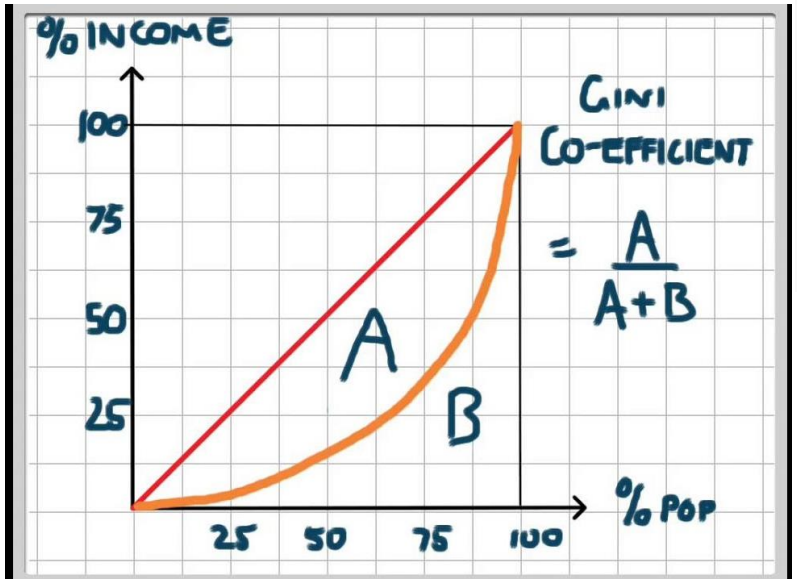
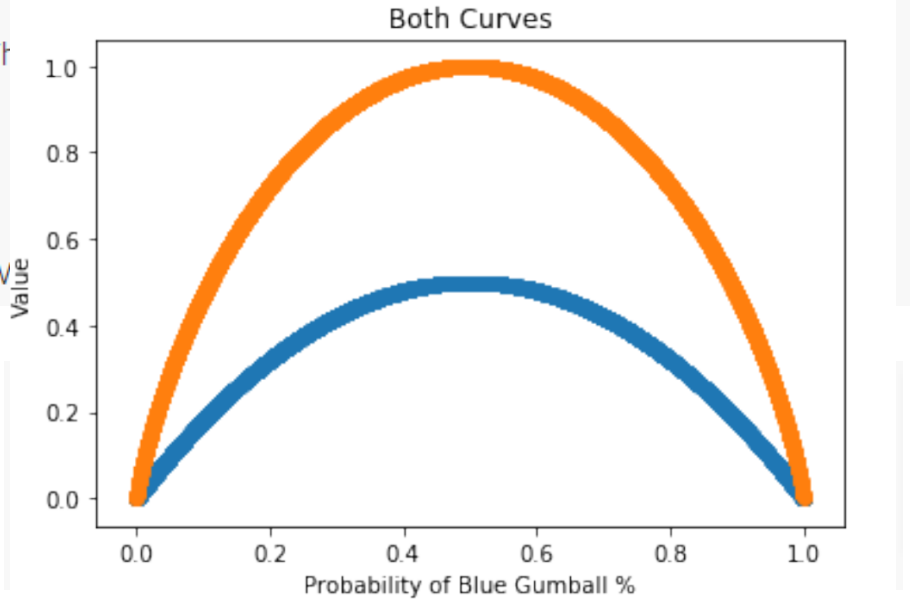
Q&A

Predictive Analytics: Week 1

Entropy and Gini

- Can you explain a little bit about Gini Index that you mention when you are showing the variables in the code



