CSCI 111-T

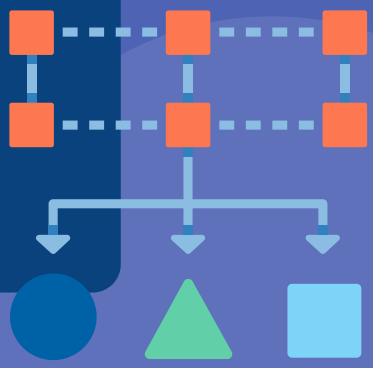
Tic-Tac-Toe Endgame Classification

Final project presentation

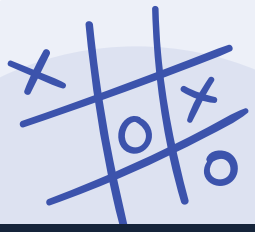
NEIL QUE | LUIS VILLADAREZ | ROMMEL YRAY

Content Outline

Topics for discussion



INTRODUCTION



Dataset Overview

- The dataset is all about the complete set of possible endgame configurations of tic-tac-toe games where "x" is played first. It consists of 10 columns and 958 observations.
- Entries for the first 9 columns are "x", "o", and "b" for blank.
- Entries for the last column are either "positive" or "negative". Positive if "x" wins, otherwise negative.

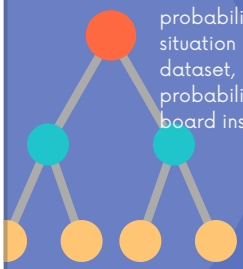
	0	1	2	3	4	5	6	7	8	9
0	x	x	x	x	o	o	x	o	o	positive
1	x	x	x	x	o	o	o	x	o	positive
2	x	x	x	x	o	o	o	o	x	positive
3	x	x	x	x	o	o	o	b	b	positive
4	x	x	x	x	o	o	b	o	b	positive

Methodology

Naive Bayes Classification

The probablity of situation A given situation B

The Naive Bayes Classification is based on the Naive bayes Theorem that aims to predict the probability of a situation A occuring given that situation B has occured. In the context of the dataset, the method will determine the probability of a win for player X given a certain board instance from the data set.

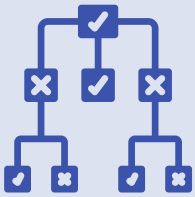


K-Nearest Neighbors

Determining an instance based on its surroundings

K-Nearest Neighbors (or KNN) is a type of supervised learning that classifies every new instance based on the K-neighboring instances that are nearest to it. In the context of the dataset, a board instance will be classified as either a win for player X or not by determining if its K nearest neighbors (in terms of board instance) will lead to a win for player X or not.

Methodology



Why Not Decision Trees?

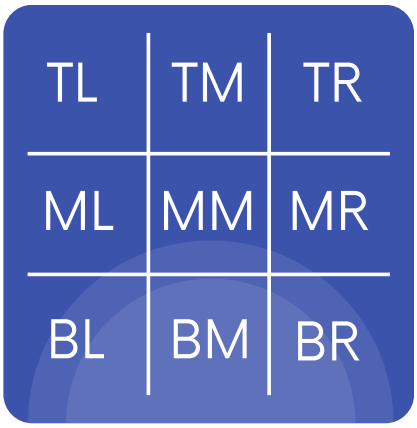
Intuitively, a Decision tree method would be ideal in determining if player X will win. However, due to the nature of the data set, it is difficult to predict an outcome because of the lack of order in terms of moves in the data set.

	0	1	2	3	4	5	6	7	8	9
0	x	x	x	x	o	o	x	o	o	positive
1	x	x	x	x	o	o	o	x	o	positive
2	x	x	x	x	o	o	o	o	x	positive
3	x	x	x	x	o	o	o	b	b	positive
4	x	x	x	x	o	o	b	o	b	positive

Methodology

Part 1: Whole Dataset

Data Preprocessing



- Columns were labled 'tl', 'tm', 'tr', 'ml', 'mm', 'mr', 'bl', 'bm', 'br', and 'result'.
- The string values were changed into integer values: B = 0, O = 1, X = 2

	tl	tm	tr	ml	mm	mr	bl	bm	br	result
0	2	2	2	2	1	1	2	1	1	positive
1	2	2	2	2	1	1	1	2	1	positive
2	2	2	2	2	1	1	1	1	2	positive
3	2	2	2	2	1	1	1	0	0	positive
4	2	2	2	2	1	1	0	1	0	positive

