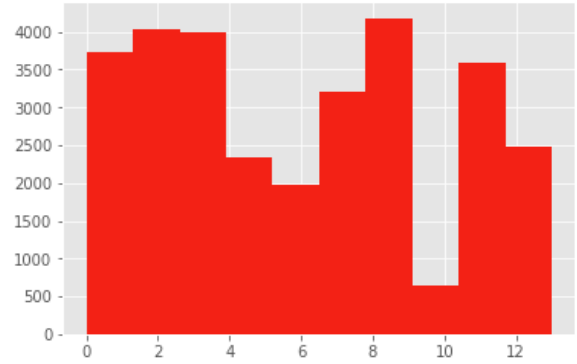

Fig.5

From the distribution graph above, I decided to changing the mapping rule for native.country into None-US and US. Then the distribution of this new mapping result is like below Fig.6. The left bar is none-US and the other is US.

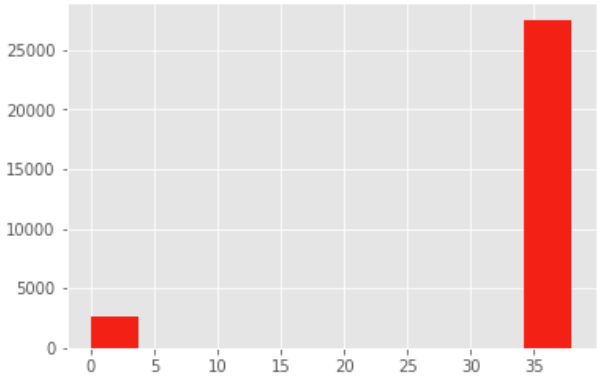

Fig.6

Besides, I also dropped columns education. Because the value in education is corresponding to the value in column education.num. So, I dropped the duplicate one.

1.3 Analysis the features with “gini information gain”.
After I apply the calculation of information gain to the features, I got the result as Fig.7 show

```
The importance of feature sex is 0.01154012701847079
The importance of feature race is 0.01314758678577576
The importance of feature native.country is 0.015229480451674554
The importance of feature capital.loss is 0.035621054080722685
The importance of feature workclass is 0.03835250607457988
The importance of feature occupation is 0.06716144079194923
The importance of feature relationship is 0.0796524495938746
The importance of feature hours.per.week is 0.08071728604970038
The importance of feature marital.status is 0.09781711278646775
The importance of feature capital.gain is 0.11112117613406401
The importance of feature education.num is 0.12161536559326792
The importance of feature age is 0.15042035578163282
The importance of feature fnlwgt is 0.17760405885781944
```

Fig.7

I interpret the information gain as the importance and picked fnlwgt and age as the two features.

1.4 Normalization

Before I used the data to train the model, I applied normalization to the training features. After normalization, the range of each column in the training data is between 0 and 1.The normalization method is helpful for the model to get a more optimal decision hyperplane. And by the way, it saved time to run the model.

