CS5644 Assignment 2

Q1. Data Analysis on Congressional Voting Records dataset

a. Introduction

This report documented the machine learning analysis of taking the votes of US congressmen/congresswomen as input and predicting whether they are a Republican or a Democrat. The process including experimenting with three different approaches to deal with missing data with the machine learning model of both the decision tree model and the Naïve Bayes classifier. The goal is to find out which scenario will have the best result for this particular case.

The voting Records dataset is from the UCI machine learning repository (https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records).

b. Methods

1) Data preprocessing

First, read the data in a CSV way to get an idea of what it looks like.

```
In [662]: data = pd.read_csv('congressional+voting+records/house-votes-84.data')
          data.head()   # No good, format is messed up
```

Out[662]:

	republican	n	y	n.1	y.1	y.2	y.3	n.2	n.3	n.4	y.4	?	y.5	y.6	y.7	n.5	y.8
0	republican	n	y	n	y	y	y	n	n	n	n	n	y	y	y	n	?
1	democrat	?	y	y	?	y	y	n	n	n	n	y	n	y	y	n	n
2	democrat	n	y	y	n	?	y	n	n	n	n	y	n	y	n	n	y
3	democrat	y	y	y	n	y	y	n	n	n	n	y	?	y	y	y	y
4	democrat	n	y	y	n	y	y	n	n	n	n	n	n	y	y	y	y

After reading the .names file understanding the data, we can find that:The data is missing the header.

I. The data is missing the header
II. There are some missing values representing there are some answers that are neither yes nor no
III. Only class-name not in the format of y/n

To apply Sklearn for building the machine learning model, some preprocessing processes are needed.

If we read the data line by line, we can see that it already has a format that can easily be converted to a data frame.

The actual data is in the ".data" suffix

```
In [764]: with open("congressional+voting+records/house-votes-84.data", "rt") as fin:
              for line in fin:
                  print(line.strip())
```

```
republican,n,y,n,y,y,y,n,n,n,y,?,y,y,y,n,y
republican,n,y,n,y,y,y,n,n,n,n,n,y,y,y,n,?
democrat,?,y,y,?,y,y,n,n,n,n,y,n,y,y,n,n
democrat,n,y,y,n,?,y,n,n,n,n,y,n,y,n,n,y
democrat,y,y,y,n,y,y,n,n,n,n,y,?,y,y,y,y
democrat,n,y,y,n,y,y,n,n,n,n,n,n,y,y,y,y
democrat,n,y,n,y,y,y,n,n,n,n,n,n,?,y,y,y
republican,n,y,n,y,y,y,n,n,n,n,n,n,y,y,?,y
republican,n,y,n,y,y,y,n,n,n,n,n,y,y,y,n,y
democrat,y,y,y,n,n,n,y,y,y,n,n,n,n,n,?,?
republican,n,y,n,y,y,y,n,n,n,n,n,?,?,y,y,n,n
republican,n,y,n,y,y,y,y,n,n,n,n,y,?,y,y,?,?
democrat,n,y,y,n,n,n,y,y,y,n,n,n,y,n,?,?
democrat,y,y,y,n,n,y,y,y,?,y,y,?,n,n,y,?
republican,n,y,n,y,y,y,n,n,n,n,n,y,?,?,n,?
republican,n,y,n,y,y,y,n,n,n,y,n,y,y,?,n,?
democrat,y,n,y,n,n,y,n,y,?,y,y,y,?,n,n,y
democrat,y,?,y,n,n,n,y,y,y,n,n,n,y,n,y,y
republican,n,y,n,y,y,y,n,n,n,n,n,?,y,y,n,n
```

After splitting the content by "," and ignoring the hidden "\n" line by line, and then prepare a list of column names from. names file, we can combine them together to become a pandas data frame. This data frame has 435 instances, and 17 attributes, which matched with what. names stated.

The next step will be replacing the element as a more convenient representation for machine learning.

I. 'n' be 0, 'y' be 1
II. 'republican' be 0, 'democrat' be 1
III. '?' be NaT value in pandas for dealing with the missing data

Finally, we can get the data is suitable for applying sklearn to perform machine learning.

Out[666]:

	class-name	handicapped-infants	water-project-cost-sharing	adoption-of-the-budget-resolution	physician-fee-freeze	el-salvador-aid	religious-groups-in-schools	anti-satellite-test-ban	aid-to-nicaraguan-contras	mx-missile	immigration
0	0	0	1	0	1	1	1	0	0	0	1
1	0	0	1	0	1	1	1	0	0	0	0
2	1	NaT	1	1	NaT	1	1	0	0	0	0
3	1	0	1	1	0	NaT	1	0	0	0	0
4	1	1	1	1	0	1	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
430	0	0	0	1	1	1	1	0	0	1	1
431	1	0	0	1	0	0	0	1	1	1	1
432	0	0	NaT	0	1	1	1	0	0	0	0
433	0	0	0	0	1	1	1	NaT	NaT	NaT	NaT
434	0	0	1	0	1	1	1	0	0	0	1

435 rows × 17 columns

2) Machine learning models

There are 3 scenarios for dealing with the missing values and 2 kinds of models we can experiment with. For each, the model will be tested with a 5-fold cross-validation to compare the performance.

