Enhancing Naïve Bayes with One-Dependence and Adaptive Weighting

James Beck, Jia Mody

01

Objective

Our Objective

Naïve Bayes in ML

Simple, efficient, effective

Challenges with Independence Assumption

Unrealistic in real world scenarios

Modifications for Real-World

Relaxing independence while maintaining efficiency

Our Approach

One-dependence estimators and adaptive weighting

02

Dataset/Preprocessing

Our Dataset

Kaggle	Instances	Attributes
Predicting house prices based on 12 attributes	545 instances, no missing values	12 attributes (i.e. # br, area, guestroom, furnishing status)

Preprocessing

No missing values

No missing values, no need to replace

No normalization

Naïve Bayes does not require normalization

Discretization

Discretized area and price, did not follow suit with others

Train Test Split

Training

436/545 instances--80%

Testing

109/545 instances--20%

```
[ ] import pandas as pd

[ ] Start coding or generate with AI.

[ ] df = pd.read_csv('/content/drive/MyDrive/ML/HousingCleaned.csv')

[ ] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

Double-click (or enter) to edit

```
[ ] from sklearn.model_selection import train_test_split

[ ] train, test = train_test_split(df, test_size=0.30, stratify=df.iloc[:, -1])

[ ] train.to_csv('train.csv', index=False)
    !cp train.csv /content/drive/MyDrive/ML