2.Implement a linear soft-margin SVM

2.0 I use the features fnlwgt and ages as the two features to train the SVM.

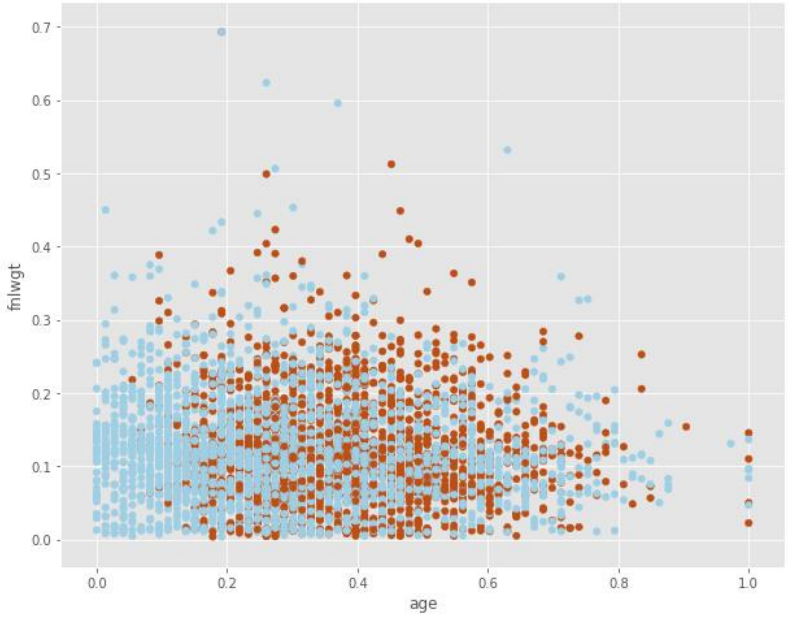

Fig.8

2.1 The C value set to 10. The decision boundary and the support vectors are showed below as Fig.9.

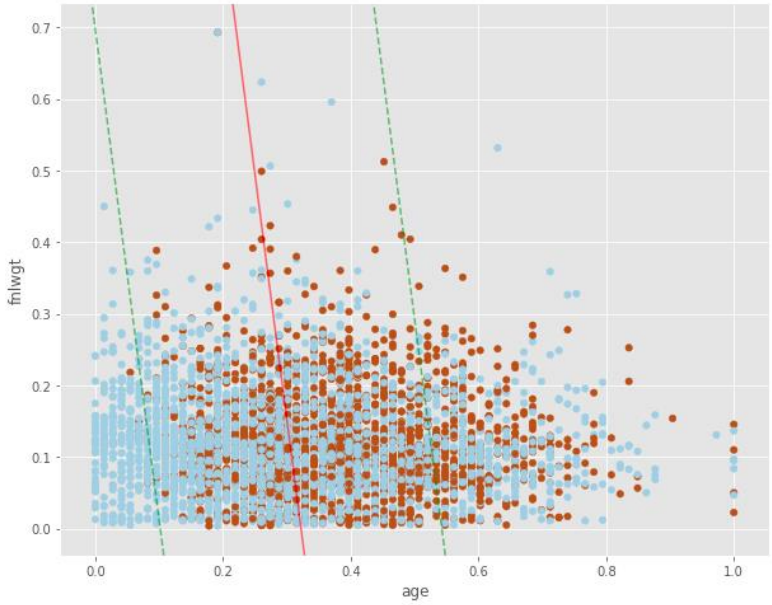

Fig.9

In the Fig.9, the red line is the decision boundary and the two-green dot line are the margin.

2.2 Through the experiment, I tried to increase the value of C from 0.01 to 100 .

Like Fig.10,11,12.
When I increased the C value, the performance of the SVM model would grow better but when the C value was too large, the performance degraded.
And the margin between the decision boundary and the support vector will become smaller.
The reason is that C value determines the extent of the tolerance of the misclassified samples. If the C value is large, it means the SVM model now become strict about the misclassified samples. So, the margin will become smaller to reduce the number of misclassified samples. In the other way, when the C value is small, it means the SVM model is more nice to the misclassified samples. So, the margin becomes larger.

