Implementation of Classification Algorithms

Data Minor

Team Members

- Sang Choi
- Jose Cruz

Table of Content

Tools

- Python3 and jupyter notebook
- Scikit Learn
- Github
- Anaconda

Dataset

- Heart Failure Detection Dataset
  - Heart Failure is the number 1 cause of death globally
  - Contains medical information of heart failure patients
  - Can predict who's most at risk of dying from the heart attack
  - Label: DEATH_EVENT, 1 or 0
  - Features
    - Age
    - Anaemia
    - Creatinine Phosphokinase: level of CPK enzyme in the blood
    - Diabetes
    - Ejection Fraction: percentage of blood leaving the heart at each contraction
    - High Blood Pressure
    - Sex
    - Etc

Dataset

Data Overview

```
df.describe()
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium
count	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.00000	299.000000
mean	60.833893	0.431438	581.839465	0.418060	38.083612	0.351171	263358.029264	1.39388	136.625418
std	11.894809	0.496107	970.287881	0.494067	11.834841	0.478136	97804.236869	1.03451	4.412477
min	40.000000	0.000000	23.000000	0.000000	14.000000	0.000000	25100.000000	0.50000	113.000000
25%	51.000000	0.000000	116.500000	0.000000	30.000000	0.000000	212500.000000	0.90000	134.000000
50%	60.000000	0.000000	250.000000	0.000000	38.000000	0.000000	262000.000000	1.10000	137.000000
75%	70.000000	1.000000	582.000000	1.000000	45.000000	1.000000	303500.000000	1.40000	140.000000
max	95.000000	1.000000	7861.000000	1.000000	80.000000	1.000000	850000.000000	9.40000	148.000000

Selected Algorithms

1. Decision Tree
2. Naive Bayes
3. K-Nearest Neighbor

Testing Method

- 10-Fold Cross Validation using Sklearn's KFold
- Accuracy Test
  - Compare accuracy between Scikit Learn to DM's implementation
- Performance Test
  - Compare runtime between Scikit Learn and DM's implementation

Decision Tree Overview

- A decision tree is a flow-chart like tree structure where an internal node represents a feature, the branches represent a decision rule, and each leaf represents the outcome.
- To build a decision tree:
  - Select your best attribute to split the data.
  - Make that attribute a decision node and break the dataset into smaller subsets.
  - Build the tree by repeating the process recursively for each child until one of the following conditions match:
    - All tuples belong to the same attribute value
    - There are no more remaining attributes
    - There are no more instances

Decision Tree Implementation

