Give Me Some Credit

葛浩　3180103494　　朱祉盈　3180103536

Overview

Overview

Problem Statement

- Improve on the state of the art in credit scoring by predicting the probability that somebody will experience financial distress in the next two years.

Our Goal

- On a given test set, predict the possibility of future financial distress(Y/N)

Overview

Problem Statement

- Improve on the state of the art in credit scoring by predicting the probability that somebody will experience financial distress in the next two years.

Our Goal

- On a given test set, predict the possibility of future financial distress(Y/N)

Overview

- Details about the dataset
  - 13 inputs: features indicating costumer's financial condition
  - 1 output : whether meet over 90 days past due (Y/N)
  - Specific data dictionary

Variable Name	Description	Type
SeriousDlqin2yrs	**Person experienced 90 days past due delinquency or worse**	**Y/N**
RevolvingUtilizationOfUnsecuredLines	Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit	percentage
age	Age of borrower in years	integer
NumberOfTime30-59DaysPastDueNotWorse	Number of times borrower has been 30-59 days past due but no worse in	integer
DebtRatio	Monthly debt payments, alimony,living costs divided by monthy gross income	percentage
MonthlyIncome	Monthly income	real
NumberOfOpenCreditLinesAndLoans	Number of Open loans (installment like car loan or mortgage) and Lines of credit (e.g. credit cards)	integer
NumberOfTimes90DaysLate	Number of times borrower has been 90 days or more past due.	integer
NumberRealEstateLoansOrLines	Number of mortgage and real estate loans including home equity lines of credit	integer
NumberOfTime60-89DaysPastDueNotWorse	Number of times borrower has been 60-89 days past due but no worse in the last 2 years.	integer
NumberOfDependents	Number of dependents in family excluding themselves (spouse, children etc.)	integer

Data Cleaning

Data Cleaning

- Data Preprocessing
- Feature Engineering

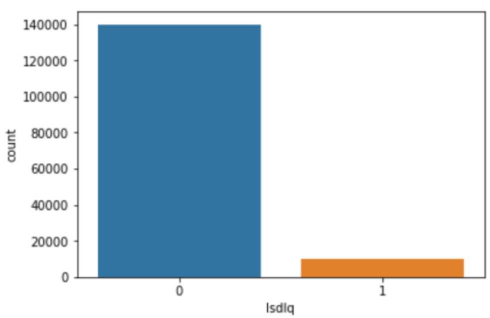
Data Cleaning |Analysis

- drop unnamed columns & rename remaining columns

	Isdlq	Revol	age	Num30-59late	DebtRatio	MonthlyIncome	Numopen	Num90late	Numestate	Num60-89late	Numdepend
0	1	0.766127	45	2	0.802982	9120.0	13	0	6	0	2.0
1	0	0.957151	40	0	0.121876	2600.0	4	0	0	0	1.0
2	0	0.658180	38	1	0.085113	3042.0	2	1	0	0	0.0
3	0	0.233810	30	0	0.036050	3300.0	5	0	0	0	0.0
4	0	0.907239	49	1	0.024926	63588.0	7	0	1	0	0.0

Data Cleaning | Analysis

- Y/N ratio in training set
- Missing Values
- Correlation Matrix




```
Isdlq               0
Revol               0
age                 0
Num30-59late        0
DebtRatio           0
MonthlyIncome   29731
Numopen             0
Num90late           0
Numestate           0
Num60-89late        0
Numdepend        3924
dtype: int64
```

	Isdlq	Revol	age	Num30-59late	DebtRatio	MonthlyIncome	Numopen	Num90late	Numestate	Num60-89late	Numdepend
Isdlq	1.000000	-0.001802	-0.115386	0.125587	-0.007602	-0.019746	-0.029669	0.117175	-0.007038	0.102261	0.046048
Revol	-0.001802	1.000000	-0.005898	-0.001314	0.003961	0.007124	-0.011281	-0.001061	0.006235	-0.001048	0.001557
age	-0.115386	-0.005898	1.000000	-0.062995	0.024188	0.037717	0.147705	-0.061005	0.033150	-0.057159	-0.213303
Num30-59late	0.125587	-0.001314	-0.062995	1.000000	-0.006542	-0.010217	-0.055312	0.983603	-0.030565	0.987005	-0.002680
DebtRatio	-0.007602	0.003961	0.024188	-0.006542	1.000000	-0.028712	0.049565	-0.008320	0.120046	-0.007533	-0.040673
MonthlyIncome	-0.019746	0.007124	0.037717	-0.010217	-0.028712	1.000000	0.091455	-0.012743	0.124959	-0.011116	0.062647
Numopen	-0.029669	-0.011281	0.147705	-0.055312	0.049565	0.091455	1.000000	-0.079984	0.433959	-0.071077	0.065322
Num90late	0.117175	-0.001061	-0.061005	0.983603	-0.008320	-0.012743	-0.079984	1.000000	-0.045205	0.992796	-0.010176
Numestate	-0.007038	0.006235	0.033150	-0.030565	0.120046	0.124959	0.433959	-0.045205	1.000000	-0.039722	0.124684
Num60-89late	0.102261	-0.001048	-0.057159	0.987005	-0.007533	-0.011116	-0.071077	0.992796	-0.039722	1.000000	-0.010922
Numdepend	0.046048	0.001557	-0.213303	-0.002680	-0.040673	0.062647	0.065322	-0.010176	0.124684	-0.010922	1.000000

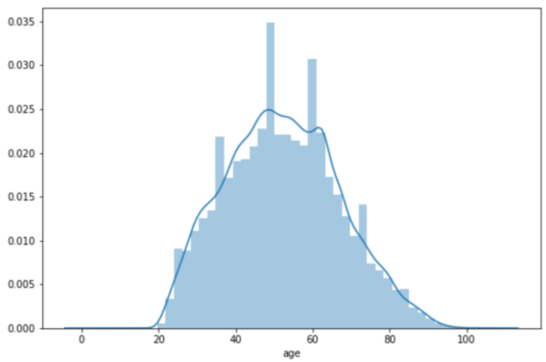
-> potential noises and unbalance in data

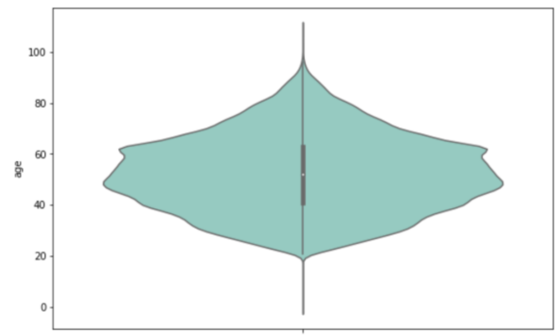
-> For missing values, no need to delete the whole row

Data Cleaning | Analysis

Look at Each Attribute

- Age distribution





-> approximately fit normal distribution

-> calculate abnormal bound accordingly

In [17]:

```
#异常值情况
age_mean=train_set['age'].mean()
age_std=train_set['age'].std()
age_lowlimit=age_mean-3*age_std
age_uplimit=age_mean+3*age_std
print('异常值下限：',age_lowlimit,'异常值上限：',age_uplimit)
```

```
异常值下限： 7.979609077364238 异常值上限： 96.6108042559691
```

In [20]:

```
#筛选异常值
age_lowlimitd=train_set.loc[train_set['age']<age_lowlimit,:]
age_uplimitd=train_set.loc[train_set['age']>age_uplimit,:]
print('异常值下限比例：{0}%'.format(age_lowlimitd.shape[0]*100/train_set.shape[0]),
     '异常值上限比例：{0}%'.format(age_uplimitd.shape[0]*100/train_set.shape[0]))
```

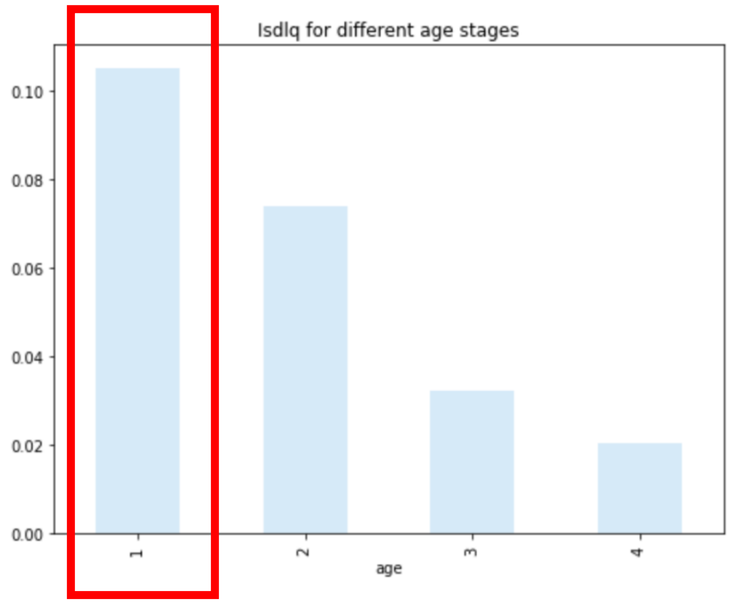
```
异常值下限比例：0.0006666666666666666% 异常值上限比例：0.03%
```

Data Cleaning | Analysis

Look at Each Attribute

- Age

use histogram to reveal the relation between age and result



-> default rates decrease with age

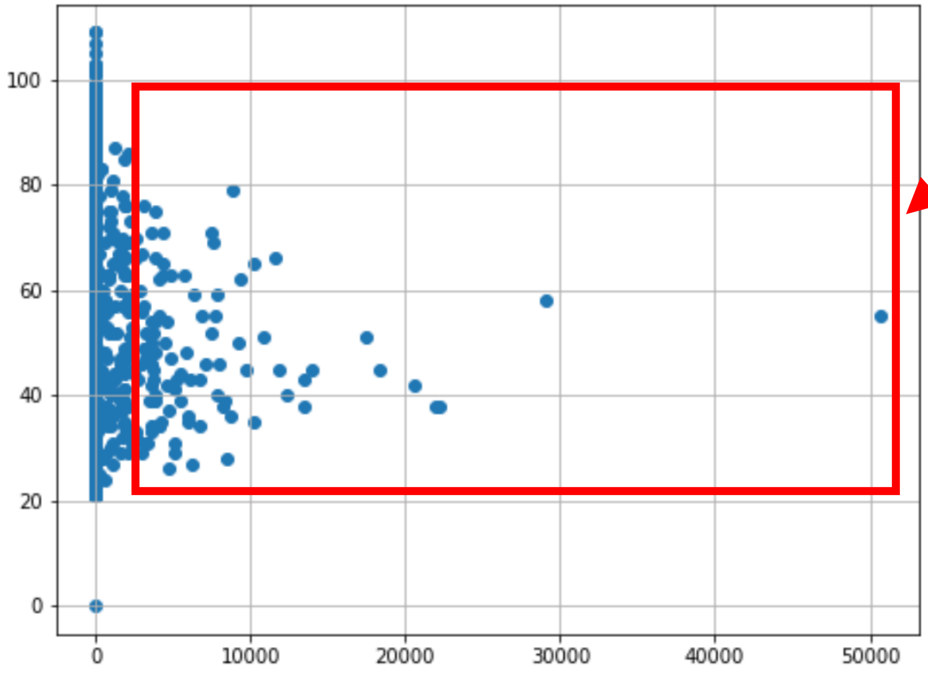
-> 18-40 is the age stage with highest default rate

Data Cleaning | Analysis

Apply similar techniques to other attributes

- Rovel ($\in(0,1)$)

distribution



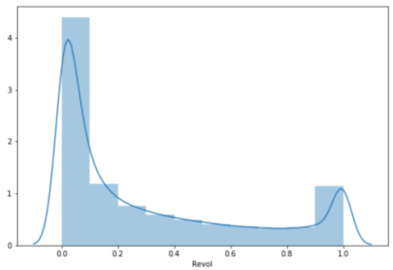
abnormal

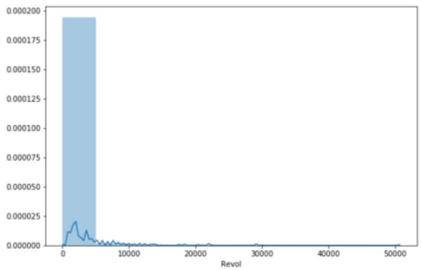
Data Cleaning | Analysis

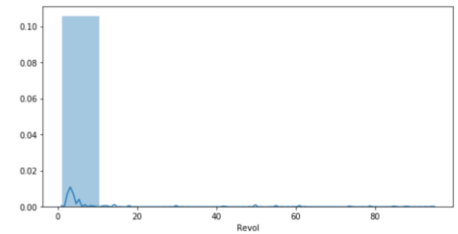
Apply similar techniques to other attributes

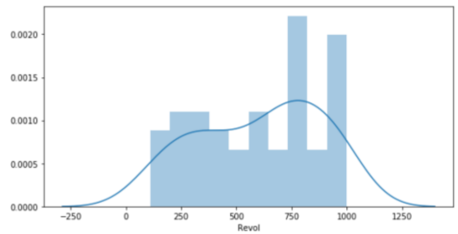
- Rovel (∈(0,1))

