HW 2

By Zhuo Leng

step 1: read data

In [194]:
```
#import read.py
import read
```

In [36]:
```
df = read.read_data('credit-data.csv')
df.head()
```

Out[36]:

	PersonID	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age	zipcode	Num 59D
0	1	1	0.766127	45	60644	2
1	2	0	0.957151	40	60637	0
2	3	0	0.658180	38	60601	1
3	4	0	0.233810	30	60601	0
4	5	0	0.907239	49	60625	1

step 2: explore data

In [205]:
```
#import explore.py
%matplotlib inline
import explore
```

In [16]:
```
df.describe()
```

Out[16]:

	PersonID	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age
count	150000.000000	150000.000000	150000.000000	150000.0
mean	75000.500000	0.066840	6.048438	52.29520
std	43301.414527	0.249746	249.755371	14.77186
min	1.000000	0.000000	0.000000	0.000000
25%	37500.750000	0.000000	0.029867	41.00000
50%	75000.500000	0.000000	0.154181	52.00000
75%	112500.250000	0.000000	0.559046	63.00000
max	150000.000000	1.000000	50708.000000	109.0000

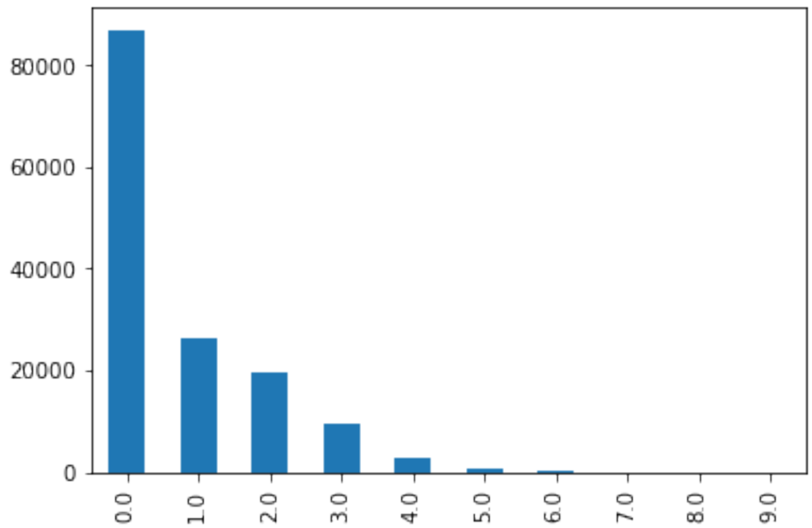
Type *Markdown* and LaTeX: α^2

In [17]:
```
explore.number_count(df, 'NumberOfDependents')
```

Out[17]:
```
0.0    86902
1.0    26316
2.0    19522
3.0     9483
4.0     2862
5.0      746
6.0      158
7.0       51
8.0       24
9.0        5
Name: NumberOfDependents, dtype: int64
```

In [18]:
```
pd.value_counts(df.NumberOfDependents, ascending=False).head(10).plot(kin
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1127b9b38>



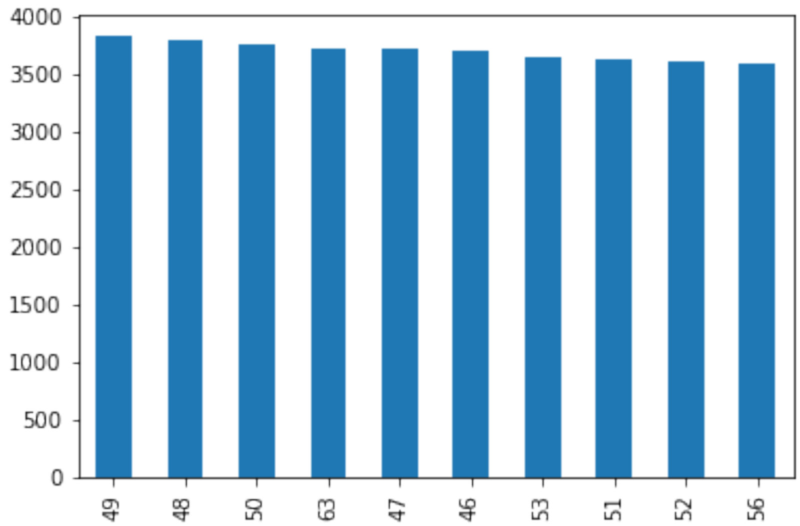
From the top 10 number acount of NumberOfDependents, we could know more than 1/2 of family has 0.0 number of dependents in family excluding themselves(spouse, children etc.). They life on their own.Number of dependents in family excluding themselves (spouse, children etc.)

In [19]:
```
explore.number_count(df, 'age')
```

Out[19]:
```
49    3837
48    3806
50    3753
63    3719
47    3719
46    3714
53    3648
51    3627
52    3609
56    3589
Name: age, dtype: int64
```

In [20]:
```
pd.value_counts(df.age, ascending=False).head(10).plot(kind = 'bar')
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x109849b38>