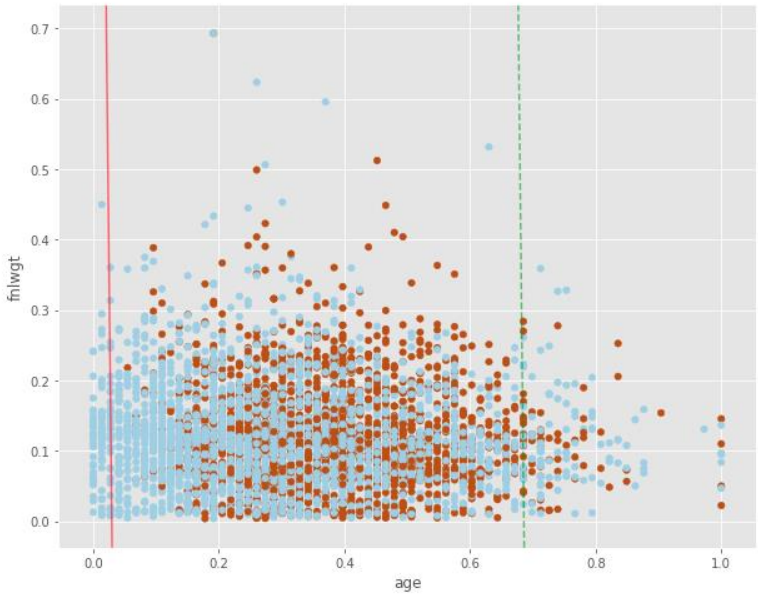

Fig.10 C=0.01

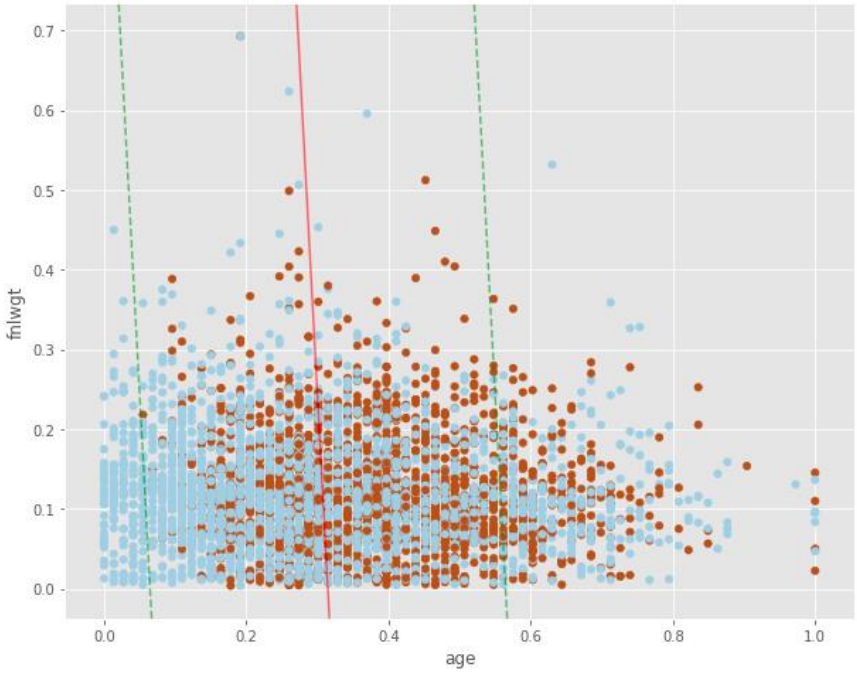

Fig.11 C=1.0

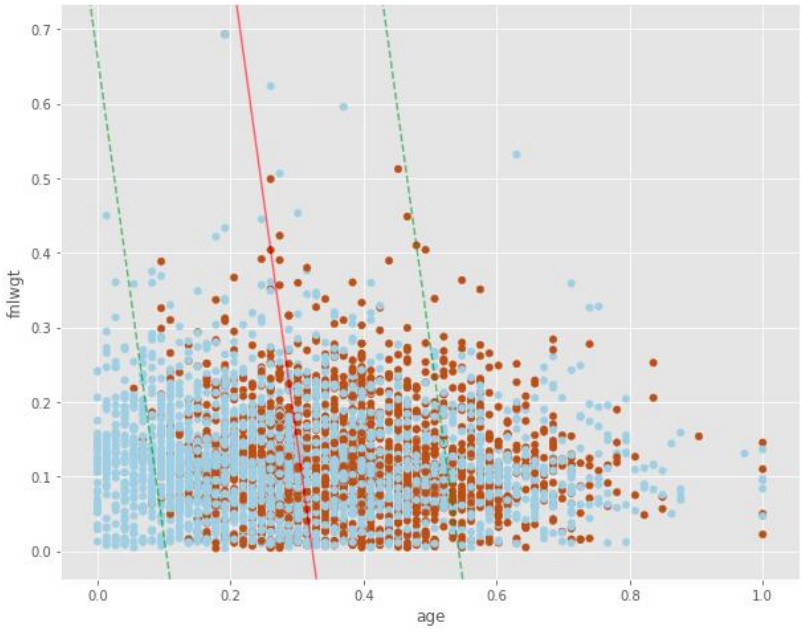

Fig.12 C=100

2.3 When I tried to find a suitable C that made the model perform the best, I tried to set the range of testing C value large and in the next run made it more specific.