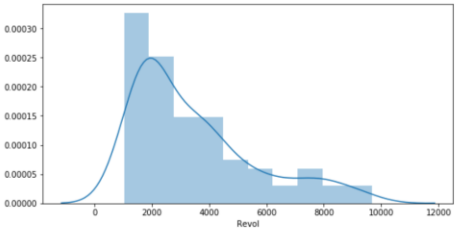
Find potential unnormalized data

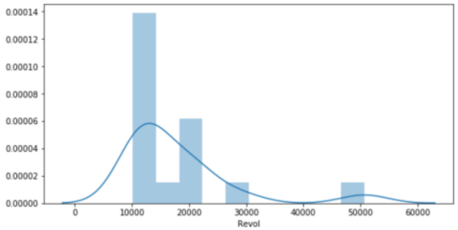












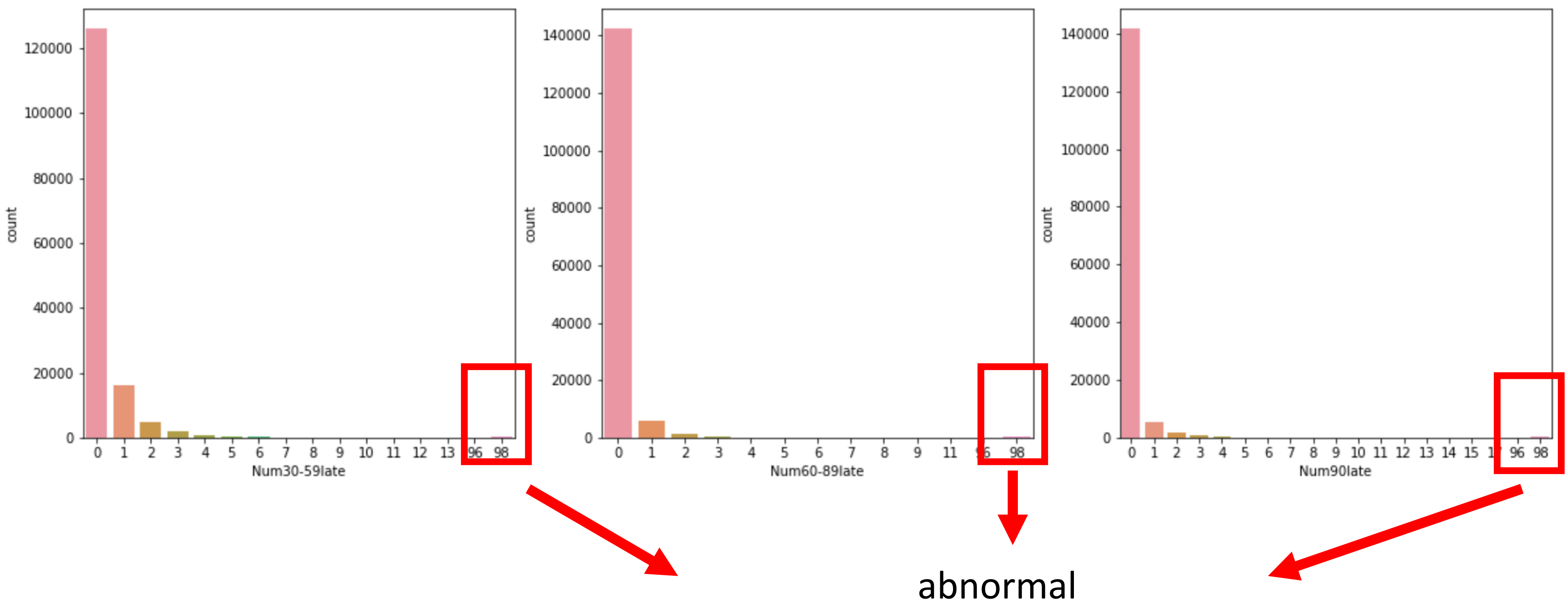
0-1违约率为：5.989963317014633% 1-10违约率为：39.52211817888279% 10-20违约率为：57.142857142857146% 20-100违约率为：18.181818181818183% 100-1000违约率为：1.9607843137254901% 1000-10000违约率为：6.410256410256411% 10000-51000违约率为：0.0%

->threshold for abnormal values is about 20

Data Cleaning | Analysis

Apply similar techniques to other attributes

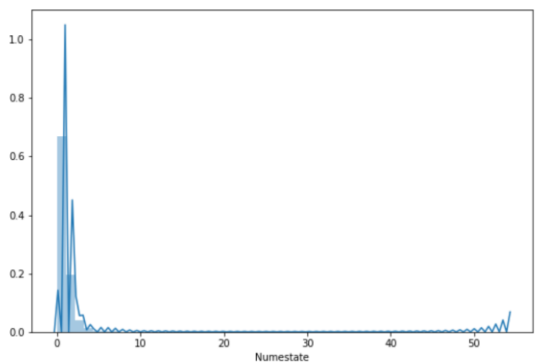
- Num30-59late Num60-89late Num90late

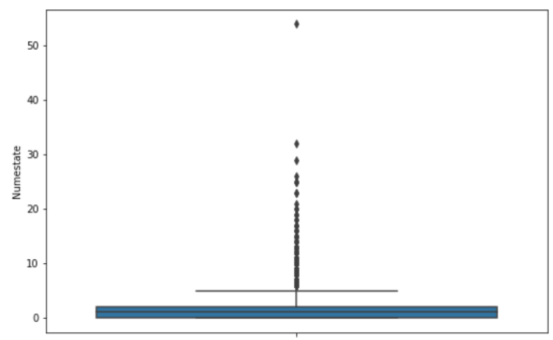


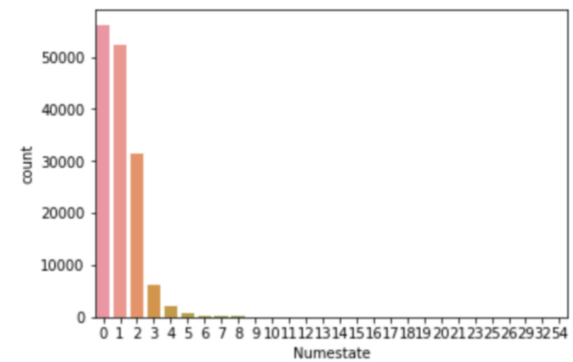
Data Cleaning | Analysis

Apply similar techniques to other attributes

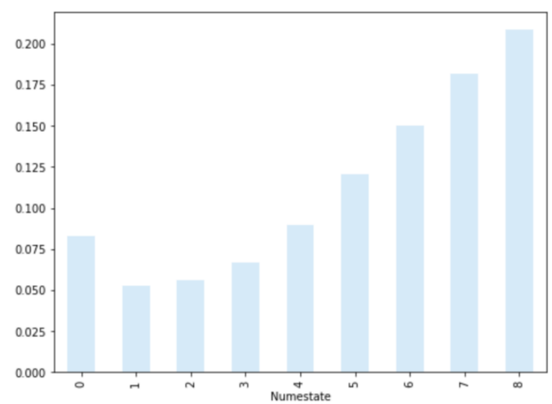
- numstate Distribution







- Relation with y



Data Cleaning | Feature Engineering

Preprocessing

- abnormal values

```
#age异常值处理
train_set=train_set[train_set['age']>0]

#Num30-59late Num60-89late Num90late异常值处理
train_set=train_set[train_set['Num30-59late']<90]
train_set=train_set[train_set['Num60-89late']<90]
train_set=train_set[train_set['Num90late']<90]

#Numestate异常值处理
train_set=train_set[train_set['Numestate']<50]
```

- missing values

```
#Numdepend缺失值处理
train_set['Numdepend']=train_set['Numdepend'].fillna('0')

#MonthlyIncome缺失值处理
#随机森林预测缺失值
data_Forest=train_set.iloc[:,[5,1,2,3,4,6,7,8,9]]
MonthlyIncome_isnull=data_Forest.loc[train_set['MonthlyIncome'].isnull(),:]
MonthlyIncome_notnull=data_Forest.loc[train_set['MonthlyIncome'].notnull(),:]

from sklearn.ensemble import RandomForestRegressor
X=MonthlyIncome_notnull.iloc[:,1:].values
y=MonthlyIncome_notnull.iloc[:,0].values
regr=RandomForestRegressor(max_depth=3, random_state=0,n_estimators=200,n_jobs=-1)
regr.fit(X,y)
MonthlyIncome_fillvalue=regr.predict(MonthlyIncome_isnull.iloc[:,1:].values).round(0)

#填充MonthlyIncome缺失值
train_set.loc[train_set['MonthlyIncome'].isnull(),'MonthlyIncome']=MonthlyIncome_fillvalue
```

Feature Extraction and Binning

- feature crossing

```
#衍生变量
train_set['AllNumlate']=train_set['Num30-59late']+train_set['Num60-89late']+train_set['Num90late']
train_set['Monthlypayment']=train_set['DebtRatio']*train_set['MonthlyIncome']
train_set['Withdepend']=train_set['Numdepend']
```

- adjust data type

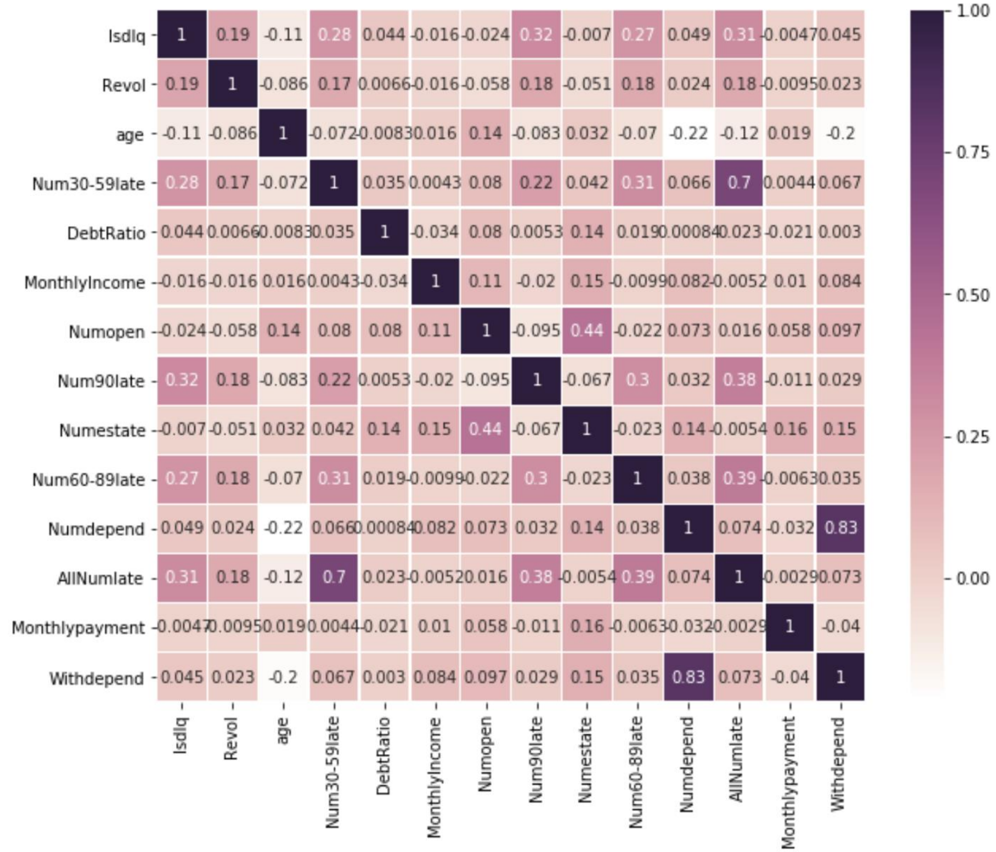
```
#数据类型转换
train_set['Numdepend']=train_set['Numdepend'].astype('int64')
train_set['Withdepend']=train_set['Withdepend'].astype('int64')
train_set['MonthlyIncome']=train_set['MonthlyIncome'].astype('int64')
train_set['Monthlypayment']=train_set['Monthlypayment'].astype('int64')
```

- Binning

```
#Revol分箱
train_set.loc[(train_set['Revol']<1),'Revol']=0
train_set.loc[(train_set['Revol']>1)&(train_set['Revol']<=20),'Revol']=1
train_set.loc[(train_set['Revol']>20),'Revol']=0#根据前文EDA分析，将大于20的数据与0-1的数据合并

#DebtRatio分箱
train_set.loc[(train_set['DebtRatio']<1),'DebtRatio']=0
train_set.loc[(train_set['DebtRatio']>1)&(train_set['DebtRatio']<2),'DebtRatio']=1
train_set.loc[(train_set['DebtRatio']>=2),'DebtRatio']=0

#Num30-59late/Num60-89late/Num90late/Numestate/Numdepend
train_set.loc[(train_set['Num30-59late']>=8), 'Num30-59late'] = 8
train_set.loc[(train_set['Num60-89late']>=7), 'Num60-89late'] = 7
train_set.loc[(train_set['Num90late']>=10), 'Num90late'] = 10
train_set.loc[(train_set['Numestate']>=8), 'Numestate'] = 8
train_set.loc[(train_set['Numdepend']>=7), 'Numdepend'] = 7

#AllNumlate分箱
train_set.loc[(train_set['AllNumlate']>1),'AllNumlate']=1#分为逾期和未逾期两种情况

#Withdepend分箱
train_set.loc[(train_set['Withdepend']>1),'Withdepend']=1#分为独生子女和非独生子女
```

Data Cleaning | Feature Selection

Use heatmap to visualize the new correlation matrix



Data Cleaning | Feature Selection

Calculate WOE, IV

- WOE(weight of evidence)

