1.1 -1.

$$E(\hat{\lambda}) = E\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n}\sum_{i=1}^{n} E(X_i) = \frac{1}{n}\sum_{i=1}^{n}\lambda$$

(if it is unbiased)

$$= \frac{1}{n} n\lambda = \lambda \Rightarrow E(\hat{\lambda}) - \lambda = 0$$ proof

-2 posterior distribution $P(\lambda|D)$

$$P(\lambda|D) = \frac{P(D|\lambda)P(\lambda)}{P(D)}$$ (Possion distributed provide $\mathcal{L}(D|\lambda) = \prod_{i=1}^{n} \frac{e^{-\lambda}\lambda^{D_i}}{D_i!}$)

$$= \frac{\left(\prod_{i=1}^{n} \frac{e^{-\lambda}\lambda^{D_i}}{D_i!}\right) \cdot \left(\frac{\beta^{\alpha}}{(\alpha-1)!} \cdot \lambda^{\alpha-1} e^{-\beta\lambda}\right)}{P(D)}$$

$$= \frac{\frac{e^{-n\lambda}\lambda^{\sum D_i}}{\underbrace{\prod_{i=1}^{n}(D_i)!}_{\to C}} \cdot \frac{\beta^{\alpha}}{(\alpha-1)!} \cdot \lambda^{\alpha-1} e^{-\beta\lambda}}{\underbrace{P(D)}_{\to C}}$$

$$\propto \lambda^{\sum D_i + \alpha - 1} e^{-(n+\beta)\lambda}$$

-3 from 2 we know

$$P(\lambda|D) \propto \lambda^{\sum D_i + \alpha - 1} e^{-(n+\beta)\lambda}$$

$$\hat{\lambda}_{MAP} = \arg\max_{\lambda} P(\lambda|D)$$

$$= \arg\max_{\lambda} \lambda^{\sum D_i + \alpha - 1} e^{-(n+\beta)\lambda}$$

$$= \arg\max_{\lambda} \left(\log\left(\lambda^{\sum D_i + \alpha - 1} e^{-(n+\beta)\lambda}\right)\right)$$

$$= \arg\max_{\lambda} \left(\left(\sum_{i=0}^{n} D_i + \alpha - 1\right) \ln \lambda - (n+\beta)\cdot\lambda\right) \quad \text{(use ln here)}$$

$$\frac{\partial\left(\left(\sum_{i=0}^{n} D_i + \alpha - 1\right) \ln\lambda - (n+\beta)\lambda\right)}{\partial \lambda}$$

n

$$= \frac{\sum_{i=0}^{n} D_i + \alpha - 1}{\lambda} - (n+\beta) = 0$$

$$\hat{\lambda}_{MAP} = \frac{\sum_{i=0}^{n} D_i + \alpha - 1}{n+\beta}$$

2.1-1

There is two kinds of label and '+' appears 12 times while '-' appears 9 times.

Therefore,

$$H(Y) = -\sum_{i=0}^{\text{labelsize}} \frac{|C_i|}{|D|} \log_2 \frac{|C_i|}{|D|}$$

$$= -\frac{9}{21}\log_2\frac{9}{21} - \frac{12}{21}\log_2\frac{12}{21} = 0.9852$$

-2.

$$IG(X_1) = H(Y) - H(Y|X_1)$$

$$= H(Y) - \left(-\sum_{i=0}^{\text{featuresize}} P_i \left(\sum_{j}^{\text{labelsize}} P_j \log P_j\right)\right)$$

$$= H(Y) + \frac{8}{21}\cdot\left(\frac{7}{8}\log\frac{7}{8} + \frac{1}{8}\log\frac{1}{8}\right)$$

$$+ \frac{13}{21}\cdot\left(\frac{5}{13}\log\frac{5}{13} + \frac{8}{13}\log\frac{8}{13}\right)$$

$$= 0.9852 - 0.2071 - 0.5951.$$

$$= 0.183$$

$$IG(X_2) = H(Y) - H(Y|X_2)$$

$$= H(Y) - \left(-\sum_{i=0}^{fs} P_i \left(\sum_{j=0}^{ls} P_j \log P_j\right)\right)$$

$$= H(Y) + \frac{10}{21}\left(\frac{7}{10}\log\frac{7}{10} + \frac{3}{10}\log\frac{3}{10}\right)$$