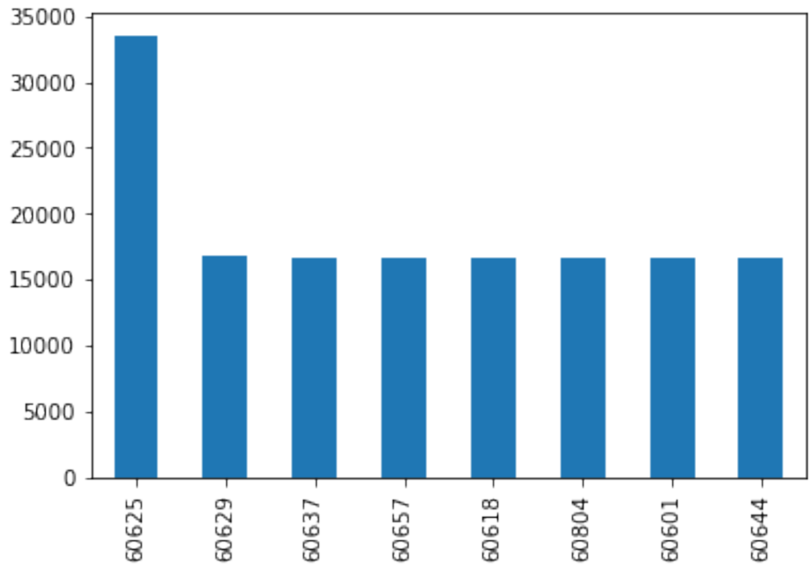
From the top 10 number acount of age, we could know the age of people don't different a lot. Most people are at the age of 49.

In [21]:
```
explore.number_count(df, 'zipcode')
```

Out[21]:
```
60625    33514
60629    16840
60637    16625
60657    16624
60618    16612
60804    16605
60601    16599
60644    16581
Name: zipcode, dtype: int64
```