  1.Binning

  2.Calculate WOE for each bin

$$WOE_i = \ln \frac{py_i}{pn_i} = \ln \frac{\#y_i/n_i}{\#y_T/n_T}$$

  3. WOE_i indicates the predictivity for Bin_i

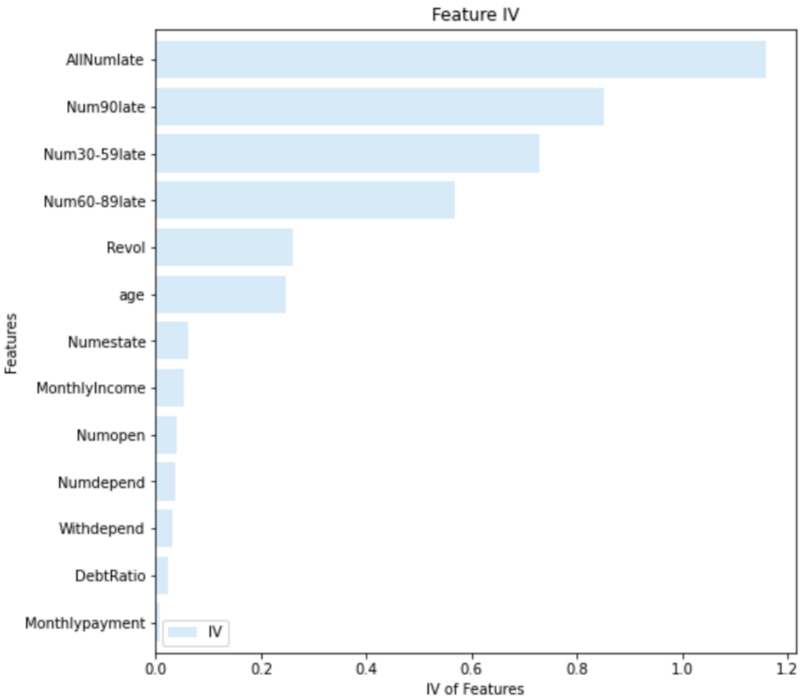
Data Cleaning | Feature Selection

Calculate WOE, IV

- IV (Information Value)
  1. $\mathbf{IV} = \sum_{i=1}^{n} IV_i$
  2. Calculate IV_i for each attribute
  $$IV_i = (py_i - pn_i) * WOE_i$$
  3. IV_i indicates the predictivity for $Attribute_i$

Data Cleaning | Feature Selection

Visualize IV of every attribute



- Filter out variables with IV values greater than 0.1： 'Num30-59late', 'Num60-89late' , 'Num90late' , 'AllNumlate', 'Revol', 'age'；
- There is a strong correlation between‘Num30-59late’ and‘AllNumlate’ (0.7). Choose the one with higher IV ('AllNumlate’)
- Final choice:
  ['Num60-89late','Num90late’,'AllNumlate',’ Revol','age']

Model and Idea Introduction

from Logistic Regression to Deep Learning

从逻辑回归到深度学习到自编码，
主要介绍创新点的灵感来源

Logistic Regression

Step 1: Function Set

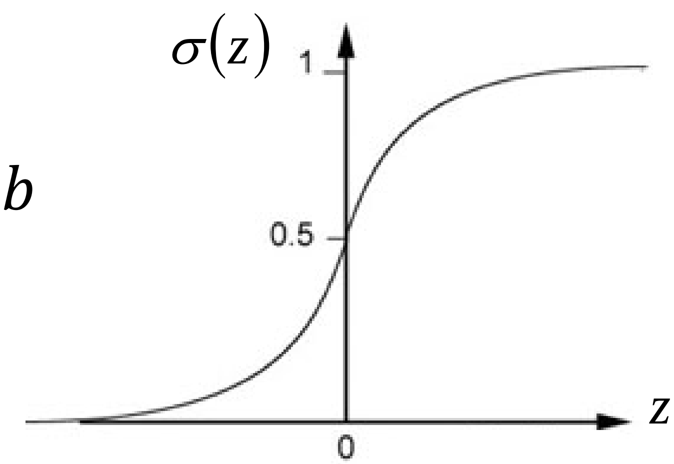
Function set: Including all different w and b

$$\begin{cases} z \geq 0 & \text{class 1} \\ z < 0 & \text{class 2} \end{cases}$$

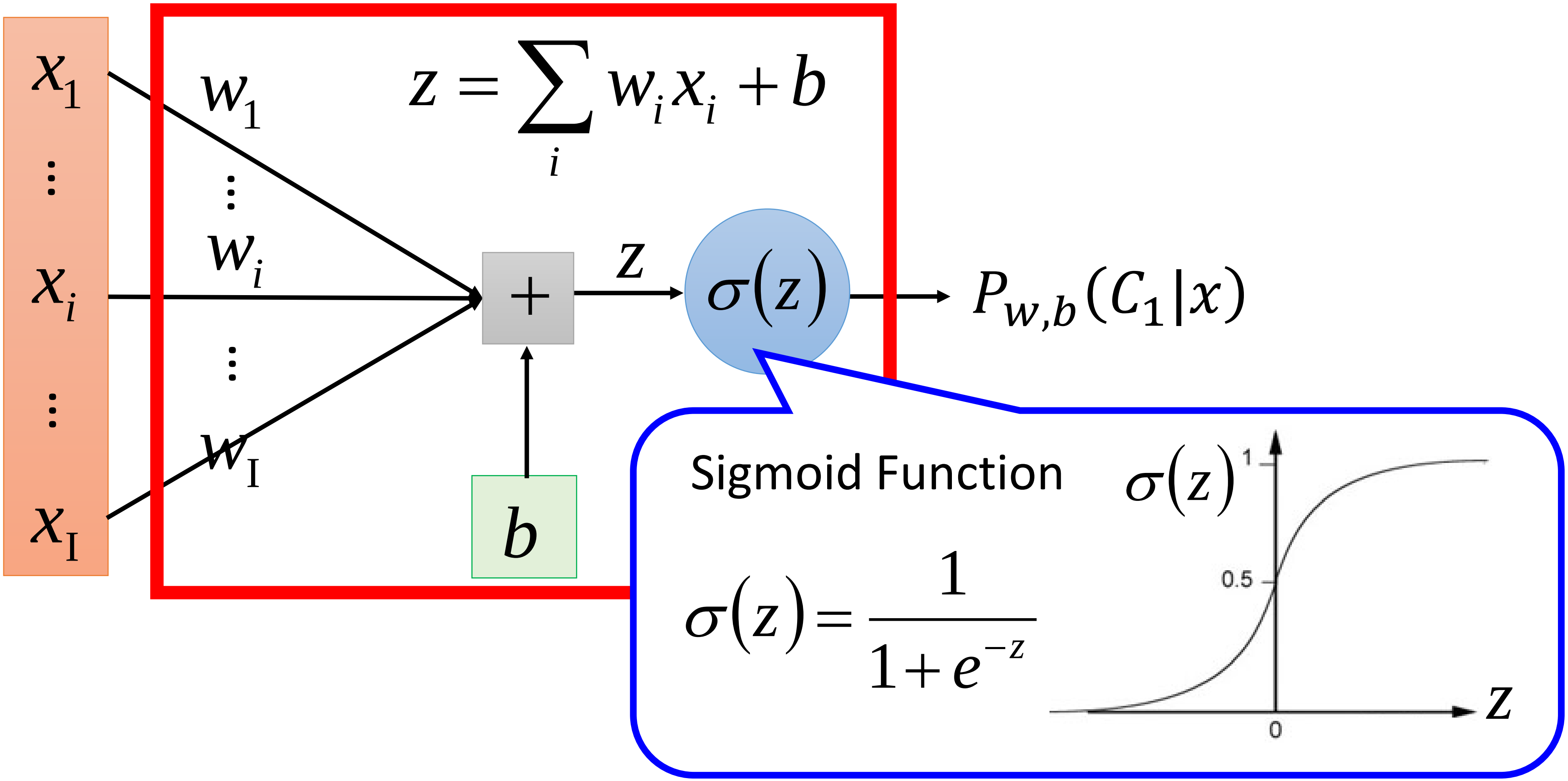
$$P_{w,b}(C_1|x) = \sigma(z)$$

$$z = w \cdot x + b = \sum_i w_i x_i + b$$

$$\sigma(z) = \frac{1}{1 + exp(-z)}$$



Step 1: Function Set



Step 2: Goodness of a Function

Training Data	x^1	x^2	x^3	……	x^N
	C_1	C_1	C_2		C_1

Assume the data is generated based on $f_{w,b}(x) = P_{w,b}(C_1|x)$

Given a set of w and b, what is its probability of generating the data?

$$L(w,b) = f_{w,b}(x^1) f_{w,b}(x^2) \left(1 - f_{w,b}(x^3)\right) \cdots f_{w,b}(x^N)$$

The most likely w^* and b^* is the one with the largest $L(w,b)$.

$$w^*, b^* = arg \max_{w,b} L(w,b)$$

Step 2: Goodness of a Function

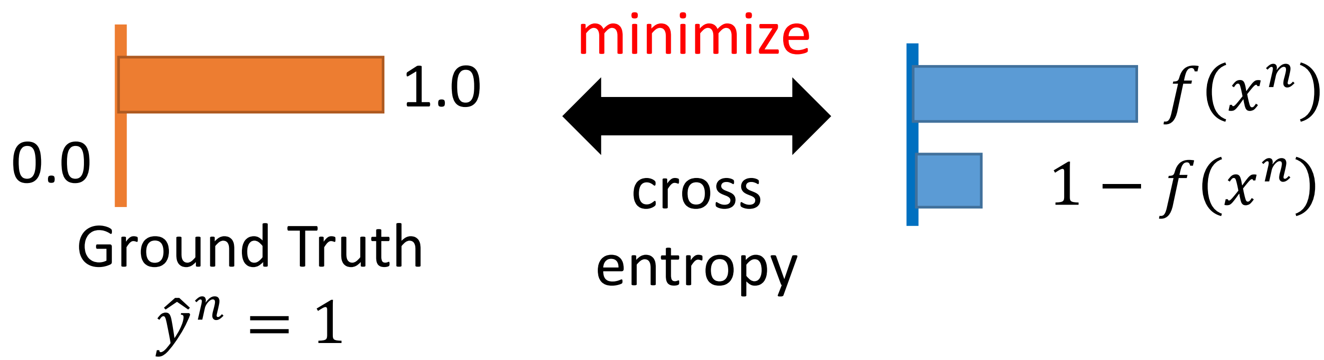
$$L(w,b) = f_{w,b}(x^1) f_{w,b}(x^2) \left(1 - f_{w,b}(x^3)\right) \cdots f_{w,b}(x^N)$$

$$-lnL(w,b) = ln f_{w,b}(x^1) + ln f_{w,b}(x^2) + ln\left(1 - f_{w,b}(x^3)\right) \cdots$$

$\hat{y}^n$: 1 for class 1, 0 for class 2

$$= \sum_n - \underline{\left[\hat{y}^n ln f_{w,b}(x^n) + (1 - \hat{y}^n) ln\left(1 - f_{w,b}(x^n)\right)\right]}$$

Cross entropy between two Bernoulli distribution



Step 3: Find the best function

Gradient:

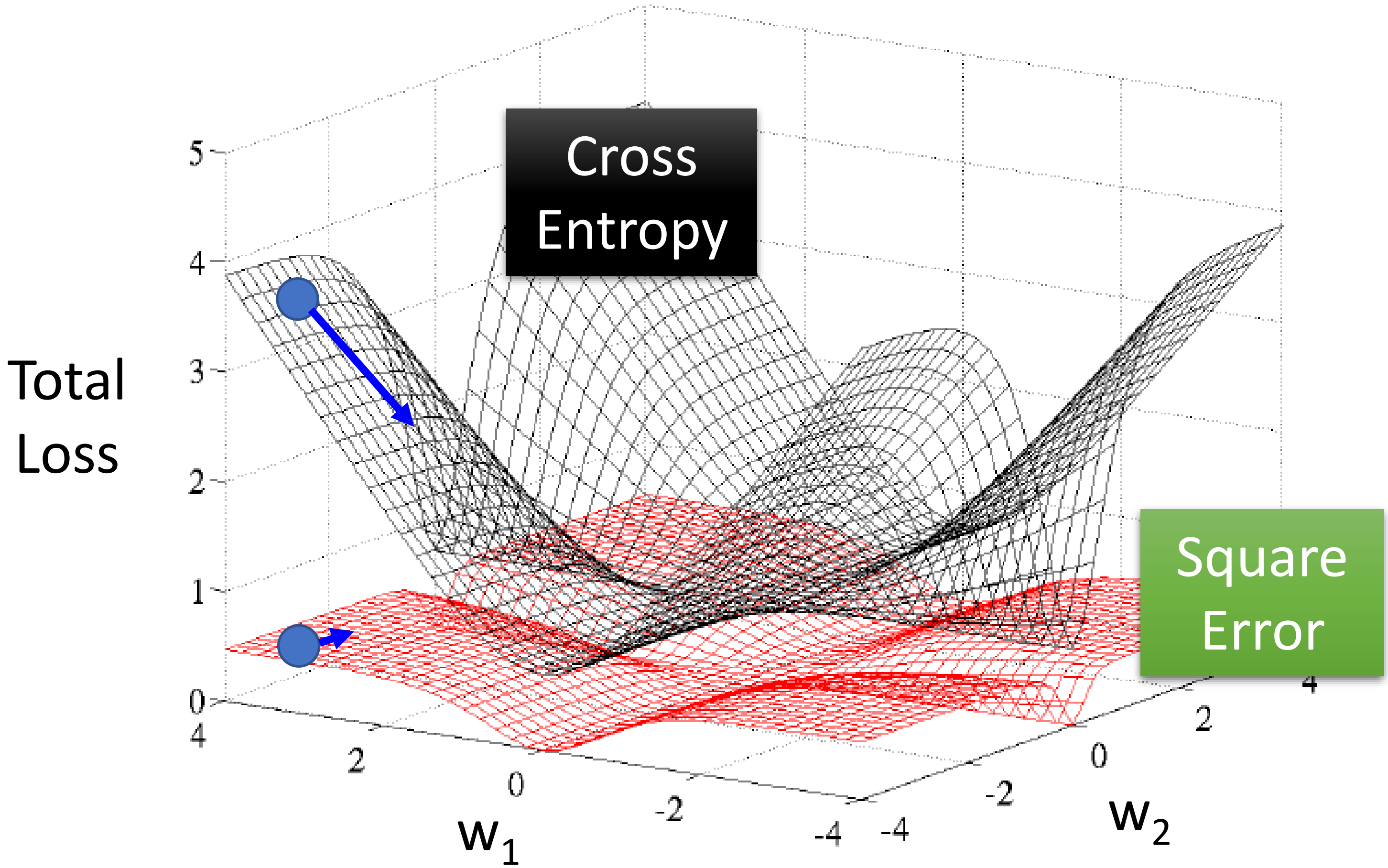
$$\frac{-lnL(w,b)}{\partial w_i} = \sum_n -\left(\hat{y}^n - f_{w,b}(x^n)\right) x_i^n$$