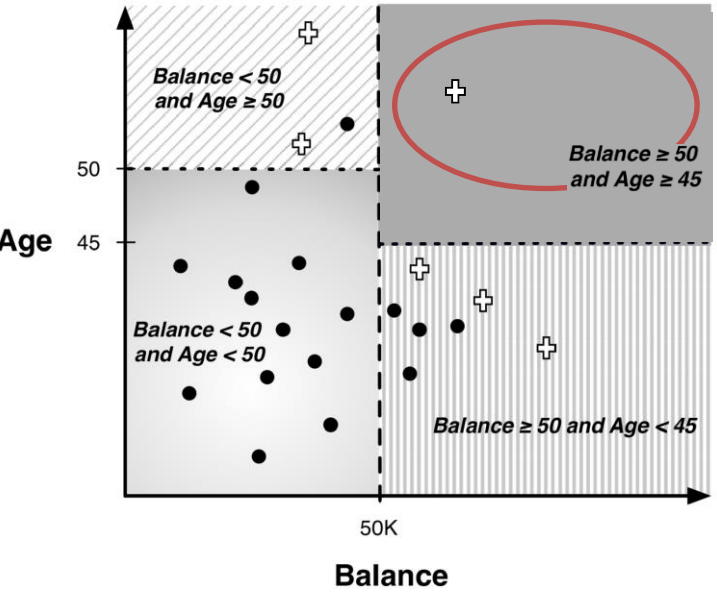


Laplace Correction

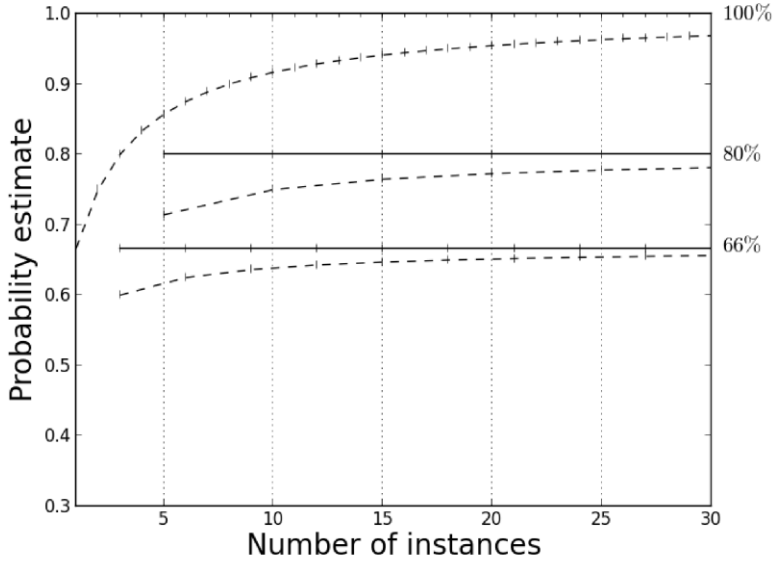
- Can you talk more about the Laplace correction, why it is used and when, and maybe give an example if there is time?
  - Why the node with only 1 data is problematic?
  - When do we need Laplace correction and what can we get from it?
  - In your add 100 example, which is (1+100) / (200+1), how did you calculate the denominator?
- Not sure what "the number of support" means when talking about laplace correction in decision tree.
- When do you use Laplace correction? Do you always use it when working with decision trees, or only when you have small sample sizes? How do you decide which values to use for the correction? Is the correction considered a "hyperparameter"?

Laplace Correction

Only One node in this leaf→ probability: 100%



- $p(c) = \frac{n+1}{n+m+2}$,
  - where n is the number of examples in the leaf belonging to class c, and m is the number of examples not belonging to class c.



1.9.2. Multinomial Naive Bayes

`MultinomialNB` implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice). The distribution is parametrized by vectors $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ for each class y, where n is the number of features (in text classification, the size of the vocabulary) and θ_{yi} is the probability $P(x_i \mid y)$ of feature i appearing in a sample belonging to class y.

The parameters θ_y is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

where $N_{yi} = \sum_{x \in T} x_i$ is the number of times feature i appears in a sample of class y in the training set T, and $N_y = \sum_{i=1}^{n} N_{yi}$ is the total count of all features for class y.