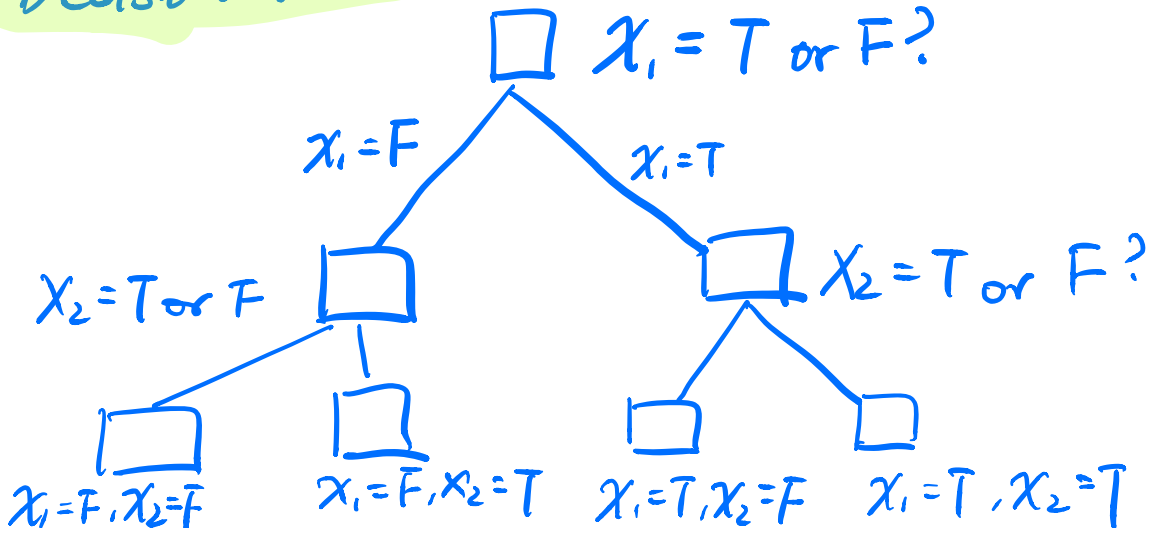
$$+\frac{11}{21}\left(\frac{3}{11}\log\frac{3}{11}+\frac{8}{11}\log\frac{8}{11}\right)$$

$$= 0.9852 - 0.4197 - 0.5207$$

$$= 0.0448$$

-3. Since $IG(X_1) > IG(X_2)$ the first layer of decision tree should be X_1. And since there is only X_2 feature left we use it as second layer feature.

Decision tree:



Algorith Details:

1. Create root
2. Rank Entropy gain (X_1, X_2), $cur = max((X_1, X_2))$
3. if entropy gain == 0, return single node
   else for type in feature cur, construct sub tree by ste 1.
4. End if all labels in cur layel is same

---

3. Perceptron

3-1

learning rate	large	small
pros	fast convergence avoid local minimum	convergence Guarantee
cons	loss number will jump higher and higher if too large	slow and go to local minimum