Gradient Descent:

$$w_i \leftarrow w_i - \eta \sum_n -\left(\underline{\hat{y}^n - f_{w,b}(x^n)}\right) x_i^n$$

Larger difference, larger update

Cross Entropy v.s. Square Error



Logistic Regression

Step 1:

$$f_{w,b}(x) = \sigma\left(\sum_i w_i x_i + b\right)$$

Output: between 0 and 1

Step 2:

$$L(f) = \sum_n l(f(x^n), \hat{y}^n)$$

Training data: $(x^n, \hat{y}^n)$

$\boxed{\hat{y}^n\text{: 1 for class 1, 0 for class 2}}$

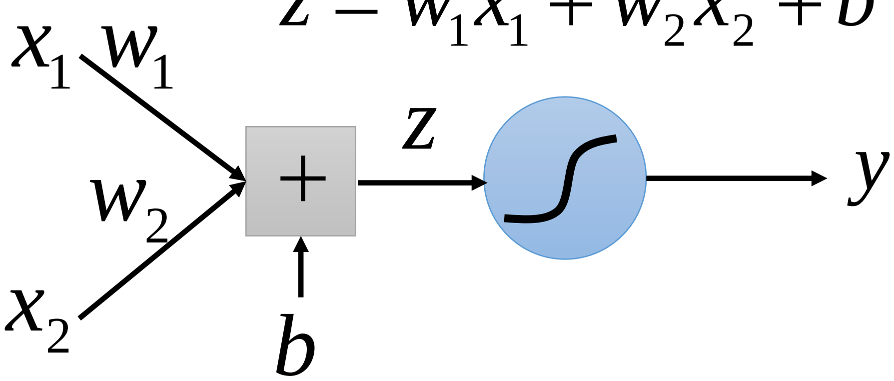
Step 3:

Logistic regression: $w_i \leftarrow w_i - \eta \sum_n -\left(\hat{y}^n - f_{w,b}(x^n)\right) x_i^n$

$$\boxed{\begin{array}{l}\text{Cross entropy:}\\ l(f(x^n), \hat{y}^n) = -\left[\hat{y}^n ln f(x^n) + (1 - \hat{y}^n) ln\left(1 - f(x^n)\right)\right]\end{array}}$$

Limitation of Logistic Regression

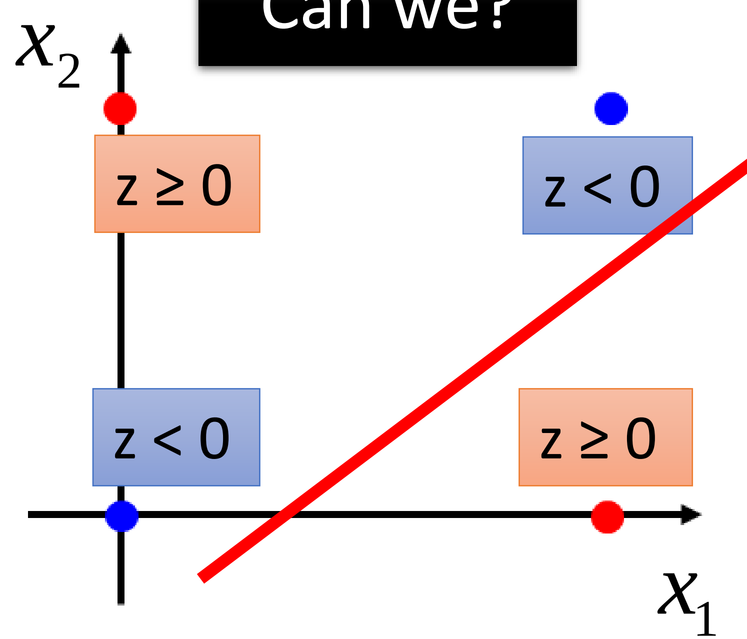
$$z = w_1 x_1 + w_2 x_2 + b$$



$$\begin{cases} Class1 & y \geq 0.5 \quad (z \geq 0) \\ Class2 & y < 0.5 \quad (z < 0) \end{cases}$$

Input Feature		Label
x_1	x_2	
0	0	Class 2
0	1	Class 1
1	0	Class 1
1	1	Class 2

Can we?



Limitation of Logistic Regression

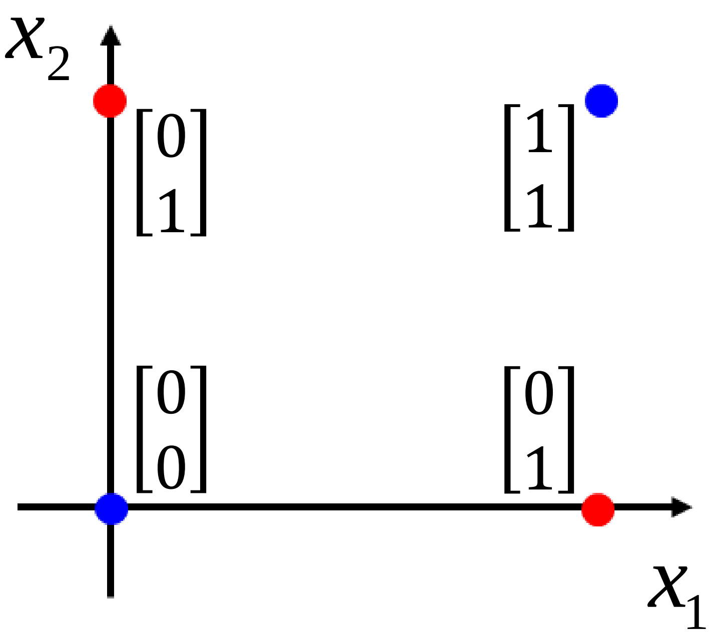
- ***Feature transformation***

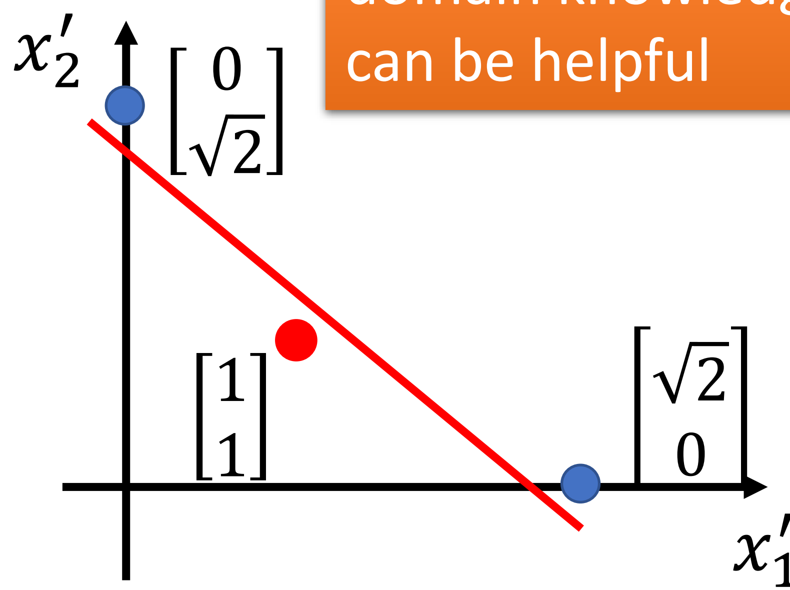
x_1': distance to $\begin{bmatrix}0\\0\end{bmatrix}$

x_2': distance to $\begin{bmatrix}1\\1\end{bmatrix}$

$$\begin{bmatrix}x_1\\x_2\end{bmatrix} \longrightarrow \begin{bmatrix}x_1'\\x_2'\end{bmatrix}$$

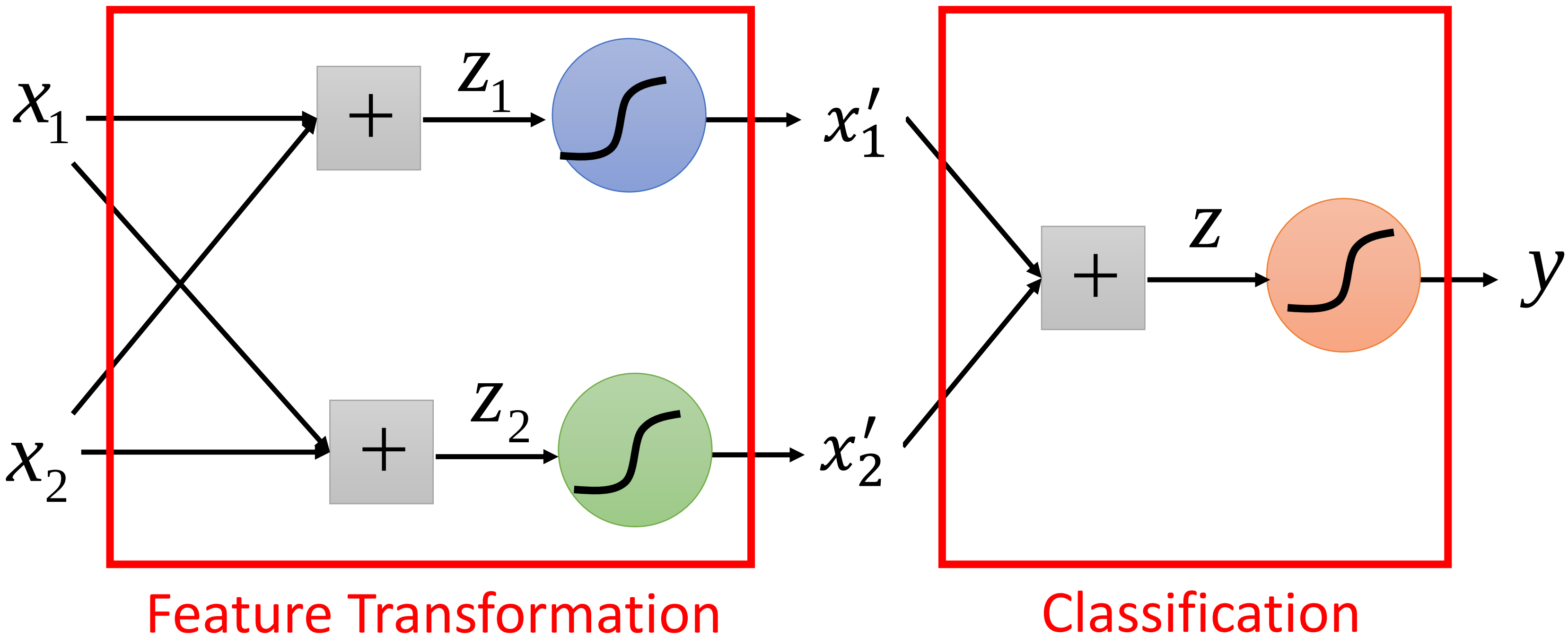
Not always easy ….. domain knowledge can be helpful