In [22]:
```
pd.value_counts(df.zipcode, ascending=False).plot(kind = 'bar')
```

Out[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x111f7a160>
```



Take a look at zipcode varibale, most people live in zipcode area 60625.

cross tab

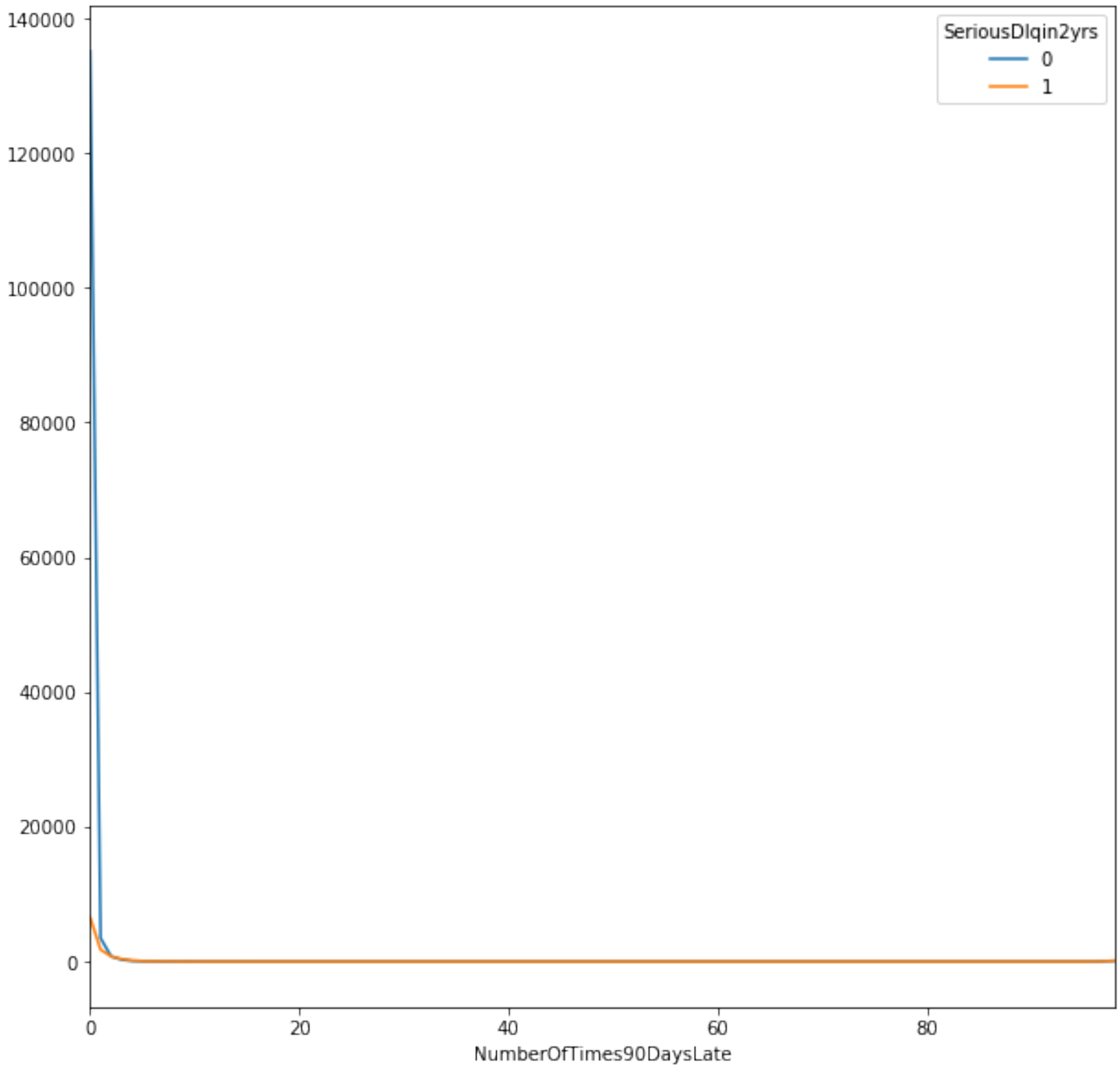
In [23]:
```
explore.crosstable(df, 'NumberOfTimes90DaysLate', 'SeriousDlqin2yrs')
```

Out[23]:

SeriousDlqin2yrs	0	1
NumberOfTimes90DaysLate		
0	135108	6554
1	3478	1765
2	779	776
3	282	385
4	96	195
5	48	83
6	32	48
7	7	31
8	6	15
9	5	14
10	3	5
11	2	3
12	1	1
13	2	2
14	1	1
15	2	0
17	0	1
96	1	4
98	121	143

In [203]:
```
pd.crosstab(df.NumberOfTimes90DaysLate, df.SeriousDlqin2yrs).plot(figsize
```

Out[203]: <matplotlib.axes._subplots.AxesSubplot at 0x127c9fc88>



From the cross table of NumberOfTimes90DaysLate and SeriousDlqin2yrs, most people experienced 90 days past due delinquency or worse have more number of times borrower has been 90 days or more past due. People do not experienced 90 days past due delinquency or worse have less number of times borrower has been 90 days or more past due.

In [25]:

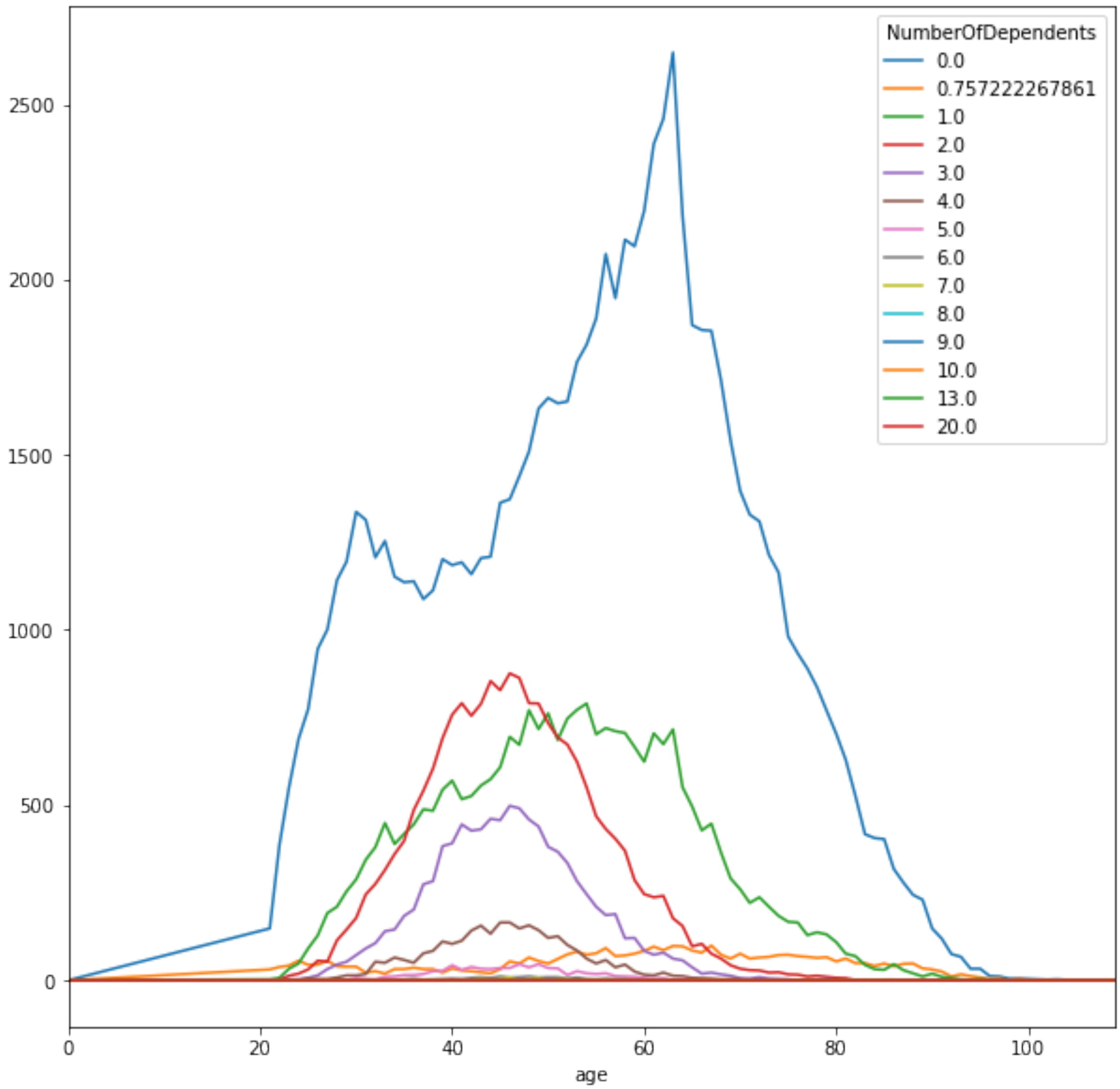
```
pd.crosstab(df.age, df.NumberOfDependents)
```

Out[25]:

NumberOfDependents	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	13.0	20.0
age													
0	0	0	1	0	0	0	0	0	0	0	0	0	0
21	148	3	1	0	0	0	0	0	0	0	0	0	0
22	385	7	2	2	0	0	0	0	0	0	0	0	0
23	550	33	13	3	0	0	0	0	0	0	0	0	0
24	689	48	19	3	1	0	0	0	0	0	0	0	0
25	774	91	31	7	5	1	0	0	0	0	0	0	0
26	946	128	56	14	4	0	0	0	0	0	0	0	0
27	1001	192	53	32	4	0	0	0	0	0	0	0	0
28	1142	210	114	45	8	1	0	0	0	0	0	0	0

In [202]:
```
pd.crosstab(df.age, df.NumberOfDependents).plot(figsize=(10,10))
```

Out[202]: <matplotlib.axes._subplots.AxesSubplot at 0x1207f2940>



From the cross table of age and NumberOfDependents, most people have 9 number of dependents in family excluding themselves (spouse, children etc.). Also, we could know, with the increase of ages, number of dependents in family excluding themselves (spouse, children etc.) will increase. However, when after age 40-60, number of dependents in family excluding themselves (spouse, children etc.) will decrease.