```
{'time <= 67.5': [{'platelets <= 214500.0': [1.0,
    {'platelets <= 224500.0': [{'time <= 57.0': [0.0, 1.0]},
      {'age <= 66.5': [{'creatinine_phosphokinase <= 85.5': [0.0,
          {'platelets <= 307500.0': [{'time <= 24.5': [1.0,
              {'serum_sodium <= 134.5': [1.0,
                {'serum_sodium <= 142.5': [0.0, 1.0]}]}]},
            1.0]}]},
        1.0]}]}]},
  {'serum_creatinine <= 1.55': [{'ejection_fraction <= 27.5': [{'time <= 78.5': [1.0,
        {'time <= 148.0': [0.0,
          {'time <= 178.0': [1.0,
            {'time <= 210.5': [0.0,
              {'serum_creatinine <= 0.9500000000000001': [0.0, 1.0]}]}]}]}]},
      {'age <= 79.0': [{'creatinine_phosphokinase <= 2307.5': [{'serum_creatinine <= 0.6499999999999999': [{'time <= 123.0':
[0.0,
              1.0]},
            {'platelets <= 349500.0': [0.0,
              {'serum_creatinine <= 1.2000000000000002': [0.0,
                {'serum_sodium <= 137.5': [1.0, 0.0]}]}]}]},
          {'diabetes <= 0.5': [0.0, 1.0]}]},
        {'platelets <= 239000.0': [{'sex <= 0.5': [1.0, 0.0]}, 1.0]}]}]},
    {'ejection_fraction <= 22.5': [1.0,
      {'creatinine_phosphokinase <= 72.5': [0.0,
        {'high_blood_pressure <= 0.5': [{'platelets <= 226000.0': [{'serum_sodium <= 137.5': [1.0,
              {'serum_sodium <= 142.5': [0.0, 1.0]}]},
            {'sex <= 0.5': [{'creatinine_phosphokinase <= 454.0': [0.0, 1.0]},
              0.0]}]},
          1.0]}]}]}]}]}
```

Decision Tree Results

	dtc_accuracy_model	dtc_model_time	mydtc_accuracy_model	mydtc_model_time	%diff
0	100.0	4	80.000000	476	11900.000000
1	100.0	3	80.000000	421	14033.333333
2	100.0	2	76.666667	386	19300.000000
3	100.0	2	73.333333	354	17700.000000
4	100.0	4	86.666667	364	9100.000000
5	100.0	3	83.333333	367	12233.333333
6	100.0	2	70.000000	395	19750.000000
7	100.0	3	66.666667	375	12500.000000
8	100.0	2	86.666667	403	20150.000000
9	100.0	3	82.758621	439	14633.333333

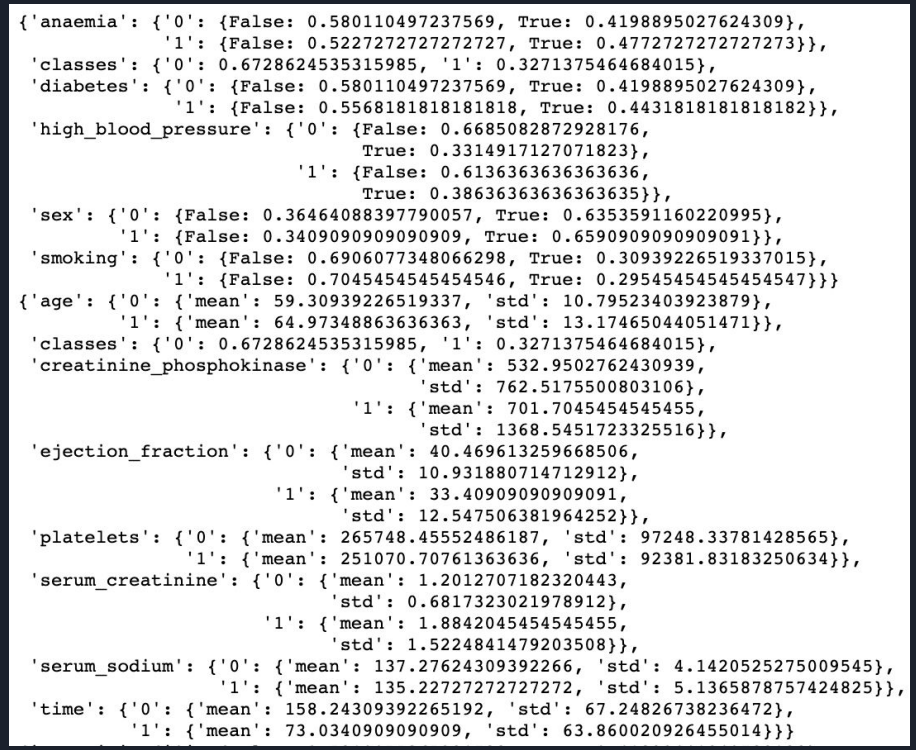
- Slightly different accuracy between sklearn and our implementation
- Significant performance upperhand for SKlearn
    - Leverages Parallelization

Naive Bayes Overview

- The Naive Bayes is a classification algorithm used for binary or multiclass classifications.
- Calculates probability of a piece of data belonging to a given class. It is naive in the sense that the calculations of the probabilities for each class are simplified to make their results tractable.
- All features are assumed to be conditionally independent.

$$P\left(class\,|data\right) = \frac{(P(class|data)\cdot P(class))}{P(data)}$$

Naive Bayes Implementation