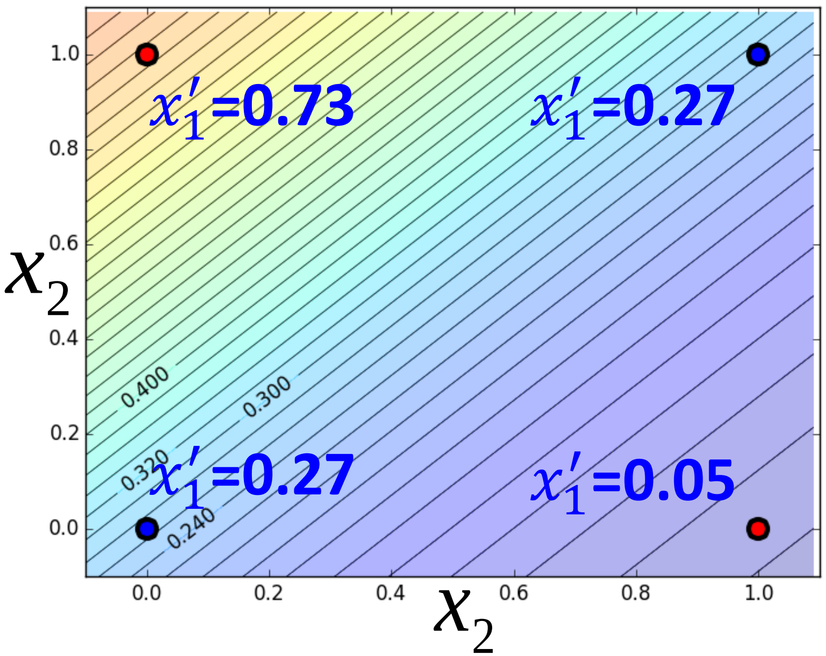
Limitation of Logistic Regression

- Cascading logistic regression models

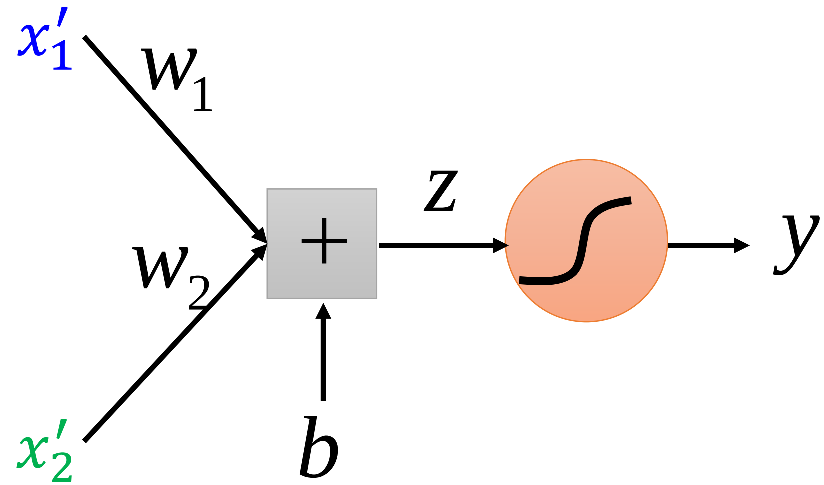


多个逻辑回归实际上
就是在做特征转换

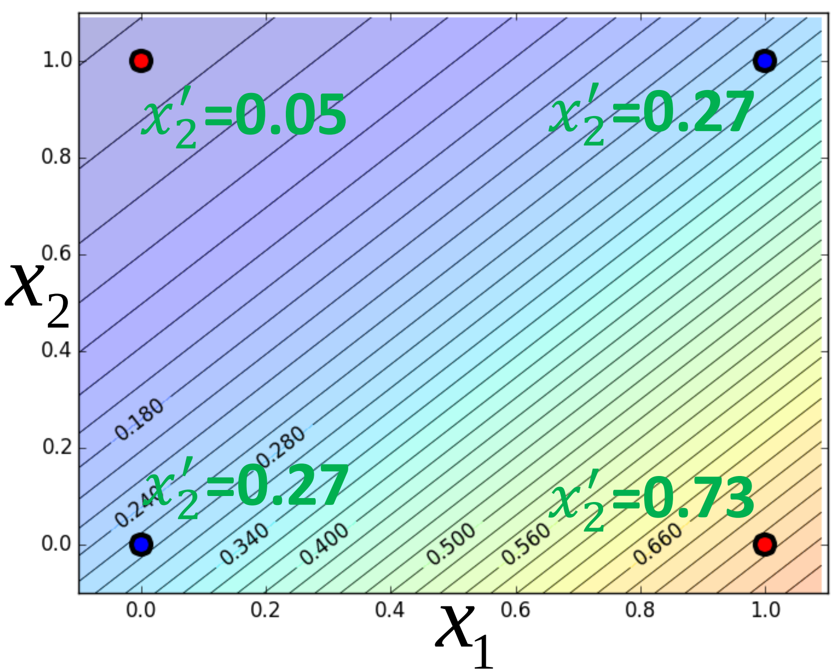
(ignore bias in this figure)



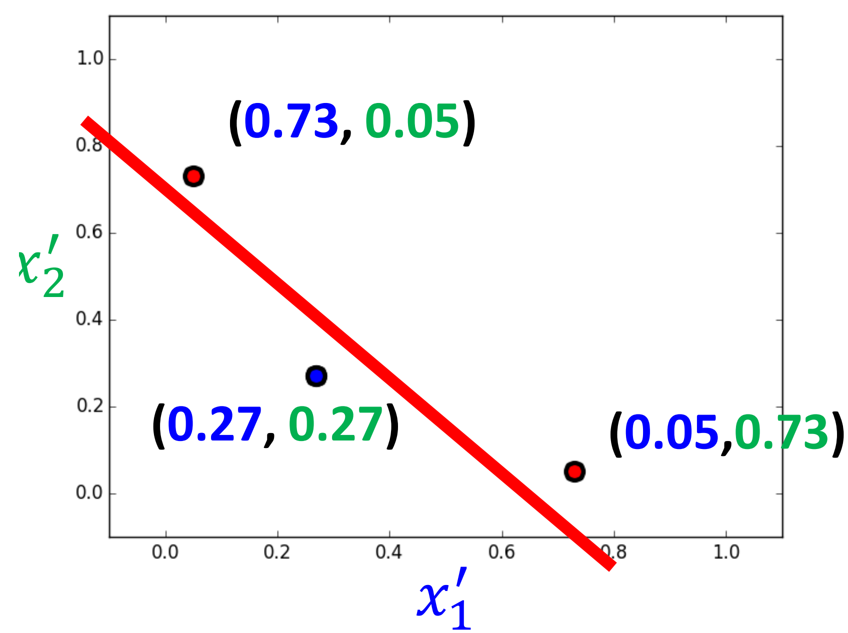
x_1'=0.73
x_1'=0.27
x_1'=0.27
x_1'=0.05
0.400
0.300
0.320
0.240
x_2
x_2



x_1'
w_1
w_2
x_2'
+
b
z
y



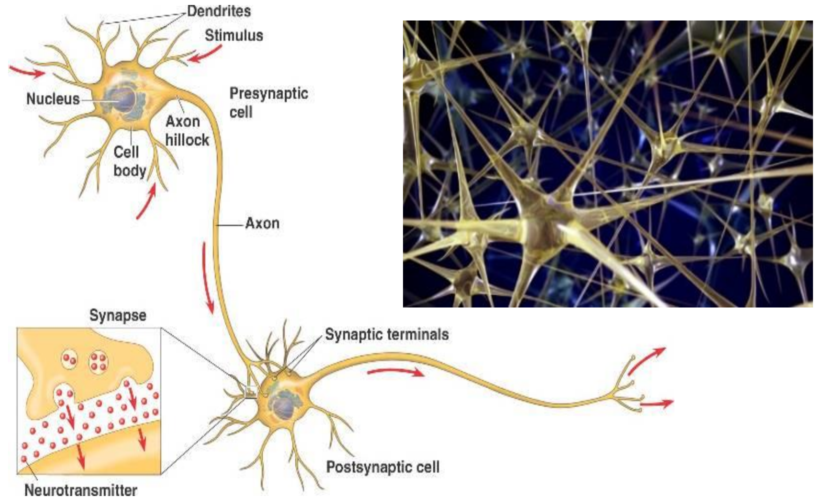
x_2'=0.05
x_2'=0.27
x_2'=0.27
x_2'=0.73
0.180
0.280
0.240
0.340
0.400
0.500
0.560
0.660
x_2
x_1

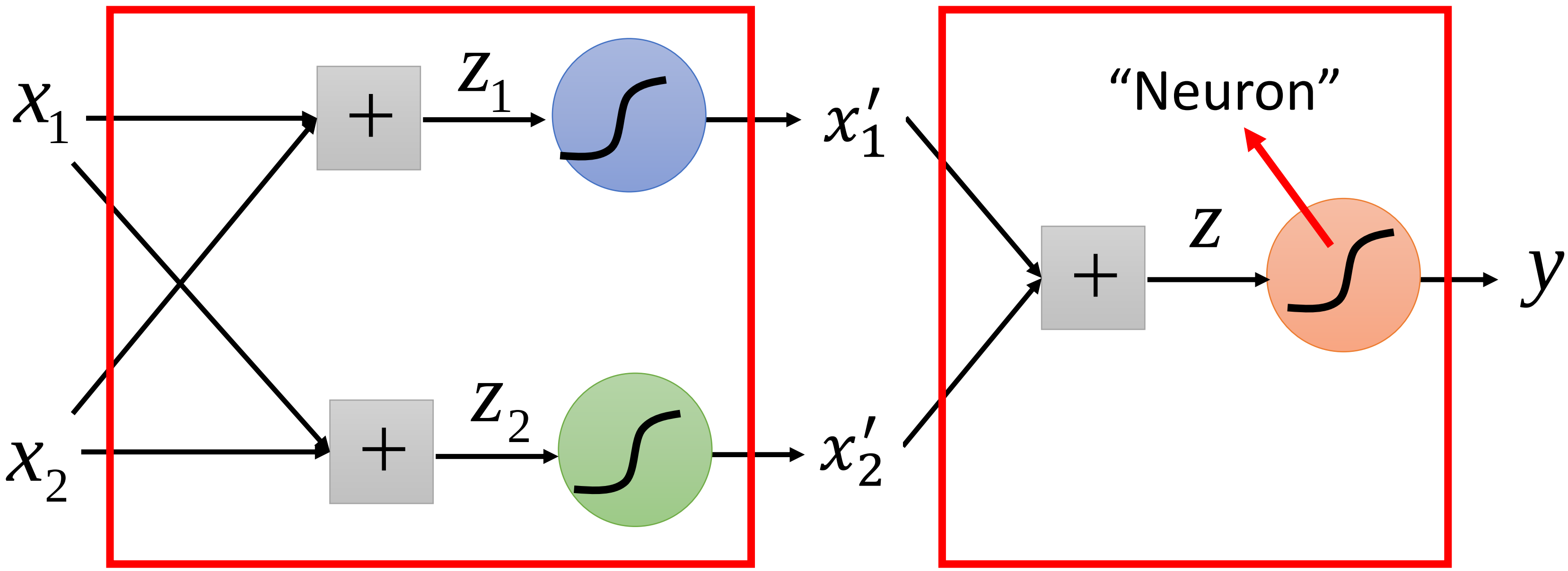


(0.73, 0.05)
(0.27, 0.27)
(0.05,0.73)
x_2'
x_1'

Deep Learning!

All the parameters of the logistic regressions are jointly learned.