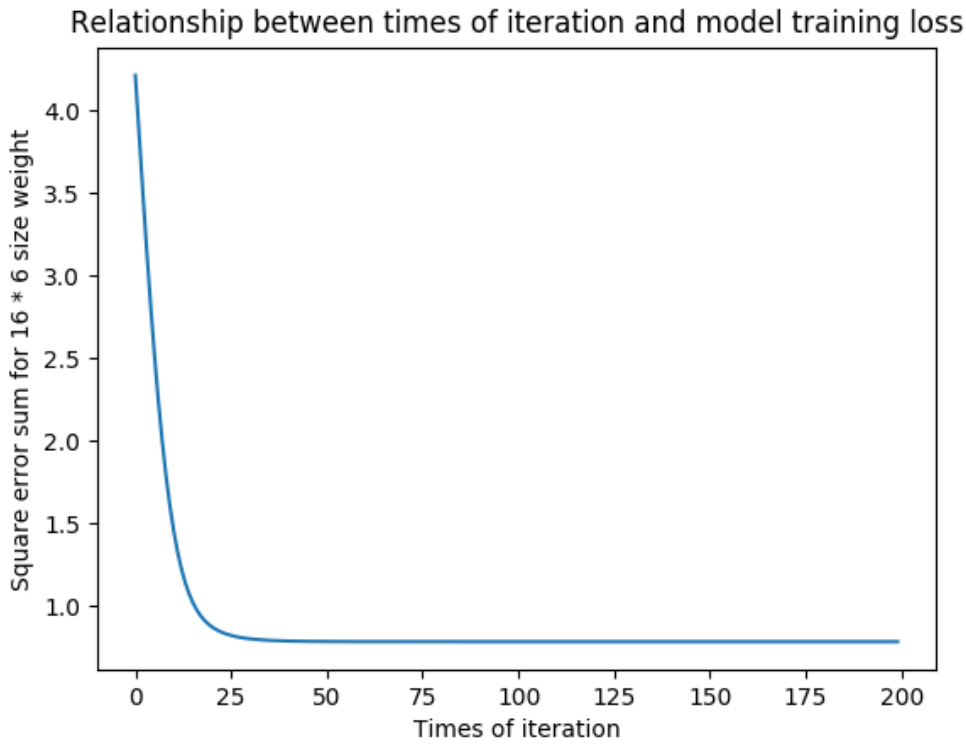
-2. Yes, the result is Converge.

Because with a fixed learning rate and the Perceptron training model, the weights of model change slower and slower and stop change after a fiexd number of iteration of trcing.

A peer of Thetas value in relation to iteration times for feature 6:

```
iterTimes: 179 Thetas: [-9.03743880e-03 -5.50611085e-04 -1
.29633885e-04  2.72762810e-06
 -2.49667511e-04 -1.41531229e-04]
iterTimes: 180 Thetas: [-9.03743880e-03 -5.50611086e-04 -1
.29633886e-04  2.72762709e-06
 -2.49667511e-04 -1.41531229e-04]
iterTimes: 181 Thetas: [-9.03743881e-03 -5.50611087e-04 -1
.29633886e-04  2.72762622e-06
 -2.49667512e-04 -1.41531230e-04]
iterTimes: 182 Thetas: [-9.03743881e-03 -5.50611088e-04 -1
.29633887e-04  2.72762546e-06
 -2.49667512e-04 -1.41531230e-04]
iterTimes: 183 Thetas: [-9.03743881e-03 -5.50611089e-04 -1
.29633887e-04  2.72762480e-06
 -2.49667512e-04 -1.41531231e-04]
iterTimes: 184 Thetas: [-9.03743881e-03 -5.50611090e-04 -1
.29633887e-04  2.72762423e-06
 -2.49667513e-04 -1.41531231e-04]
iterTimes: 185 Thetas: [-9.03743881e-03 -5.50611091e-04 -1
.29633888e-04  2.72762373e-06
 -2.49667513e-04 -1.41531231e-04]
iterTimes: 186 Thetas: [-9.03743881e-03 -5.50611092e-04 -1
.29633888e-04  2.72762330e-06
 -2.49667513e-04 -1.41531232e-04]
iterTimes: 187 Thetas: [-9.03743881e-03 -5.50611092e-04 -1
.29633888e-04  2.72762292e-06
 -2.49667513e-04 -1.41531232e-04]
iterTimes: 188 Thetas: [-9.03743881e-03 -5.50611093e-04 -1
.29633888e-04  2.72762259e-06
 -2.49667513e-04 -1.41531232e-04]
iterTimes: 189 Thetas: [-9.03743881e-03 -5.50611093e-04 -1
.29633888e-04  2.72762231e-06
 -2.49667513e-04 -1.41531232e-04]
```

As you can see, after 180 times of iteration, the weights[0] never change, and the other feature weights are also go to converge, when Iter ups to 1000, the Thetas never charge, which means it is converge



This is the relationship between times of iteration in trainiy process and the loss, as iteration times increase, loss almost stay in a constant value, which mean converge

-3 I tried four kinds of model and 2 kinds of feature in this prediction task.

Model: ① perceptron model update by square dist error of hypo and label, and use (16 x feature/size thetas tabel and use argmax to validate correctness.

② perceptron update by L1-distance and use 5 as a threthold to do validation

③ peceptron use 0-1 error and use 5 as a threthod to do validation

④ peceptron use (16 x fsize) Thetas and 0-1 error to update and 5 threthod to validte.