C-value	0.001	0.1	10	100	1000
Accuracy	0.7511	0.8299	0.8393	0.8392	0.8390

In the next run, I narrowed down the value range to C value into much smaller region.

C-value	0.05	0.2	0.5	5	15
Accuracy	0.8268	0.8337	0.8376	**0.8395**	0.8393

3.Implement a kernel SVM

3.1 The result of each kernel with each metric is as below

Kernels	Precision	Recall	F1_score	Variance	Accuracy
polynomial	0.7362	0.5511	0.6301	0.1516	0.8290
RBF	0.7515	0.5663	0.6325	0.1483	0.8420
Linear	0.7499	0.4162	0.5351	0.1191	0.8200

P.s. In this result, 1 is the label that >=50 and 0 is the label corresponding to label < 50

In the table, the “RBF” kernel has the highest precision, recall and accuracy. Next is the polynomial. And linear kernel is last. That makes sense. During the exploration of the data, the training data is not easy to be separated by straight line. So through the kernel which projects the training data into another dimension might be helpful for classifying.
3.2 Implemented with my way

3.2.1 Use the “sigmoid” kernel
During the long time tuning the parameter, the performance of sigmoid kernel can only get to The table as below.

“sigmoid” kernel with C = 1.5

Precision	Recall	Variance	F1-score	Accuracy
0.7131	0.5065	0.1456	0.5921	0.8264

3.2.2 Try the random forest model
Besides, I also tried the random forest and get a better result.

Precision	Recall	Variance	F1-score	Accuracy
0.7507	0.6347	0.1662	0.6876	0.8565

As for these four metrics except for “variance” , the performance of the random forest beats the SVM model.

In the graph Fig.13, the blue line is the performance curve of Random Forest model and the red line is for “SVM” model. It tells the “Random forest” model perform better than “SVM”

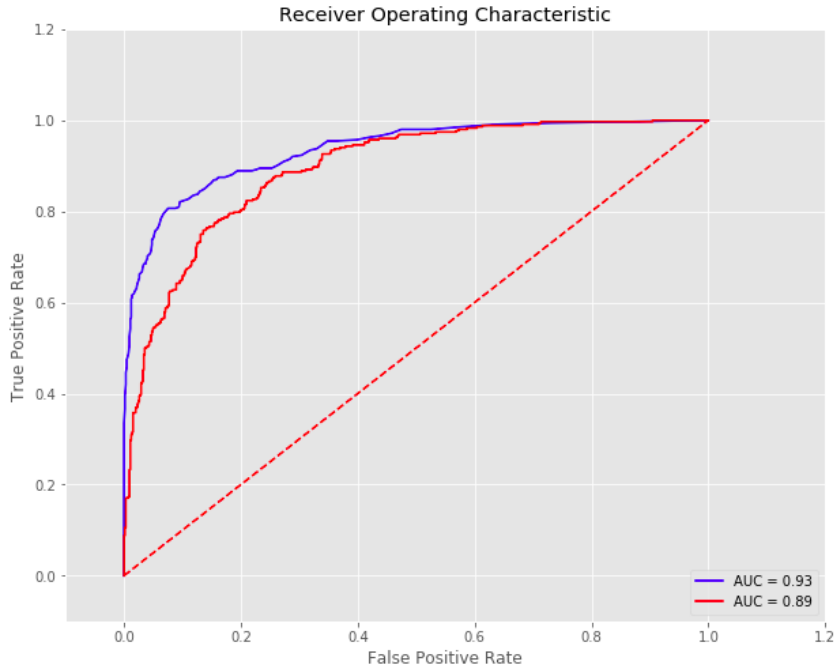

Fig. 13