```
{'anaemia': {'0': {False: 0.5801104972237569, True: 0.4198895027624309},
             '1': {False: 0.5227272727272727, True: 0.4772727272727273}},
 'classes': {'0': 0.6728624535315985, '1': 0.3271375464684015},
 'diabetes': {'0': {False: 0.5801104972237569, True: 0.4198895027624309},
              '1': {False: 0.5568181818181818, True: 0.4431818181818182}},
 'high_blood_pressure': {'0': {False: 0.6685082872928176,
                               True: 0.3314917127071823},
                         '1': {False: 0.6136363636363636,
                               True: 0.38636363636363635}},
 'sex': {'0': {False: 0.36464088397790057, True: 0.6353591160220995},
         '1': {False: 0.3409090909090909, True: 0.6590909090909091}},
 'smoking': {'0': {False: 0.6906077348066298, True: 0.30939226519337015},
             '1': {False: 0.7045454545454546, True: 0.29545454545454547}}}
{'age': {'0': {'mean': 59.30939226519337, 'std': 10.79523403923879},
         '1': {'mean': 64.97348863636363, 'std': 13.17465044051471}},
 'classes': {'0': 0.6728624535315985, '1': 0.3271375464684015},
 'creatinine_phosphokinase': {'0': {'mean': 532.9502762430939,
                                    'std': 762.5175500803106},
                              '1': {'mean': 701.7045454545455,
                                    'std': 1368.5451723325516}},
 'ejection_fraction': {'0': {'mean': 40.469613259668506,
                             'std': 10.931880714712912},
                       '1': {'mean': 33.40909090909091,
                             'std': 12.547506381964252}},
 'platelets': {'0': {'mean': 265748.45552486187, 'std': 97248.33781428565},
               '1': {'mean': 251070.70761363636, 'std': 92381.83183250634}},
 'serum_creatinine': {'0': {'mean': 1.2012707182320443,
                            'std': 0.6817323021978912},
                      '1': {'mean': 1.8842045454545455,
                            'std': 1.5224841479203508}},
 'serum_sodium': {'0': {'mean': 137.27624309392266, 'std': 4.1420525275009545},
                  '1': {'mean': 135.22727272727272, 'std': 5.1365878757424825}},
 'time': {'0': {'mean': 158.24309392265192, 'std': 67.24826738236472},
          '1': {'mean': 73.0340909090909, 'std': 63.860209264550l4}}}
```

- Naive Bayes Training creates a dictionary to calculate priori probabilities
- Numerical Data Type
  - Mean and Std
- Categorical Data Type
  - Class distribution

Naive Bayes Results

clf_scikit	clf_dm	clf_accuracy	clf_sklearn_time	clf_dm_time	%diff
0.733333	0.733333	100.0	10.0	26.0	-383.333333
0.800000	0.800000	100.0	6.0	25.0	-400.000000
0.600000	0.600000	100.0	6.0	25.0	-520.000000
0.666667	0.666667	100.0	9.0	33.0	-414.285714
0.466667	0.466667	100.0	5.0	23.0	-366.666667
0.766667	0.766667	100.0	7.0	27.0	-433.333333
0.666667	0.666667	100.0	10.0	25.0	-325.000000
0.666667	0.666667	100.0	8.0	24.0	-575.000000
0.700000	0.700000	100.0	6.0	23.0	-460.000000
0.724138	0.724138	100.0	7.0	26.0	-316.666667

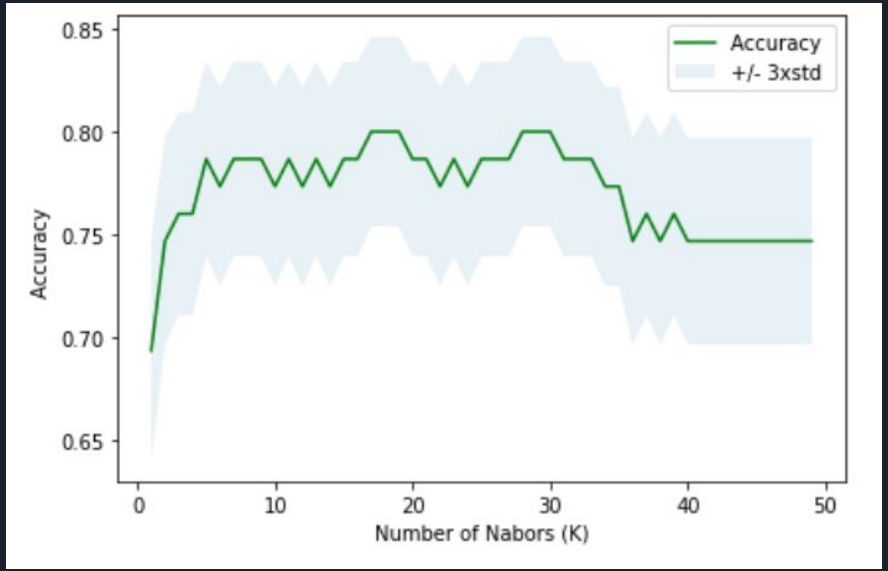
gnb_scikit	gnb_dm	gnb_accuracy	gnb_sklearn_time	gnb_dm_time	%diff
0.800000	0.800000	100.0	6.0	29.0	-383.333333
0.900000	0.900000	100.0	6.0	30.0	-400.000000
0.666667	0.666667	100.0	5.0	31.0	-520.000000
0.733333	0.733333	100.0	7.0	36.0	-414.285714
0.600000	0.600000	100.0	6.0	28.0	-366.666667
0.800000	0.800000	100.0	6.0	32.0	-433.333333
0.733333	0.733333	100.0	8.0	34.0	-325.000000
0.833333	0.833333	100.0	4.0	27.0	-575.000000
0.933333	0.933333	100.0	5.0	28.0	-460.000000
0.724138	0.724138	100.0	6.0	25.0	-316.666667

- Compared with Sklearn's Categorical NB and Gaussian NB
- 100% prediction match between SKlearn and DM's implementation
- However, significant performance upperhand for SKlearn
    - Leverages Parallelization(3-6x)

K-Nearest Neighbors

- This is a type of classification algorithm used to cluster neighbors together to a predefined K value.
- Vectors in a multidimensional feature space are used for the training data.
- Find K nearest tuples and predict based on labels of these neighbors.
- Uses Euclidean distance for similarity

KNN: Finding the right K-value



KNN Results

	knn_accuracy_model	knn_model_time	myknn_accuracy_model	myknn_model_time	%diff
0	10.000000	8	10.000000	5359	66987.500000
1	16.666667	5	16.666667	5393	107860.000000
2	56.666667	5	56.666667	5358	107160.000000
3	83.333333	5	83.333333	5511	110220.000000
4	83.333333	5	83.333333	5328	106560.000000
5	83.333333	5	83.333333	5328	106560.000000
6	66.666667	6	66.666667	5405	90083.333333
7	83.333333	5	83.333333	13230	264600.000000
8	90.000000	16	90.000000	18768	117300.000000
9	89.655172	17	89.655172	18437	108452.941176

Thank you