step 3: pre-processing data: handing missing value

In [ ]:
```
#import PreProcess.py
import PreProcess
```

In [37]:
```
df_lng = pd.melt(df)
df_lng.head()
```

Out[37]:

	variable	value
0	PersonID	1.0
1	PersonID	2.0
2	PersonID	3.0
3	PersonID	4.0
4	PersonID	5.0

In [38]:
```
PreProcess.print_null_freq(df)
```

Out[38]:

value	False	True
variable		
DebtRatio	150000	0
MonthlyIncome	120269	29731
NumberOfDependents	146076	3924
NumberOfOpenCreditLinesAndLoans	150000	0
NumberOfTime30-59DaysPastDueNotWorse	150000	0
NumberOfTime60-89DaysPastDueNotWorse	150000	0
NumberOfTimes90DaysLate	150000	0
NumberRealEstateLoansOrLines	150000	0
PersonID	150000	0
RevolvingUtilizationOfUnsecuredLines	150000	0
SeriousDlqin2yrs	150000	0
age	150000	0
zipcode	150000	0

From the crosstable table, we could know that MonthlyIncome and NumberOfDependents are two variables with missing value. We need to fill them with mean value.

In [39]:
```
df.MonthlyIncome.describe()
```

Out[39]:
```
count    1.202690e+05
mean     6.670221e+03
std      1.438467e+04
min      0.000000e+00
25%      3.400000e+03
50%      5.400000e+03
75%      8.249000e+03
max      3.008750e+06
Name: MonthlyIncome, dtype: float64
```

In [40]:
```
#fill MonthlyIncome na with mean
PreProcess.fill_na(mean,df,'MonthlyIncome')
```

In [41]:
```
df.NumberOfDependents.describe()
```

Out[41]:
```
count    146076.000000
mean          0.757222
std           1.115086
min           0.000000
25%           0.000000
50%           0.000000
75%           1.000000
max          20.000000
Name: NumberOfDependents, dtype: float64
```

In [42]:
```
#fill MonthlyIncome na with mean
PreProcess.fill_na(mean,df,'NumberOfDependents')
```

In [48]:
```
#check again for null value by using cross table
PreProcess.print_null_freq(df)
```

Out[48]:

value	False
variable	
DebtRatio	150000
MonthlyIncome	150000
NumberOfDependents	150000
NumberOfOpenCreditLinesAndLoans	150000
NumberOfTime30-59DaysPastDueNotWorse	150000
NumberOfTime60-89DaysPastDueNotWorse	150000
NumberOfTimes90DaysLate	150000
NumberRealEstateLoansOrLines	150000
PersonID	150000
RevolvingUtilizationOfUnsecuredLines	150000
SeriousDlqin2yrs	150000
age	150000
zipcode	150000