Methodology

Data Splitting

- The dataset was split into two parts wherein the values of the first 9 columns would be X while values of the result column would be Y
- 30% test size was chosen since it yielded the most accurate model performance

```
X = games_int[['tl', 'tm', 'tr', 'ml', 'mm', 'mr', 'bl', 'bm', 'br']].values
Y = games_int['result']
```

```
t_size = 0.3
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=t_size, random_state=0)
print('Distribution of classes for the test data:')
pd.Series(y_test).value_counts()
```

```
Distribution of classes for the test data:

positive    188
negative    100
Name: result, dtype: int64
```

Naive-Bayes and K-NN

- Naive-Bayes and K-Nearest Neighbors classifications were created by fitting the training and the testing sets
- We also tried some specific predictions where x in the middle and o bottom middle using the created models

```
predict_specific(nb_model1, [0,0,0,0,2,0,0,1,0])
```

```
predict_specific(knn_model1, [0,0,0,0,2,0,0,1,0])
```

- Confusion Matrix and other metrics

Methodology

Part 2: Taking Sample

Taking Sample from Dataset

- Only used "mm" or middle-middle and "bm" or bottom-middle entries as the sample for gathering predictions
- Created the confusion matrix and computed for the metrics
- Created a new data frame with only valid moves for mm and bm to compute for the metrics for comparison
- Predicted a specific case

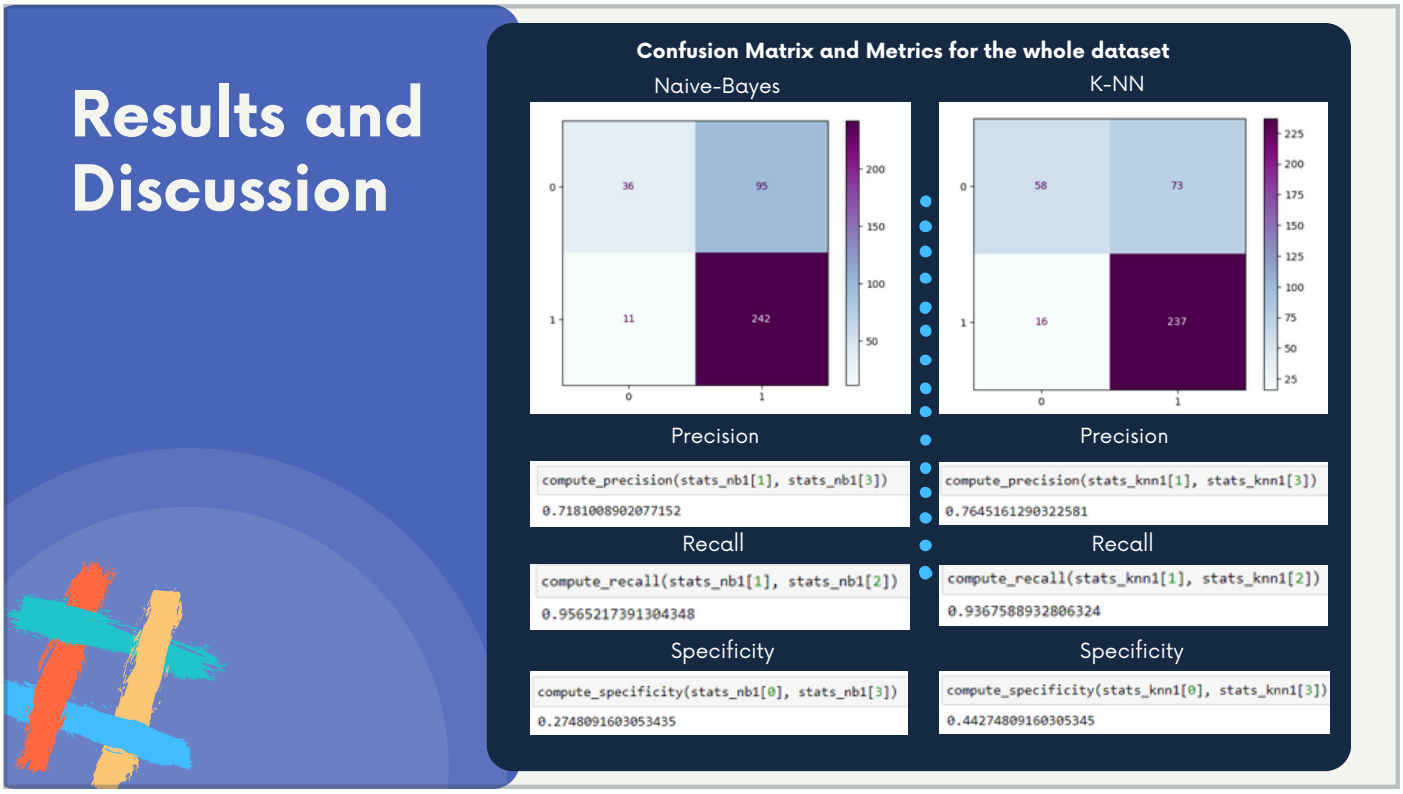
	mm	bm	result	prediction
1	o	x	positive	negative
2	o	o	positive	positive
5	o	b	positive	positive
6	o	o	positive	positive
8	o	o	positive	positive
...	...	...	...	...
951	x	o	negative	positive
952	o	o	negative	positive
954	x	o	negative	positive
955	o	o	negative	positive
956	x	o	negative	positive