Label: feature 6: drink number of half-pint

feature 7: selector field.

The first model of (16 x fsize), sqare dist and argmax works best but is a little slow than othe easy theta model.

The accuracy for "drink number" judged by threthold 5 is 74.1% on averge with 50 times random validate (90% train, 10% test) while the accuray for

"selector field" is 57.8% for 30 times random validation as well

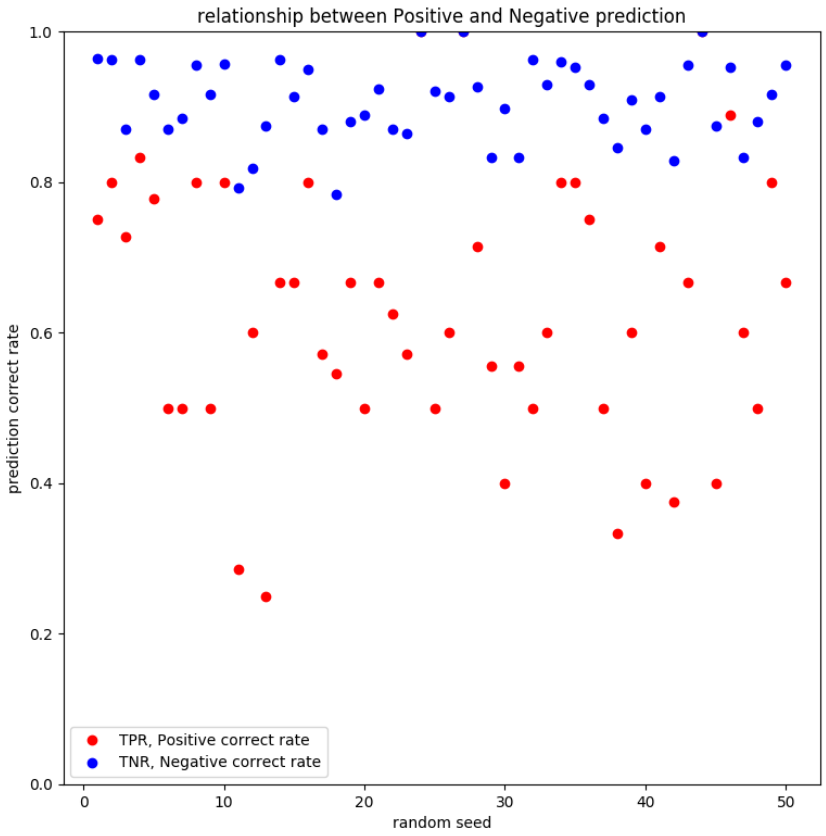
Since accuracy for feature 6 is higher, I use it as my confidence metric base to show the inside view more clearly.

confidenceMetric

random test seed	total test	truePositive	trueNegative	falsePositive	falseNegative	TPR	TNR
1	35	3	27	1	4	0.75	0.964
2	35	4	25	1	5	0.8	0.962
3	35	8	20	3	4	0.727	0.87
4	35	5	25	1	4	0.833	0.962
5	35	7	22	2	4	0.778	0.917
6	35	3	20	3	9	0.5	0.87
7	35	3	23	3	6	0.5	0.885
8	35	4	21	1	9	0.8	0.955
9	35	2	22	2	9	0.5	0.917
10	35	4	22	1	8	0.8	0.957

p.s. positive is "drink pint" >5

As you can see, TNR is larger than TPR generally, so our model has a higher accuracy at predicting negative (<0.5) result



this is relationship between classification and prediction accuracy. the blue one means negative (drink ≤ 5). and red one is positive (drink > 5). obviously, we have more confidence with negative predict.

-4. If taking 7th feature as label, a new synthetic feature will increase accuracy from 46.6% to 57.8%.

If taking 6th feature as label, a new synthetic feature will increase accuracy from 57.8% to 74.1%. However, if adding more synthetic feature such as one more feature, the accuracy will decrease from 74.1% to 69.8%.

Reason: We can think the synthetic feature as a basic value for this model. Because not all components of labels Y come from Input Data X, there are still some existy basic value for Y. However, if we add more synthetic feature, it will make the model more complex but it is meaningless