```
Now all the variables have 150000 number of no-null value. So we
successfully fill the missing value with mean of relavent variables.
After clean the data, we need to yield a new csv.
```

In [49]:
```
df.to_csv("credit-data-post-import.csv", index=False)
```

step 4: Generate Features/Predictors

In [64]:
```
#import feature.py
import feature
df.head()
```

Out[64]:

oans	NumberOfTimes90DaysLate	NumberRealEstateLoansOrLines	NumberOfTime60-89DaysPastDueNotW
	0	6	0
	0	0	0
	1	0	0
	0	0	0
	0	1	0

In [70]:
```
#categorical features
#SeriousDlqin2yrs
df['SeriousDlqin2yrs_False'] = feature.categorical_var(df,'SeriousDlqin2y
df['SeriousDlqin2yrs_True'] = feature.categorical_var(df,'SeriousDlqin2yr
feature.categorical_var(df,'SeriousDlqin2yrs')
```

Out[70]:

	0	1
0	0	1
1	1	0
2	1	0
3	1	0
4	1	0
5	1	0
6	1	0
7	1	0
8	1	0
9	1	0

In [78]:
```
#discretize continuous variables
#RevolvingUtilizationOfUnsecuredLines
max(df['RevolvingUtilizationOfUnsecuredLines']) - min(df['RevolvingUtiliza
```

Out[78]: 50708.0

In [83]:
```
ationOfUnsecuredLines) = 50708, so we choose bin = 100
edLines_dis'] = feature.discretize_continuous_var(df,'RevolvingUtilization
edLines_dis']
```