Feature Transformation

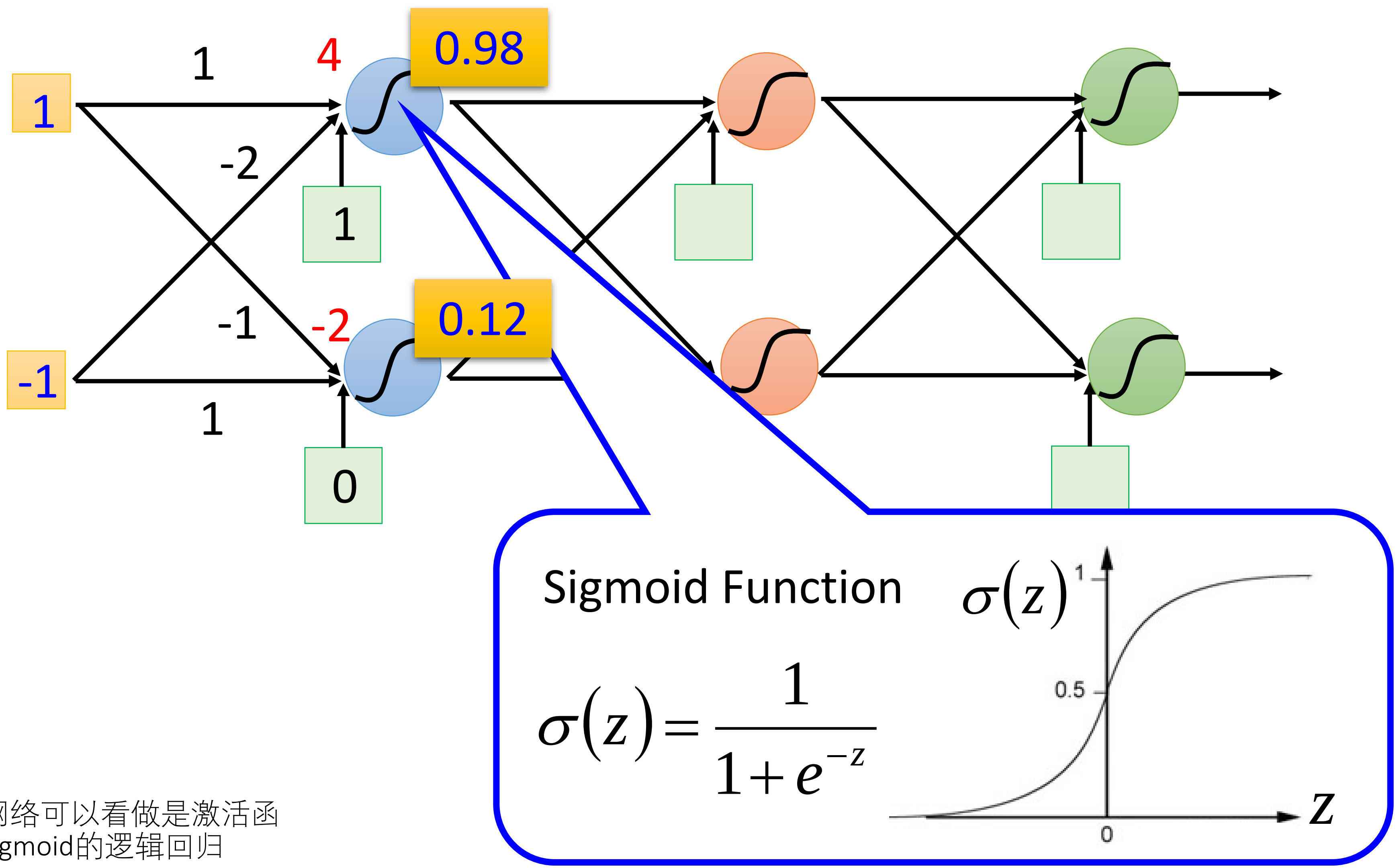
Classification

神经网络可以看做是多个逻辑回归连接的结果

Neural Network

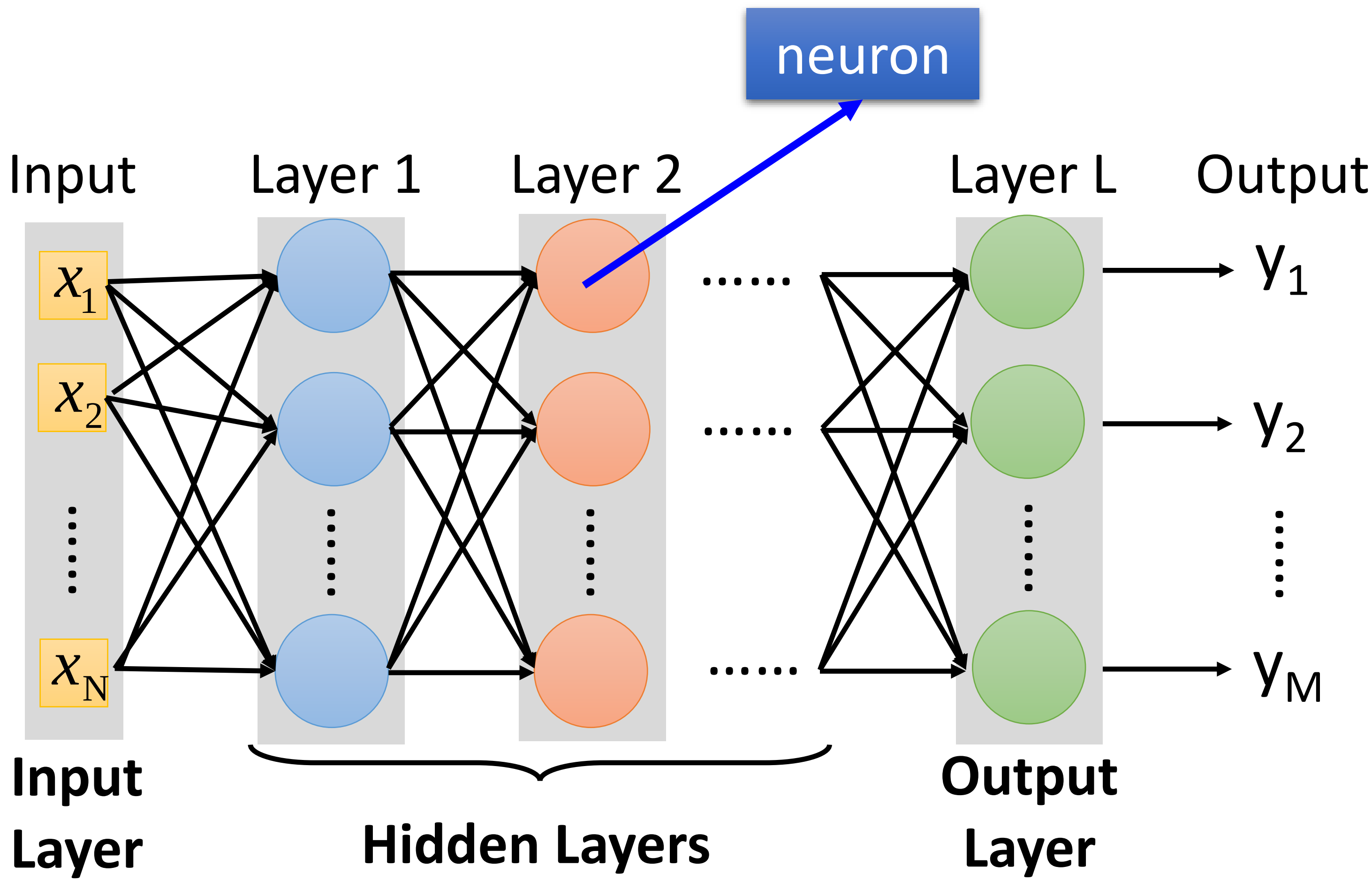
Deep Learning

Fully Connect Feedforward Network



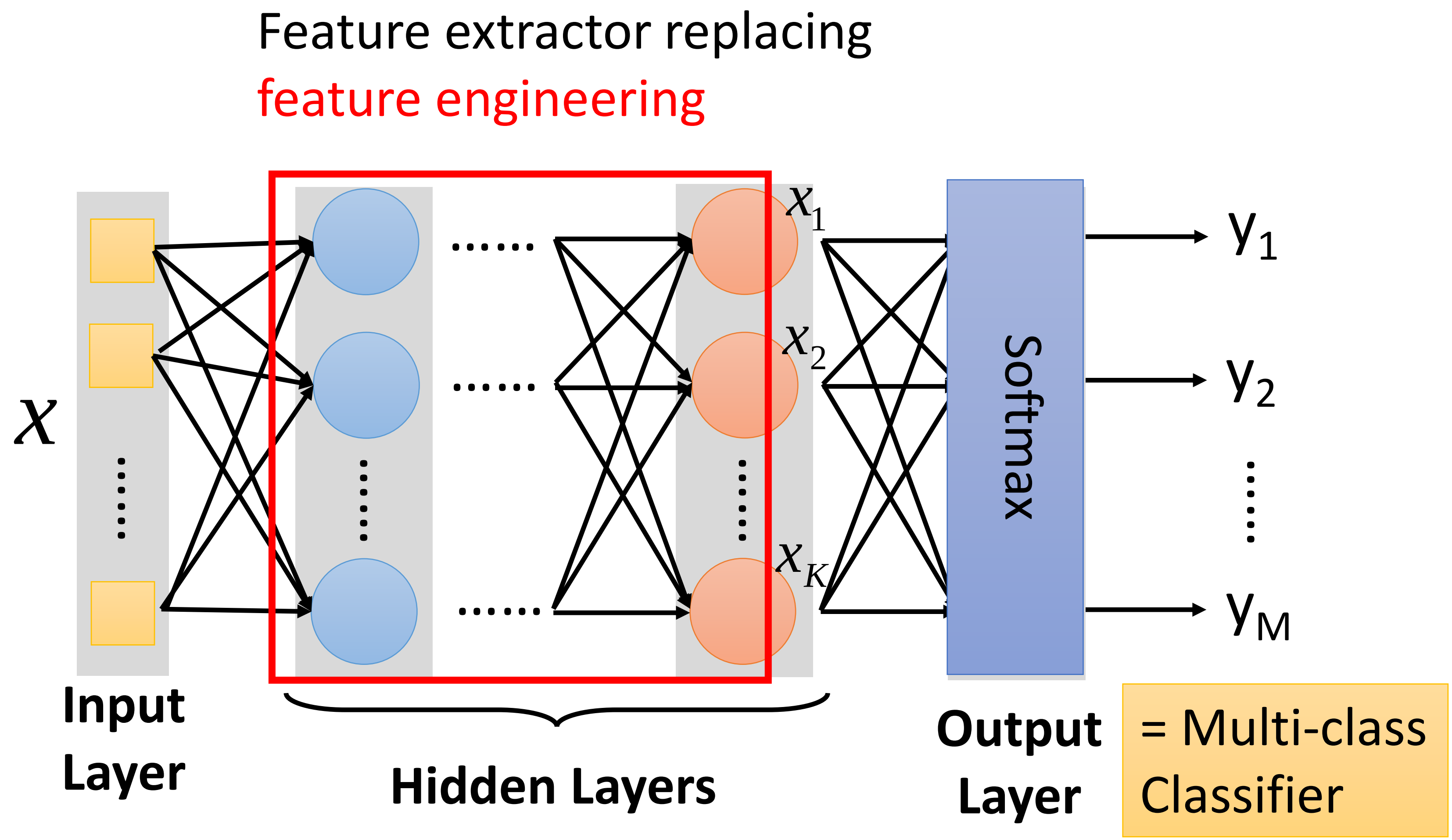
神经网络可以看做是激活函数为sigmoid的逻辑回归

Fully Connect Feedforward Network



神经网络的隐藏层本质
上就是特征提取器

Output Layer as **Multi-Class Classifier**

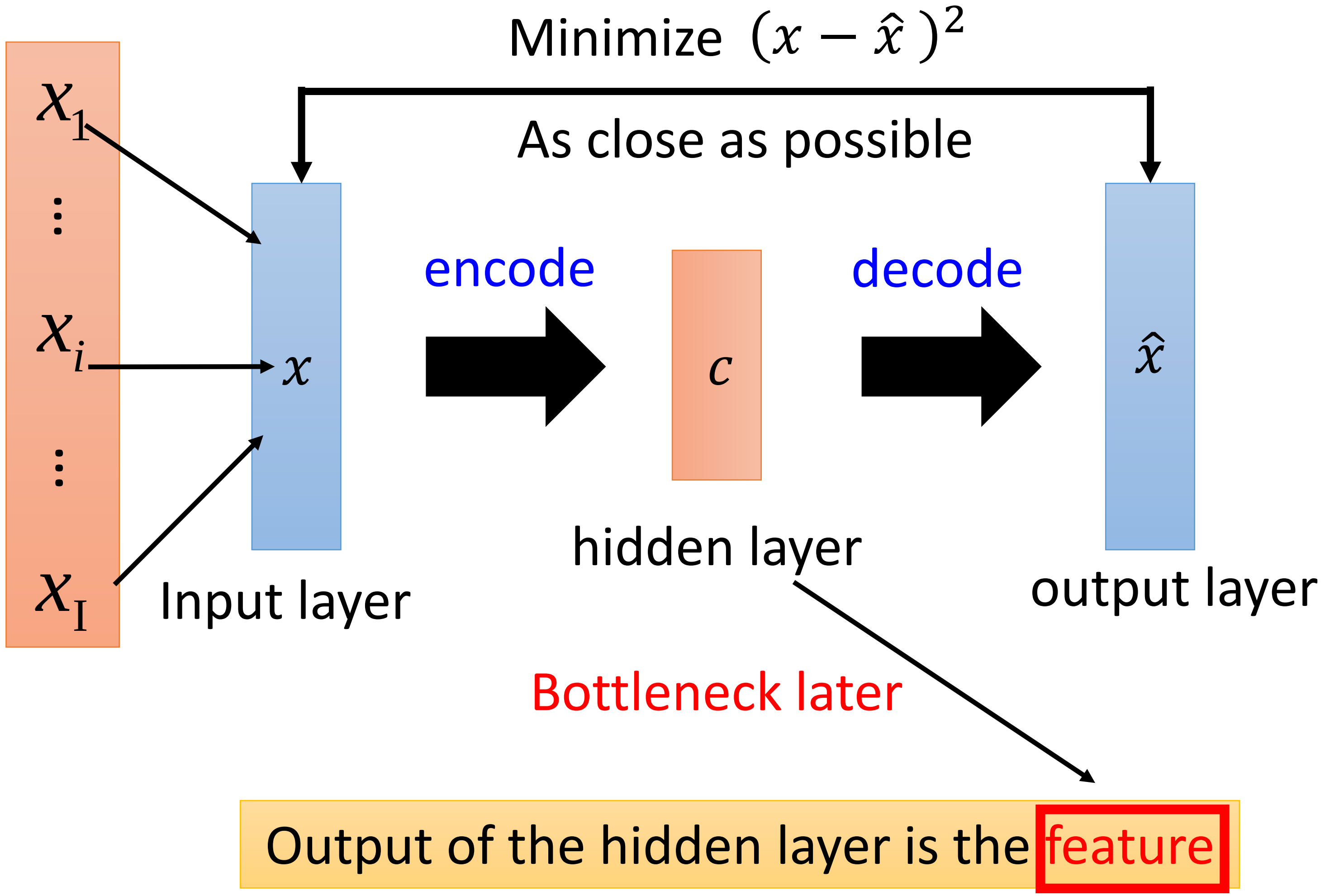


神经网络的隐藏层本质
上就是特征提取器

Auto-Encoder
for feature engineer

创新点： 尝试利用autoencoder实现特征工程
利用神经网络提取特征

Auto-encoder



Experiment Result Analysis

Lightgbm

model:

```python
import lightgbm as lgb
lgb_train = lgb.Dataset(X_train, Y_train)
lgb_val = lgb.Dataset(X_validation, Y_validation)

params = {
    'boosting_type': 'gbdt',
    'objective': 'binary',
    'metric': {'auc'},
    'num_leaves': 15,
    'max_depth': -1,
    'min_data_in_leaf': 64,
    'learning_rate': 0.1,
    'feature_fraction': 0.8,
    'bagging_fraction': 0.8,
    'bagging_freq': 1,
#     'lambda_l1': 1,
#     'lambda_l2': 0.001,  # 越小l2正则程度越高
#     'min_gain_to_split': 0.2,
    'verbose': -1,
    'is_unbalance': False,
    'num_boost_round': 200
}

lgbm = lgb.train(params=params, train_set=lgb_train, valid_sets=lgb_val)
```

result:

```
Validation set:
auc by sklearn: 0.8175
--------------------
Training set:
auc by sklearn: 0.8193
```