The smoothing priors $\alpha \geq 0$ accounts for features not present in the learning samples and prevents zero probabilities in further computations. Setting $\alpha = 1$ is called Laplace smoothing, while $\alpha < 1$ is called Lidstone smoothing.

KNN

- Regarding distance metrics for KNN, can you calculate distance metrics for categorical variables? What if you have both numerical and categorical independent variables in your KNN?

- How would KNN work with Categorical features?

In [1]:
```
data = [
    {'price': 850000, 'rooms': 4, 'neighborhood': 'Queen Anne'},
    {'price': 700000, 'rooms': 3, 'neighborhood': 'Fremont'},
    {'price': 650000, 'rooms': 3, 'neighborhood': 'Wallingford'},
    {'price': 600000, 'rooms': 2, 'neighborhood': 'Fremont'}
]
```

You might be tempted to encode this data with a straightforward numerical mapping:

In [2]:
```
{'Queen Anne': 1, 'Fremont': 2, 'Wallingford': 3};
```

It turns out that this is not generally a useful approach in Scikit-Learn: the package's models make the fundamental assumption that numerical features reflect algebraic quantities. Thus such a mapping would imply, for example, that *Queen Anne* < *Fremont* < *Wallingford*, or even that *Wallingford* - *Queen Anne* = *Fremont*, which (niche demographic jokes aside) does not make much sense.

In this case, one proven technique is to use *one-hot encoding*, which effectively creates extra columns indicating the presence or absence of a category with a value of 1 or 0, respectively. When your data comes as a list of dictionaries, Scikit-Learn's `DictVectorizer` will do this for you:

In [3]:
```
from sklearn.feature_extraction import DictVectorizer
vec = DictVectorizer(sparse=False, dtype=int)
vec.fit_transform(data)
```

Out[3]:
```
array([[     0,      1,      0, 850000,      4],
       [     1,      0,      0, 700000,      3],
       [     0,      0,      1, 650000,      3],
       [     1,      0,      0, 600000,      2]], dtype=int64)
```

Naïve Bayesian

- In general, I'm still struggling to understand the Naive Bayes classification. Specifically, I didn't completely understand the two self assessment questions.
  1. Why need the Naive assumption in the Naive Bayesian classifier?
  2. How to calculate P(X_k|C_i) when X_k is categorical/numerical?
- Why don't we just calculate P(Ci|X) directly instead of using the Bayesian Theorem to transform it? Is it because we can only use the naive assumption in this form? If so, when the dataset is large enough to have no sparse X combinations, can we use P(Ci|X) directly?
- For Naive Bayes, what are common ways to determine a cutoff probability for classification?

Learnt Model

- It seems like the model of Logistics Regression is a function, the model of Decision Tree is a tree, the model of KNN is just data points. Then what's the trained model of Naive Bayes? Is it the probabilities like P(X1|C1)...P(Xn|Cn), P(X1)...P(Xn) and P(C1)...P(C2) ready for use for combinations?

Multi-Class Measurement

- At the end of the 1-2 slides for Decision Trees and Basic Measurement, there's a multi-class confusion matrix slide (rather than binary). Could you go over a multi-class confusion matrix example? If there isn't enough time, could you touch on the difference(s) (if any) when using a multi-class confusion matrix?

The Confusion Matrix (Multi-Class)

- Assume that the target variable has more than two values (classes); e.g., C_1, C_2, ..., C_k
  - E.g., Risk = High, Medium, or Low
- Predicted(C_i)
  - Set of data points predicted to be of class C_i by the model
- Actual(C_i)
  - Set of data points that actually are of class C_i
- Then:

$$\text{Precision}_{C_i} = \frac{|\text{Predicted}(C_i) \cap \text{Actual}(C_i)|}{|\text{Predicted}(C_i)|}$$

Questions-1

- Regression vs Classification

- Supervised Learning vs Unsupervised Learning

Questions-2

- Hyper-Parameter Vs Parameter
- Overfitting
- How to set Hyper-Parameters in decision tree?
- Why need the Naive assumption in the Naive Bayesian classifier?