```
149984    (-50.708, 507.08]
149985    (-50.708, 507.08]
149986    (-50.708, 507.08]
149987    (-50.708, 507.08]
149988    (-50.708, 507.08]
149989    (-50.708, 507.08]
149990    (-50.708, 507.08]
149991    (-50.708, 507.08]
149992    (-50.708, 507.08]
149993    (-50.708, 507.08]
149994    (-50.708, 507.08]
149995    (-50.708, 507.08]
149996    (-50.708, 507.08]
149997    (-50.708, 507.08]
149998    (-50.708, 507.08]
149999    (-50.708, 507.08]
Name: RevolvingUtilizationOfUnsecuredLines_dis, dtype: category
Categories (100, object): [(-50.708, 507.08] < (507.08, 1014.16] < (10
14.16, 1521.24] < (1521.24, 2028.32] ... (48679.68, 49186.76] < (49186
.76, 49693.84] < (49693.84, 50200.92] < (50200.92, 50708]]
```

In [75]:
```
#discretize continuous variables
#age
max(df['age']) - min(df['age'])
```

Out[75]: 109

In [76]:
```
#because age varibale (max-min) =109, we choose bins = 20
df['age_dis'] = feature.discretize_continuous_var(df,'age', 20)
df['age_dis']
```

Out[76]:
```
0          (43.6, 49.05]
1          (38.15, 43.6]
2          (32.7, 38.15]
3          (27.25, 32.7]
4          (43.6, 49.05]
5          (70.85, 76.3]
6          (54.5, 59.95]
7          (38.15, 43.6]
8          (21.8, 27.25]
9          (54.5, 59.95]
10         (27.25, 32.7]
11         (49.05, 54.5]
12         (43.6, 49.05]
13         (38.15, 43.6]
14         (70.85, 76.3]
15         (59.95, 65.4]
16         (76.3, 81.75]
17         (49.05, 54.5]
18         (38.15, 43.6]
```

In [92]:
```
#categorical features
#zipcode
zip = feature.categorical_var(df,'zipcode')
zip
```

Out[92]:

	60601	60618	60625	60629	60637	60644	60657	60804
0	0	0	0	0	0	1	0	0
1	0	0	0	0	1	0	0	0
2	1	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0
5	0	0	0	1	0	0	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	1
9	0	0	0	1	0	0	0	0

In [95]:
```
df['zip_60601'] = zip.iloc[:,0]
df['zip_60618'] = zip.iloc[:,1]
df['zip_60625'] = zip.iloc[:,2]
df['zip_60629'] = zip.iloc[:,3]
df['zip_60637'] = zip.iloc[:,4]
df['zip_60644'] = zip.iloc[:,5]
df['zip_60657'] = zip.iloc[:,6]
df['zip_60804'] = zip.iloc[:,7]
```

In [105]:
```
#categorical features
#NumberOfTime30-59DaysPastDueNotWorse
num = pd.get_dummies(df.iloc[:,5], prefix = 'NumberOfTime30-59DaysPastDue
num
```

Out[105]:

	NumberOfTime30-59DaysPastDueNotWorse_0	NumberOfTime30-59DaysPastDueNotWorse_1	NumberOfTime30-59DaysPastDueNotWors
0	0	0	1
1	1	0	0
2	0	1	0
3	1	0	0
4	0	1	0
5	1	0	0
6	1	0	0
7	1	0	0
8	1	0	0
9	1	0	0

In [114]:
```
#concat variables to df
df = pd.concat([df, num], axis=1)
df
```

Out[114]:

	PersonID	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age	zipcode
0	1	1	0.766127	45	60644
1	2	0	0.957151	40	60637
2	3	0	0.658180	38	60601
3	4	0	0.233810	30	60601
4	5	0	0.907239	49	60625
5	6	0	0.213179	74	60629
6	7	0	0.305682	57	60637
7	8	0	0.754464	39	60625
8	9	0	0.116951	27	60804
9	10	0	0.189169	57	60629

In [115]:
```
#discretize continuous variables
#DebtRatio
max(df['DebtRatio']) - min(df['DebtRatio'])
```

Out[115]: 329664.0

In [116]:
```
#because age varibale (max-min) =329664, we choose bins = 100
df['DebtRatio_dis'] = feature.discretize_continuous_var(df,'age', 100)
df['age_dis']
```

Out[116]:
```
0         (43.6, 49.05]
1         (38.15, 43.6]
2         (32.7, 38.15]
3         (27.25, 32.7]
4         (43.6, 49.05]
5         (70.85, 76.3]
6         (54.5, 59.95]
7         (38.15, 43.6]
8         (21.8, 27.25]
9         (54.5, 59.95]
10        (27.25, 32.7]
11        (49.05, 54.5]
12        (43.6, 49.05]
13        (38.15, 43.6]
14        (70.85, 76.3]
15        (59.95, 65.4]
16        (76.3, 81.75]
17        (49.05, 54.5]
18        (38.15, 43.6]
```

In [117]:
```
#discretize continuous variables
#MonthlyIncome
max(df['MonthlyIncome']) - min(df['MonthlyIncome'])
```

Out[117]:
```
3008750.0
```

In [118]:
```
#because age varibale (max-min) =329664, we choose bins = 1000
df['MonthlyIncome_dis'] = feature.discretize_continuous_var(df,'MonthlyIn
df['MonthlyIncome_dis']
```

Out[118]:
```
0            (9026.25, 12035]
1         (-3008.75, 3008.75]
2           (3008.75, 6017.5]
3           (3008.75, 6017.5]
4         (63183.75, 66192.5]
5           (3008.75, 6017.5]
6           (6017.5, 9026.25]
7           (3008.75, 6017.5]
8           (6017.5, 9026.25]
9           (21061.25, 24070]
10        (-3008.75, 3008.75]
11          (6017.5, 9026.25]
12          (12035, 15043.75]
13          (12035, 15043.75]
14        (-3008.75, 3008.75]
15           (9026.25, 12035]
16          (6017.5, 9026.25]
17          (6017.5, 9026.25]
18          (3008.75, 6017.5]
```

In [127]:
```
#categorical features
#NumberOfDependents
num = feature.categorical_var(df,'NumberOfDependents')
num
```

Out[127]:

	0.0	0.757222267861	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	13.0	20.0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	1	0	0	0	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	0	0	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	1	0	0	0	0	0	0	0	0	0	0

In [137]:

```
num = num.add_prefix('NumberOfDependents')
num
```

149990	0	0
149991	1	0
149992	0	0
149993	1	0
149994	1	0
149995	1	0
149996	0	0
149997	1	0
149998	1	0
149999	1	0

150000 rows × 14 columns

In [138]:

```
#concat variables to df
df = pd.concat([df, num], axis=1)
df
```

Out[138]:

	PersonID	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age	zipcode
0	1	1	0.766127	45	60644
1	2	0	0.957151	40	60637
2	3	0	0.658180	38	60601
3	4	0	0.233810	30	60601
4	5	0	0.907239	49	60625
5	6	0	0.213179	74	60629
6	7	0	0.305682	57	60637
7	8	0	0.754464	39	60625
8	9	0	0.116951	27	60804
9	10	0	0.189169	57	60629

step 5: build classifier: Logistic Regression

The first model I try to build is a logistic regression model

In [164]:
```
#import model.py
import model
```

In [149]:
```
#the feature I choose for my first model is as below:
#I choose to use all variables in this model, except personID as my featu
# 'SeriousDlqin2yrs' is my dependent variables
#features
features = df.columns[2:13]
features
```

Out[149]:
```
Index(['RevolvingUtilizationOfUnsecuredLines', 'age', 'zipcode',
       'NumberOfTime30-59DaysPastDueNotWorse', 'DebtRatio', 'MonthlyIn
come',
       'NumberOfOpenCreditLinesAndLoans', 'NumberOfTimes90DaysLate',
       'NumberRealEstateLoansOrLines', 'NumberOfTime60-89DaysPastDueNo
tWorse',
       'NumberOfDependents'],
      dtype='object')
```

In [186]:
```
df[features]
```

Out[186]:

	RevolvingUtilizationOfUnsecuredLines	age	zipcode	NumberOfTime30-59DaysPastDueNotWorse	De
0	0.766127	45	60644	2	0.:
1	0.957151	40	60637	0	0.:
2	0.658180	38	60601	1	0.(
3	0.233810	30	60601	0	0.(
4	0.907239	49	60625	1	0.(
5	0.213179	74	60629	0	0.:
6	0.305682	57	60637	0	57
7	0.754464	39	60625	0	0.:
8	0.116951	27	60804	0	46
9	0.189169	57	60629	0	0.(

In [151]:
```
outcome_var = df.columns[1]
outcome_var
```

Out[151]: 'SeriousDlqin2yrs'

In [160]:
```
#build model
lm = model.logistic_regression(df, features, outcome_var)
lm
```

Out[160]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept
=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs
=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0
001,
          verbose=0, warm_start=False)
```