Methodology

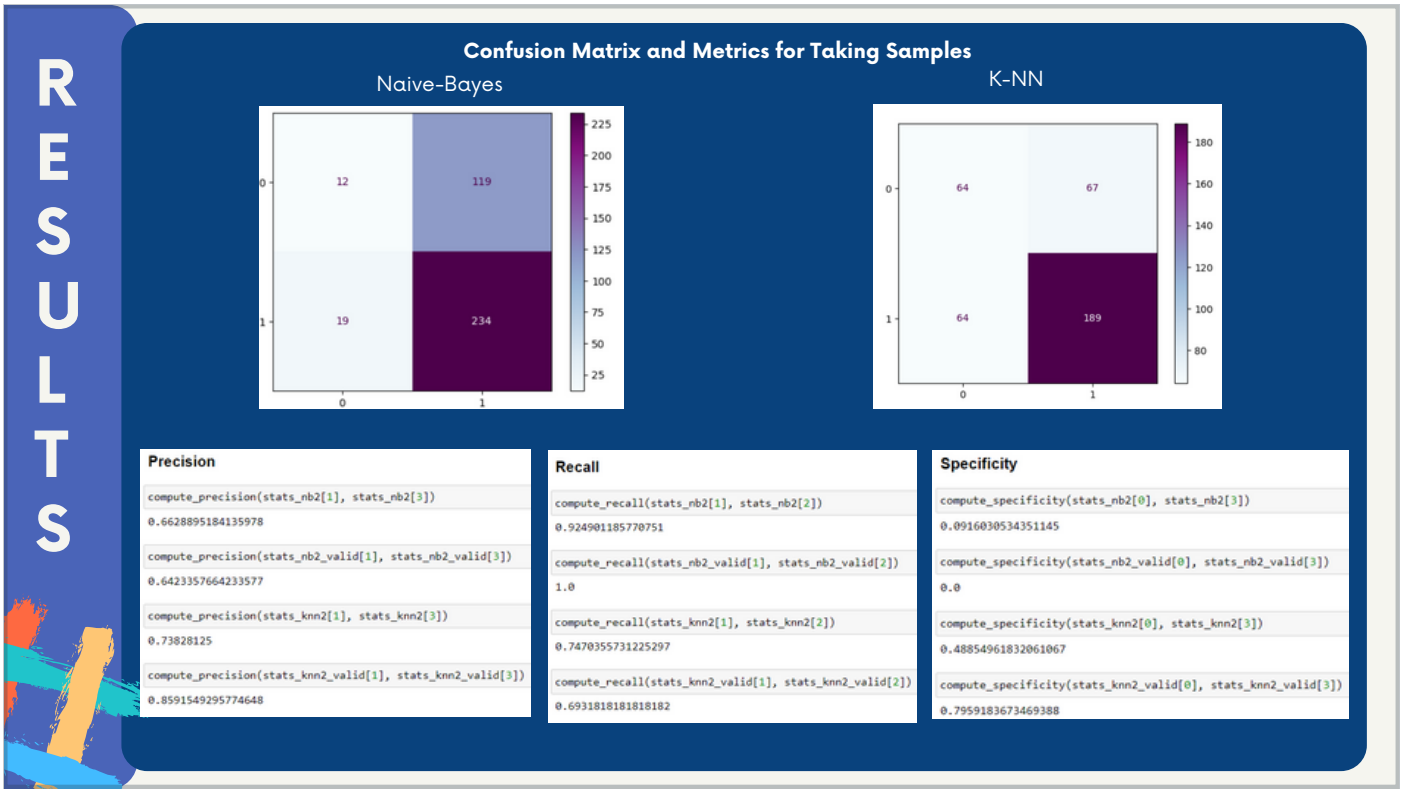
Part 3: Downsample

Downsampling Positive Results

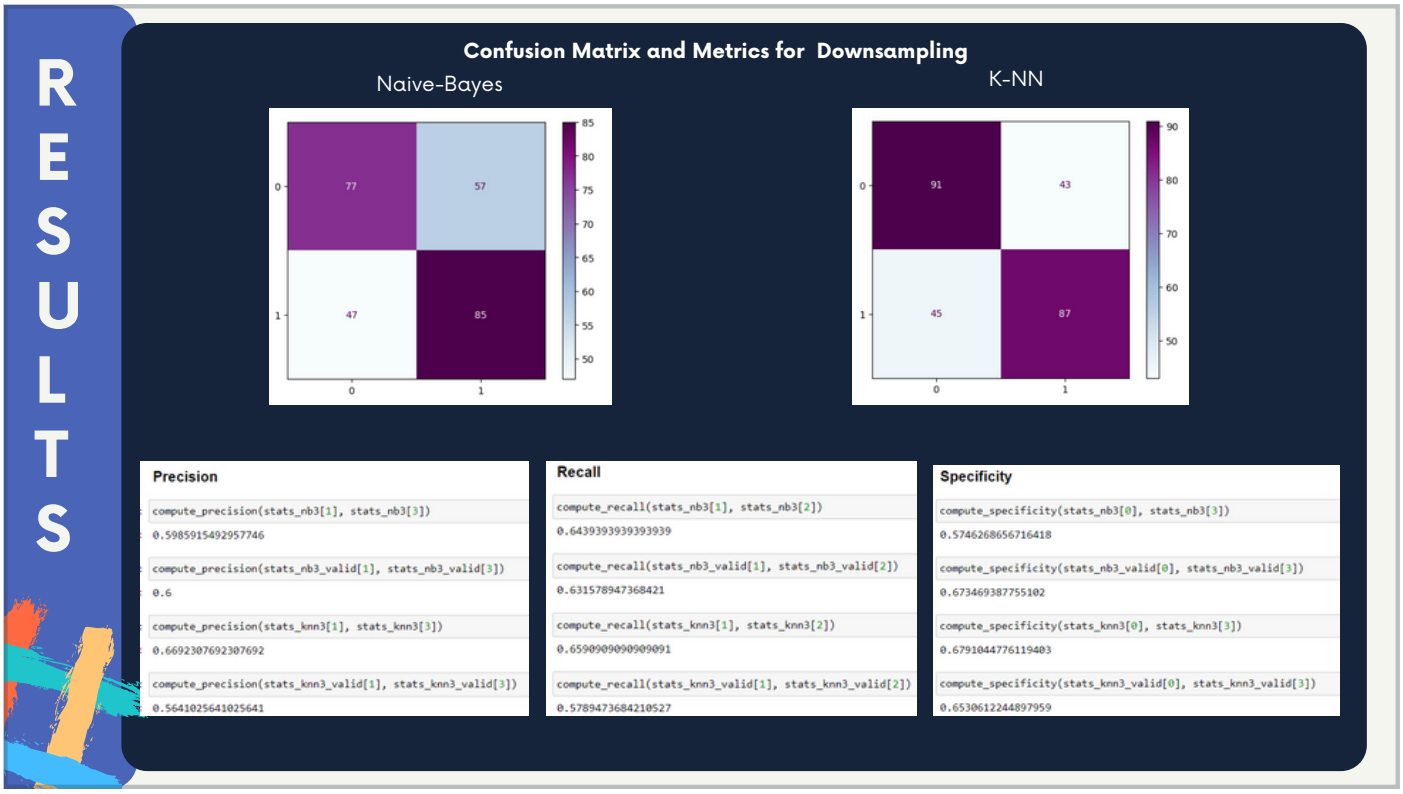
- Reduced the number of positive results from 626 to only 332 so that positive and negative results would be equal since it may led the model to lean into having positive predictions
- Created Naive-Bayes Model and K-NN
- Took sample with only 2 moves and created a new dataframe with valid configuration
- Created Confusion Matrix and other metrics
- Specific Predictions



- too many positive predictions, as seen in confusion matrices having higher numbers at the right
- 'good' at making positive predictions through recall
- pretty bad at making negatives
- the model might have just predicted positive mostly because it can give high accuracy (.724 and .8281)



- still very many positive predictions for the nb model
- nb for valid boards have 1 recall and 0 specificity -> can be seen in distribution of preds: 0-49-0-88
- knn better, knn valid boards pretty nice, but can be attributed to chance because valid boards only consist of x-o or o-x, the number of instances for each may have helped in giving high metrics
- accuracy for each at least 0.64, highest is 0.73



- almost same accuracy across the board, around .65 with .77 highest
- lower precision than others
- lower recall but higher specificity compared to others
- based on metrics, this may provide more realistic view of performance of models

Results and Discussion

Comparison

- Throughout all trials, we observed that KNN has higher accuracy except for one case
- This is likely the case because KNN can take into account the columns as a whole, compared to NB that assumes independence of attributes
- There is also a bit of optimization done in the implementation of KNN, as we take the accuracy of the best performing model given a range of values for k

Thank You!

NEIL QUE | LUIS VILLADAREZ | ROMMEL YRAY
CSCI 111-T