不加自编码的特征工程在模型上的结果

Random Forest

model:

```
from sklearn.ensemble import RandomForestClassifier

for i in range(5, 10):
    for j in range(10, 20):
        rfc = RandomForestClassifier(n_estimators=j, max_depth=i, random_state=0)
        rfc.fit(X_train, Y_train)
        auc_train_rfc = roc_auc_score(Y_train, rfc.predict_proba(X_train)[:,1])
        auc_val_rfc = roc_auc_score(Y_validation, rfc.predict_proba(X_validation)[:,1])
        print("estimators: %d depth: %d\t auc_train: %.4f\t auc_validation: %.4f" % \
              (j, i, auc_train_rfc, auc_val_rfc))
```

result:

```
estimators: 10 depth: 8  auc_train: 0.8195     auc_validation: 0.8177
estimators: 11 depth: 8  auc_train: 0.8195     auc_validation: 0.8178
estimators: 12 depth: 8  auc_train: 0.8195     auc_validation: 0.8178
estimators: 13 depth: 8  auc_train: 0.8195     auc_validation: 0.8178
estimators: 14 depth: 8  auc_train: 0.8195     auc_validation: 0.8178
estimators: 15 depth: 8  auc_train: 0.8195     auc_validation: 0.8178
estimators: 16 depth: 8  auc_train: 0.8194     auc_validation: 0.8178
estimators: 17 depth: 8  auc_train: 0.8194     auc_validation: 0.8178
estimators: 18 depth: 8  auc_train: 0.8194     auc_validation: 0.8178
estimators: 19 depth: 8  auc_train: 0.8195     auc_validation: 0.8178
```

不加自编码的特征工程在模型上的结果

Logistic Regression

model:

```
from sklearn.linear_model import LogisticRegression
LR=LogisticRegression(random_state=0,
                      solver="sag",
                      penalty="l2",
                      class_weight="balanced",
                      C=1.0,
                      max_iter=500)

LR.fit(X_train, Y_train)
```

```
LogisticRegression(C=1.0, class_weight='balanced', dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=500, multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='sag', tol=0.0001, verbose=0,
                   warm_start=False)
```

result:

```
Validation set:
auc by sklearn: 0.8177
--------------------
Training set:
auc by sklearn: 0.8182
```

不加自编码的特征工程在模型上的结果

Make Score Cards

Take logistic regression as an example

制作评分卡

以Logic Regression模型为例，制作评分卡，更好地量化用户信用

```
intercept=LR.intercept_
coef=LR.coef_
coe=coef[0].tolist()
coe_df=pd.DataFrame({'feature':IV_info,'coe':coe})

import math
B=20/math.log(2)
A=600+B*math.log(1/20)
#基础分
score=round(A-B*intercept[0],0)
```

```
featurelist = []
woelist = []
cutlist = []
for k,v in woe_dict.items():
    if k in IV_info:
        for n in range(0,len(v)):
            featurelist.append(k)
            woelist.append(v[n])
            cutlist.append(cut_dict[k][n])
scoreboard = pd.DataFrame({'feature':featurelist,'woe':woelist,'cut':cutlist},
                          columns=['feature','cut','woe'])
score_df=pd.merge(scoreboard,coe_df)
score_df['score']=round(-B*score_df['woe']*score_df['coe'],0)
score_df.drop('coe',axis=1,inplace=True)
score_df
```

Result

	feature	cut	woe	score
0	Revol	0.0	0.119231	2.0
1	Revol	1.0	-2.226411	-38.0
2	age	21.0	-0.487183	-11.0
3	age	39.0	-0.252557	-6.0
4	age	48.0	-0.078292	-2.0
5	age	56.0	0.430123	10.0
6	age	65.0	1.054168	25.0
7	Num90late	0.0	0.375825	5.0
8	Num90late	1.0	-1.971860	-28.0
9	Num90late	2.0	-2.646308	-37.0
10	Num90late	3.0	-2.961503	-42.0
11	Num90late	4.0	-3.358818	-47.0
12	Num90late	5.0	-3.197806	-45.0
13	Num90late	6.0	-3.055631	-43.0
14	Num90late	7.0	-4.138243	-58.0
15	Num90late	8.0	-3.566457	-50.0
16	Num90late	9.0	-3.679786	-52.0
17	Num90late	10.0	-2.817220	-40.0
18	Num60-89late	0.0	0.274309	3.0
19	Num60-89late	1.0	-1.850365	-20.0
20	Num60-89late	2.0	-2.657322	-29.0
21	Num60-89late	3.0	-2.915869	-32.0
22	Num60-89late	4.0	-3.135674	-35.0
23	Num60-89late	5.0	-3.129739	-35.0

Performance

with common feature engineering

model: lightgbm < LR < random forest

result: 81.75 < 81.77 < 81.78

the performance is nearly the same, but it’s not the end!

we’ll try WOE+Auto-Encoder to make new feature extractor!

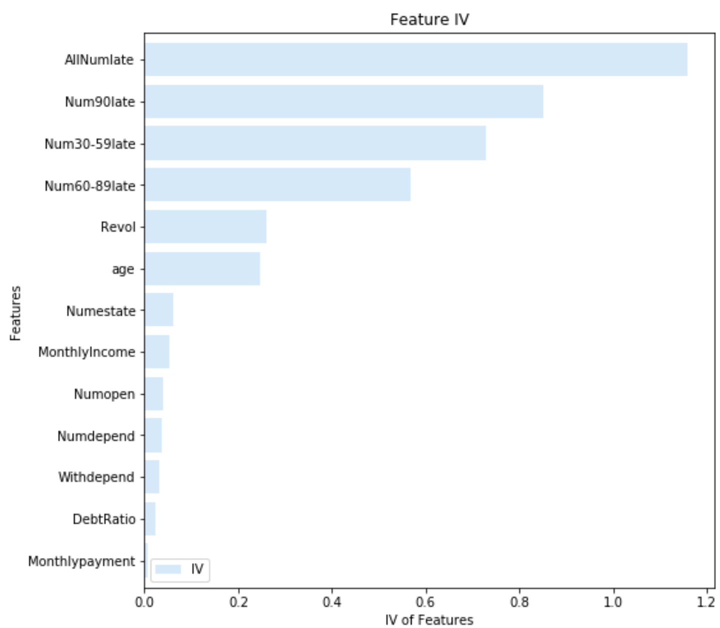
不加自编码的特征工程在模型上的结果

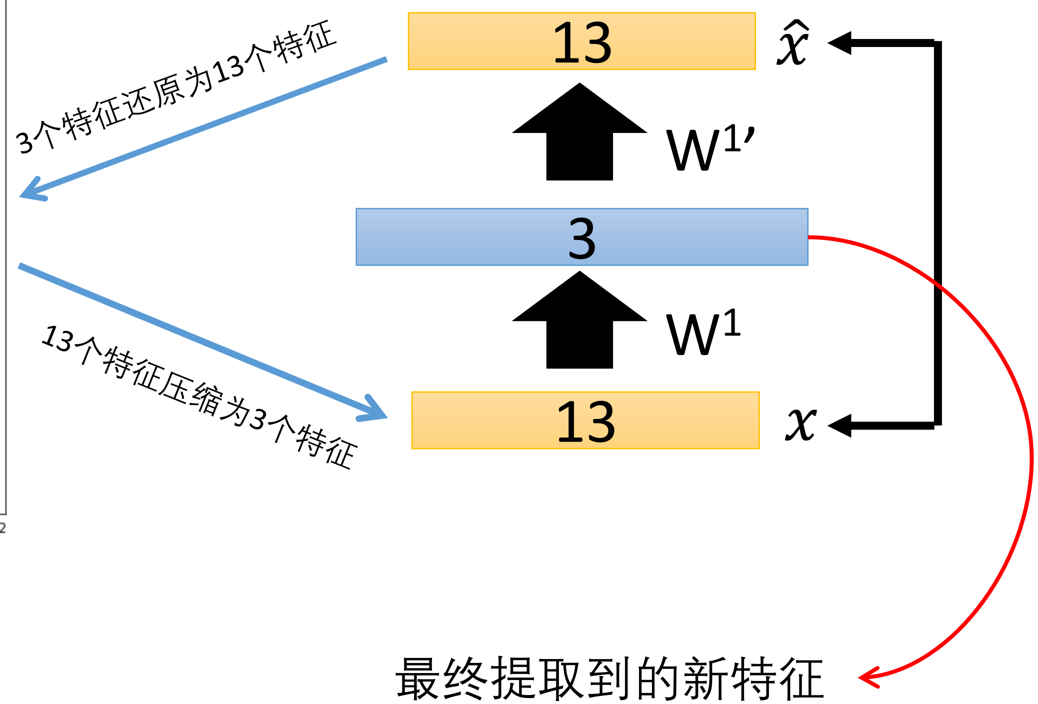
New Idea

we'll try WOE+Auto-Encoder to make new feature extractor!

初步特征工程+WOE编码后得到的13个特征

神经网络自编码进一步提取为3个特征





创新点：WOE编码+Autoencoder=New Feature

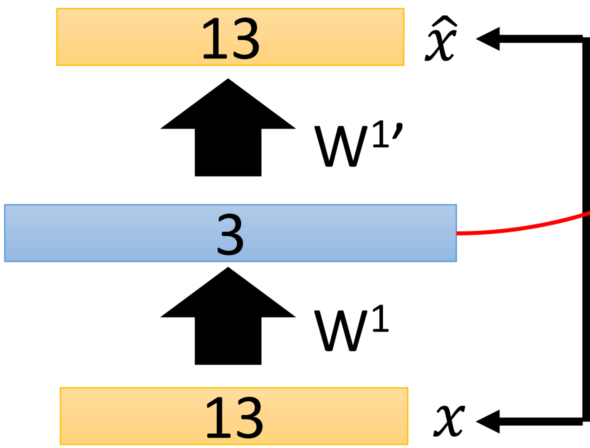
New Idea

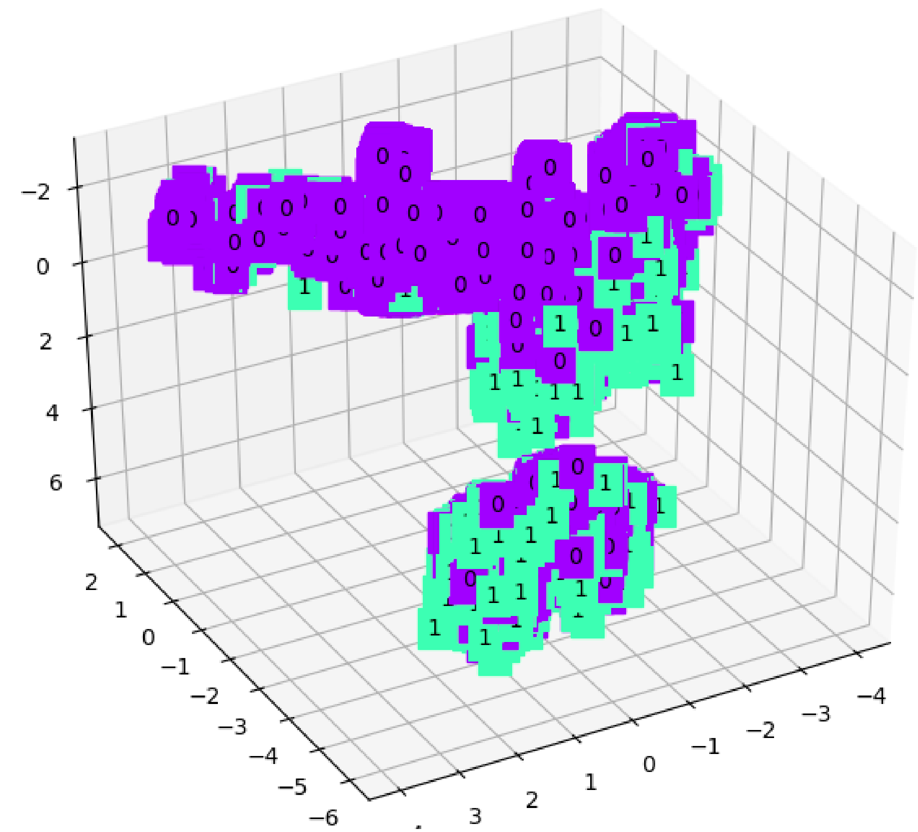
we'll try WOE+Auto-Encoder to make new feature extractor!