Discard instances that have missing feature values		Naïve Bayes classifier
Treat "missing" as if it is a value	X	Decision Trees
Impute missing values		

The approach for 3 scenarios is implemented as:
Discard instances that have missing feature values -> Drop the row if any missing value
Treat "missing" as if it is a value -> Replace missing value to be -1, become three-valued features
Impute missing values -> for each feature, replace missing values with the most common value

c. Results

1) Metrics and Scoring by directly using sklearn.metrics with 80/20 train/test split

I) Discard instances that have missing feature values

Decision Trees

Precision Score	Recall Score	Test F1 Score
0.93679	0.93617	0.93605

Naïve Bayes classifier

Precision Score	Recall Score	Test F1 Score
0.91721	0.91490	0.91434

II) Treat "missing" as if it is a value

Decision Trees

Naïve Bayes classifier

Precision Score	Recall Score	Test F1 Score
0.93142	0.93103	0.93071

Precision Score	Recall Score	Test F1 Score
0.88991	0.88506	0.88561

III) Impute missing values

Decision Trees

Precision Score	Recall Score	Test F1 Score
0.96559	0.96552	0.96542

Naïve Bayes classifier

Precision Score	Recall Score	Test F1 Score
0.93278	0.93103	0.93140

2) 5-Fold Cross Validation using model_selection.cross_validate library

I) Discard instances that have missing feature values

Decision Trees

Precision Score	Recall Score	Test F1 Score
0.94574	0.94385	0.94383

Naïve Bayes classifier

Precision Score	Recall Score	Test F1 Score
0.91901	0.91425	0.91416

II) Treat "missing" as if it is a value

Decision Trees

Precision Score	Recall Score	Test F1 Score
0.94563	0.94253	0.94257

Naïve Bayes classifier

Precision Score	Recall Score	Test F1 Score
0.90487	0.89655	0.89743

III) Impute missing values

Decision Trees

Precision Score	Recall Score	Test F1 Score
0.95775	0.95632	0.95639

Naïve Bayes classifier

Precision Score	Recall Score	Test F1 Score
0.91084	0.90345	0.90425

Whether using a cross-validation or not, from the tables, we can observe that among all scenarios, impute missing values have the best overall scoring. When it comes to the models, decision trees often perform better than the Naïve Bayes classifier for this particular dataset. The difference between whether to use a cross-validation is inconsistent but minimal.

In sum, the best approach in this case is to impute missing values and apply them to a decision tree.

d. Discussion

Both Decision Tree and Naïve Bayes classifier are one of the most commonly used supervised machine learning algorithms for classification tasks. In this particular dataset, the decision tree model performs better in general to predict the class name. There are some assumptions about why a decision tree is doing well. Firstly, the features are in Boolean discrete responses, and a decision tree can often capture relations better since it can clearly split the two values of each response. Secondly, in some situations, a simpler model often performs better than a complex one, especially when the dataset is not significant. One of the disadvantages of a decision tree is it is easier to be overfitting. However, for this dataset's scale, the tree's depth won't be too deep for learning too much. Lastly, a Naïve Bayes classifier is built to estimate conditional probabilities of features given classes, prior class probabilities. If the dataset is not large, some features only appear a few times or never occur. This will affect the estimation result since a Naïve Bayes classifier relies on the correctness of the probabilities.

e. Conclusion

In conclusion, we performed a machine learning analysis to predict if a congressman/congresswoman is a Republican or a Democrat based on their voting history. There are some missing data in the dataset, so we experimented 3 different approaches to deal with the missing values. In each scenario, both a decision tree and a Naïve Bayes classifier are applied. The result showed that imputing missing values and applying them to a decision tree gives the best scoring result. This may be due to the nature of the models and the scale of the dataset. A decision tree often performs well in Boolean output values and a smaller dataset.

Q2. For what type of dataset would you choose decision trees as a classifier over Naive Bayes? Vice versa?

Using Decision Trees:

1. When the features in the dataset are rather dependent, it is better to use a decision tree since it can capture the underlying relations better.
2. If we want a clear explanation for the prediction result, a decision tree is better since it provides a clear principle of making decisions in its path.
3. A decision tree is better for non-linear relations between the features and the target.
4. If the features seem not equally important

Dataset example: Iris Dataset, Car Evaluation Dataset

Using Naïve Bayes Classifier:

1. When the features in the dataset are rather independent, NBC learning by simply comprises of computing conditional probabilities and treating each feature independently.
2. If the explanation for the relations between features and prediction result is not a primary target.
3. If there are tremendous features, NBC often learns more efficiently and effectively.
4. If the features seem equally important

Dataset example: Spam email detection, Newsgroups Dataset