    test.to_csv('test.csv', index=False)
    !cp test.csv /content/drive/MyDrive/ML
```

03

Our Algorithm

Related Work

01	AODE	Averaged One-Dependence Estimators Small & Intermediate Datasets
02	AISWNB	Artificial Immune System $O(n \times m \times w^2 \times a)$
03	WANBIA	Conditional Log Likelihood Mean Squared Error
04	LWNB	Locally Weighted Algorithm Complexity

Our Algorithm

Superparent TAN

$$P(y|x_1, x_2, \dots, x_n) = P(y) \prod_{i=1}^{n} P(x_i|x(parent_i), y)$$

Instance is allowed to dependent on parent.
CMI (nonlinear) vs Cramer’sV (linear)

AISWNB

weight vectors are initialized

Vectors are fitted to training data

$$v_i^{t+1} = w_i^t + F \cdot N(0,1) \cdot (w_s^t - w_i^t)$$

04
Experimentation

Experiment Overview

CMI and Cramer's V to Select Parent

Compare CMI (attribute dependency with class) and Cramer'sV to select the super parent.

$$I(X_i; X_j|Y) = \sum_{x_i, x_j, y} P(x_i, x_j, y) log \frac{P(x_i, x_j|y)}{P(x_i|y)P(x_j|y)}$$

$$V(X_i; X_j) = \sqrt{\frac{\chi^2}{n(k-1)}}$$

Smoothing

Test multiple k values to find the optimal smoothing degree for handling unseen attribute-label pairs in Naive Bayes.

AISWNB Improvement Threshold

Experiment with the number of generations (m) and improvement threshold (T) to balance accuracy and training time.

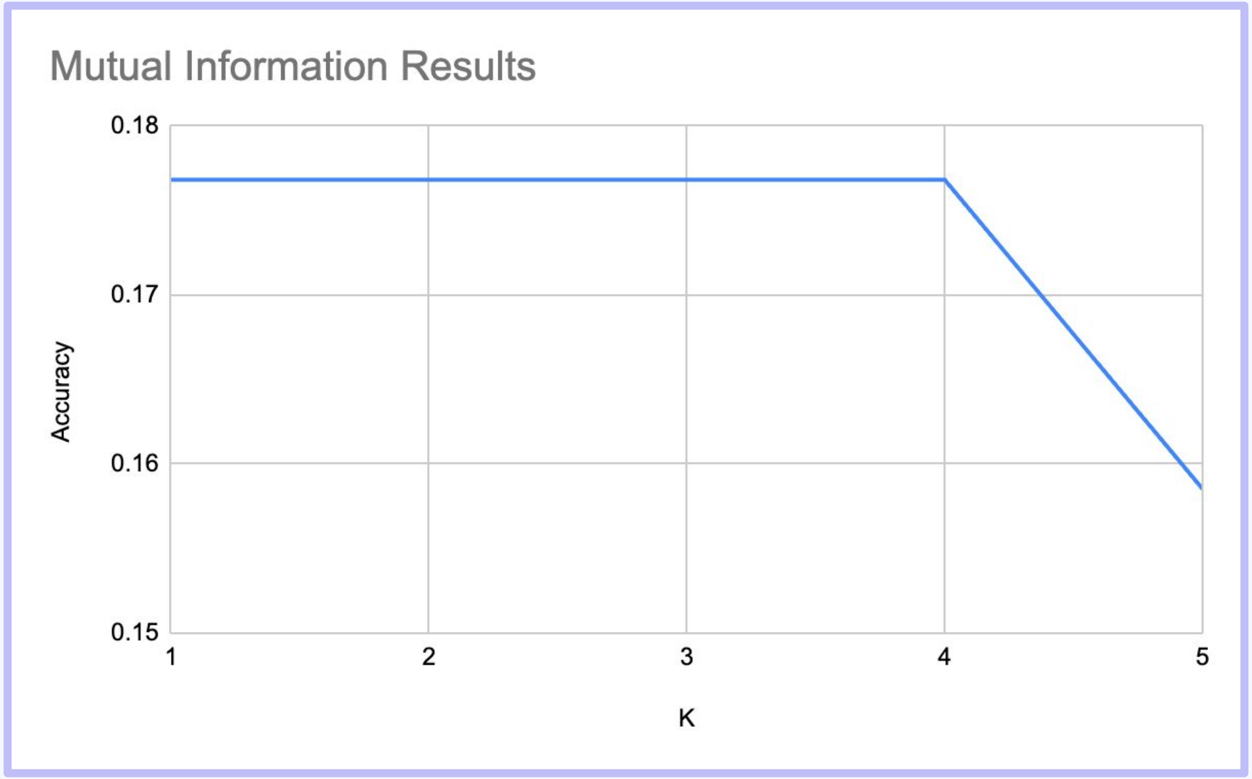
05

Discussion

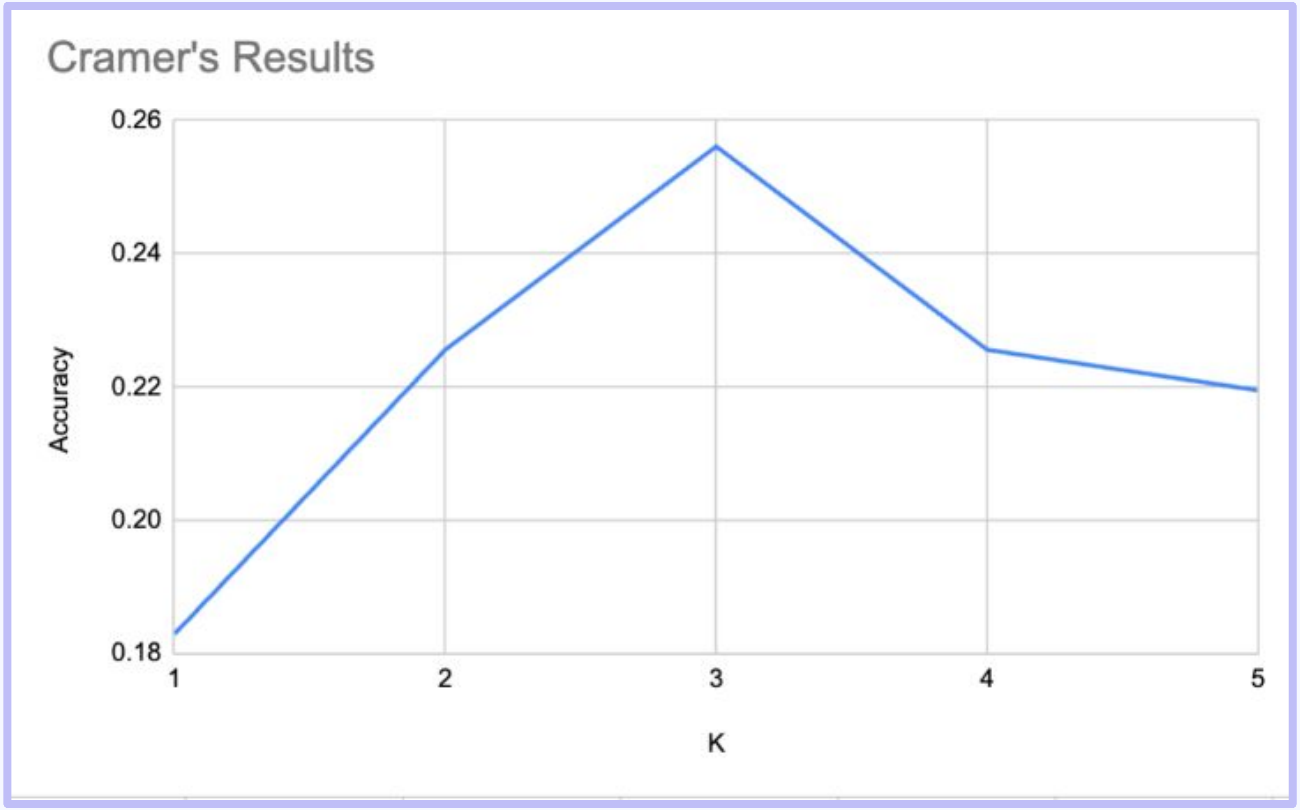
Results

Algorithm	Accuracy
Naive Bayes Normal	0.2256097561
MI, k=1	0.1768292683
MI, k=2	0.1768292683
MI, k=3	0.1768292683
MI, k=4	0.1768292683
MI, k=5	0.1585365854
Cramers k=1	0.1829268293
Cramers k=2	0.2256097561
Cramers k=3	0.256097561
Cramers k=4	0.2256097561
Cramers k=5	0.2195121951

Mutual Information Results



Cramer's Results



AISWNB

Algoirthm	Housing Test	Student Test	Housing Train	Student Train
Naive Bayes	0.2256097561	0.19	0.3648293963	0.2657142857
Superparent TAN	0.256097561	0.2433333333	0.4304461942	0.3414285714
AISW Superparent TAN	0.256097561	0.2366666667	0.4409448819	0.3428571429

Runtime: 3.6s vs. 33.62s!!

06

Conclusion

Conclusion

1. Results
   a. Best k value: 3
   b. Best algorithm: Cramer’sV, AISW w/ Superparent TAN

2. Future Studies
   a. Expanding project to more complex datasets
   b. Multiple dependencies
   c. Vast array of datasets
   d. Optimizing attribute weighting

Thanks!

Do you have any questions?

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**

Please keep this slide for attribution