In [171]:
```
print(lm.coef_)
```

```
[[  7.77870825e-06  -2.90907430e-02  -1.70008042e-05   1.22972174e-02
   -1.53991320e-05  -2.80009662e-05  -1.29996122e-03   1.08649202e-02
    4.97825084e-04   8.57494163e-03   1.69177367e-03]]
```

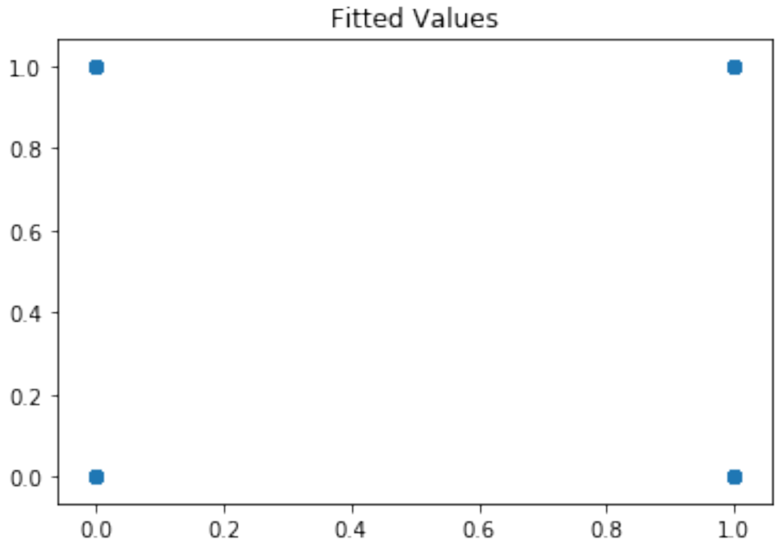
In [172]:
```
# predict
lm.predict(df[features])
```

Out[172]: array([0, 0, 0, ..., 0, 0, 0])

In [180]:
```
# make predictions
expected = df[outcome_var]
predicted = lm.predict(df[features])
```

In [182]:
```
#to see the predicted value vs. actual value
model.plot_pred_actual(df, features, outcome_var)
```

Out[182]: <matplotlib.text.Text at 0x125cb1eb8>



step 6: Evaluate Classifier

In [189]:
```
import evaluation
```

In [185]:
```
#Explained variance score: 1 is perfect prediction
#and 0 means that there is no linear relationship
#between X and y.
evaluation.accuracy(lm, df, features, outcome_var)
```

Out[185]: 0.9333200000000004

From the accuracy score above, we could know there are strong linear relationship between feature and outcome.

In [192]:
```
# summarize the fit of the model by using model.mse
mse = model.mse(lm, df, features, outcome_var)
mse
```

Out[192]: 0.06668000000000003

In [193]:
```
print (lm.intercept_, lm.coef_, mse)
```

```
[ -1.87112635e-07] [[  7.77870825e-06  -2.90907430e-02  -1.70008042e-0
5   1.22972174e-02
   -1.53991320e-05  -2.80009662e-05  -1.29996122e-03   1.08649202e-02
    4.97825084e-04   8.57494163e-03   1.69177367e-03]] 0.06668
```

The mse of this model is 0.06668000000000003, which is not very large. However, we still need compare it with other model's mse in future study.

In [ ]:
```
```