WOE编码+Auto-encoder
13-dimension -> 3-dimension

三维特征空间可视化

最终提取到的新特征





创新点：WOE编码+Autoencoder=New Feature

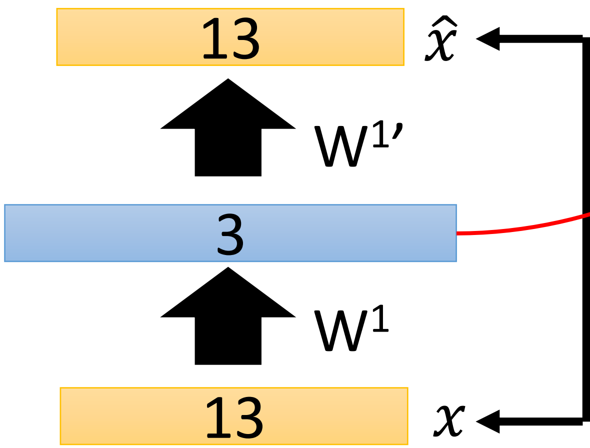
New Idea

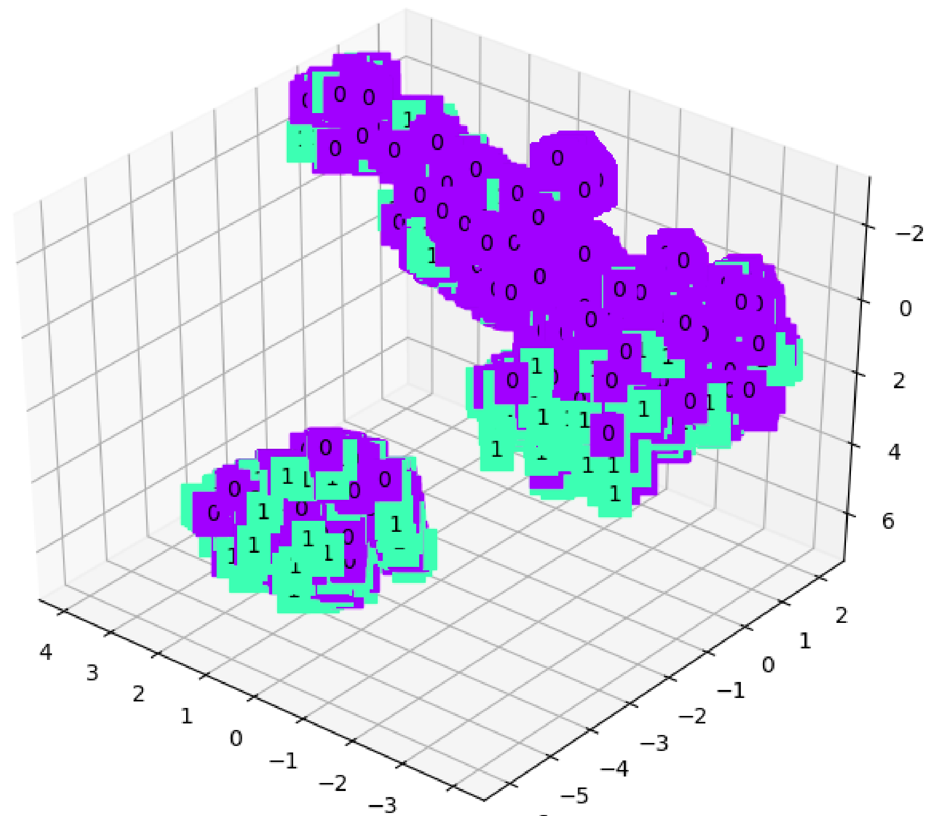
we'll try WOE+Auto-Encoder to make new feature extractor!

WOE编码+Auto-encoder
13-dimension -> 3-dimension

三维特征空间可视化

最终提取到的新特征





创新点：WOE编码+Autoencoder=New Feature

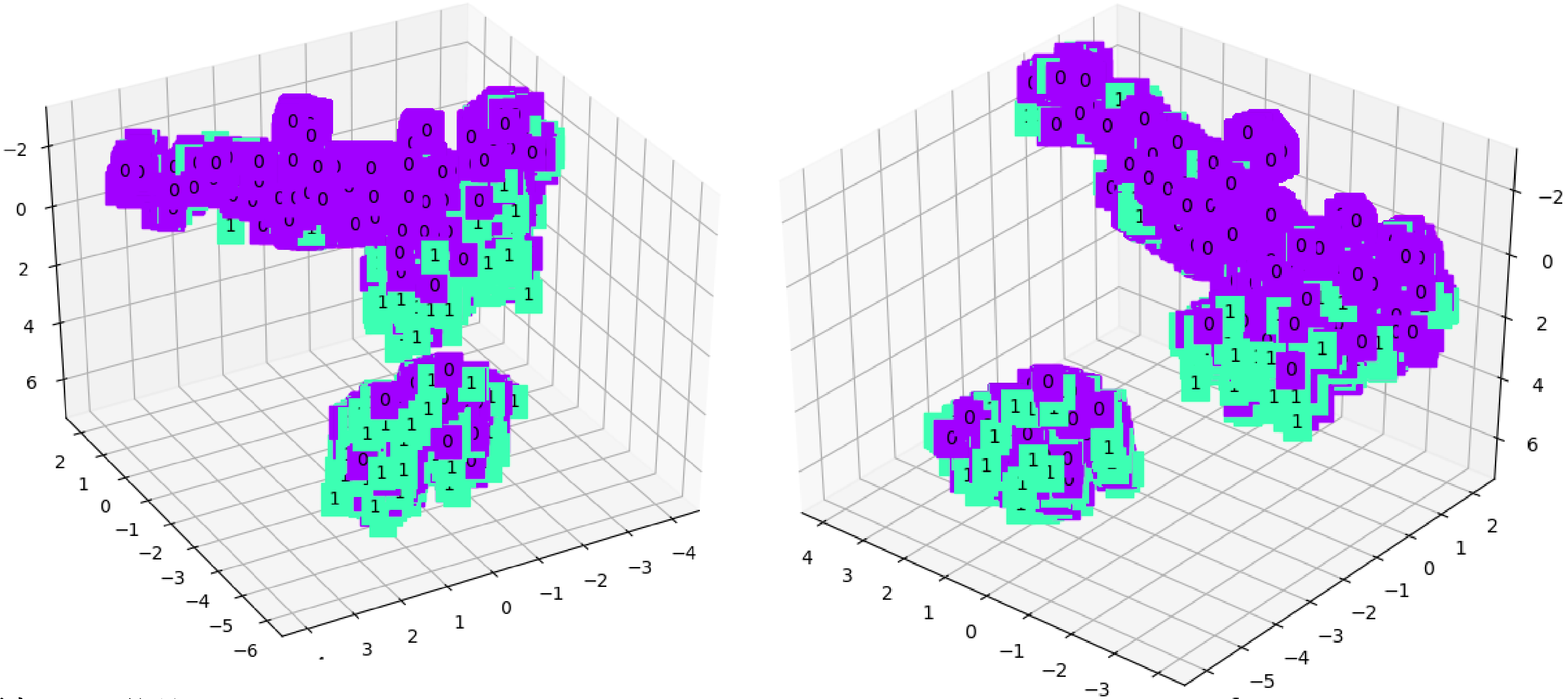
New Idea

we'll try WOE+Auto-Encoder to make new feature extractor!

WOE编码+Auto-encoder
13-dimension -> 3-dimension

→ 三维特征空间可视化

分类效果较为**明显**!



创新点：WOE编码+Autoencoder=New Feature

New Idea

we'll try WOE+Auto-Encoder to make new feature extractor!

WOE编码+Auto-encoder
13-dimension -> 3-dimension

三维特征空间可视化

implement with PyTorch

Auto-encoder神经网络实现示意图:

```
class AutoEncoder(nn.Module):
    def __init__(self):
        super(AutoEncoder, self).__init__()

        self.encoder = nn.Sequential(
            nn.Linear(13, 64),
            nn.Tanh(),
            nn.Linear(64, 32),
            nn.Tanh(),
            nn.Linear(32, 3)
        )

        self.decoder = nn.Sequential(
            nn.Linear(3, 32),
            nn.Tanh(),
            nn.Linear(32, 64),
            nn.Tanh(),
            nn.Linear(64, 13)
        )

    def forward(self, x):
        encode = self.encoder(x)
        decode = self.decoder(encode)
        return encode, decode
```

创新点：WOE编码+Autoencoder=New Feature

Performance with New Idea

相同参数的模型 + 不同的特征工程	old		new	
Lightgbm:	Validation set: auc by sklearn: 0.8175 -------------------- Training set: auc by sklearn: 0.8193	→	Validation set: auc by sklearn: 0.8269 -------------------- Training set: auc by sklearn: 0.8517	1%~3% ↑
Random Forest:	estimators: 19 depth: 9 auc_train: 0.8195 auc_validation: 0.8177	→	estimators: 19 depth: 9 auc_train: 0.8420 auc_validation: 0.8227	1%~3% ↑
Logistic Regression:	Validation set: auc by sklearn: 0.8177 -------------------- Training set: auc by sklearn: 0.8182	→	Validation set: auc by sklearn: 0.7752 -------------------- Training set: auc by sklearn: 0.7730	3%~4% ↓

WOE + Auto-Encoder在
相同模型上的结果对比

Performance with New Idea

相同参数的模型 + 不同的特征工程

Lightgbm: 1%~3% ↑

Random Forest: 1%~3% ↑

Logistic Regression: 3%~4% ↓

WOE编码+Auto-encoder的特征工程对树模型的性能改进贡献了一定的作用!

WOE + Auto-Encoder在相同模型上的结果对比

Performance with New Idea

DNN Classifier

(simple implement)

```
class CreditClassifier(nn.Module):
    def __init__(self):
        super(CreditClassifier, self).__init__()
        self.layer1 = nn.Linear(3, 32)
        self.layer2 = nn.Linear(32, 16)
        self.layer3 = nn.Linear(16, 1)

    def forward(self, x):
        x = torch.relu(self.layer1(x))
        x = torch.relu(self.layer2(x))
        x = torch.sigmoid(self.layer3(x))
        return x
```

神经网络分类器

Mean Square Error

Adam (lr=0.01)

```
classifier = CreditClassifier()

optim = torch.optim.Adam(classifier.parameters(), lr=0.01)

mse_loss = nn.MSELoss()

EPOCH = 100

for epoch in range(EPOCH):
    for batch, (x, y) in enumerate(train_loader):
        y_hat = classifier(x)
        loss = mse_loss(y_hat, y.type(torch.FloatTensor).view(-1,1))
        optim.zero_grad()
        loss.backward()
        optim.step()

        if batch % 100 == 0:
            print('epoch %d batch %d loss: %.4f' % (epoch, batch, loss.data.item()))
```

WOE + Auto-Encoder在DNN上的结果

Performance with New Idea

DNN Classifier

(simple implement)

```
class CreditClassifier(nn.Module):
    def __init__(self):
        super(CreditClassifier, self).__init__()
        self.layer1 = nn.Linear(3, 32)
        self.layer2 = nn.Linear(32, 16)
        self.layer3 = nn.Linear(16, 1)

    def forward(self, x):
        x = torch.relu(self.layer1(x))
        x = torch.relu(self.layer2(x))
        x = torch.sigmoid(self.layer3(x))
        return x
```

神经网络分类器

Mean Square Error

result:

```
auc_train: 0.8307
auc_validation: 0.8242
```

Adam (lr=0.01)

WOE + Auto-Encoder在DNN上的结果

Performance Compare

WOE + Auto-encoder

DNN classifier:

```
auc_train: 0.8307
auc_validation: 0.8242
```

Lightgbm:

```
Validation set:
auc by sklearn: 0.8269
--------------------
Training set:
auc by sklearn: 0.8517
```

1%~3% ↑

Random Forest:

```
estimators: 19 depth: 9
auc_train: 0.8420
auc_validation: 0.8227
```

1%~3% ↑

Logistic Regression:

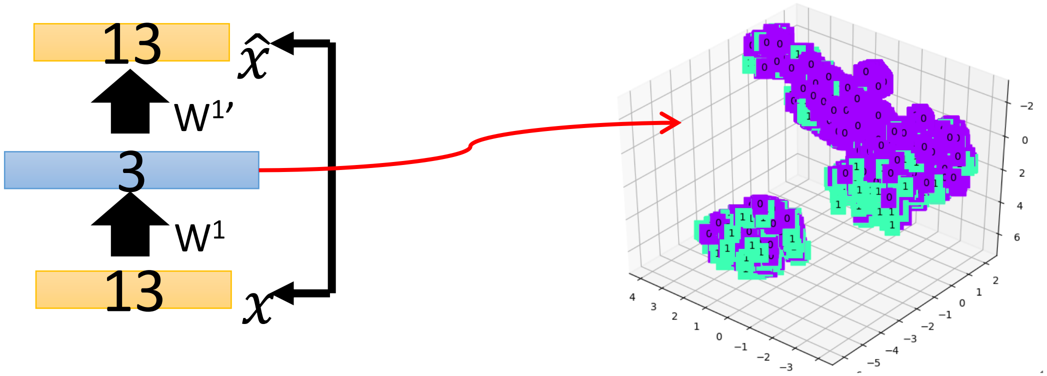
```
Validation set:
auc by sklearn: 0.7752
--------------------
Training set:
auc by sklearn: 0.7730
```

3%~4% ↓

WOE + Auto-Encoder在
不同模型上的结果对比

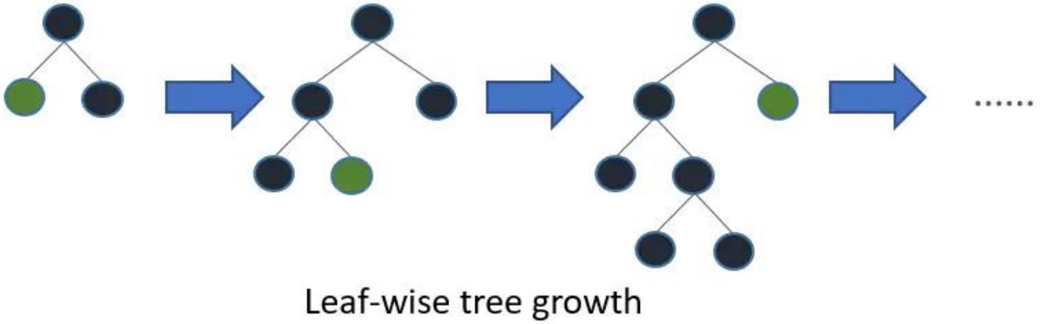
Final Choice

WOE + Auto-encoder



+

LightGBM



Thanks