OSDA_big_homework_report
Xie Pujun
04.12.2023

1. Dataset and Binarize

I choose three dataset in the big homework.

(1)congressional voting records Mushroom Classification (kaggle.com)
(2)mushroom classification Mushroom Classification (kaggle.com)
(3)spambase spambase (kaggle.com)

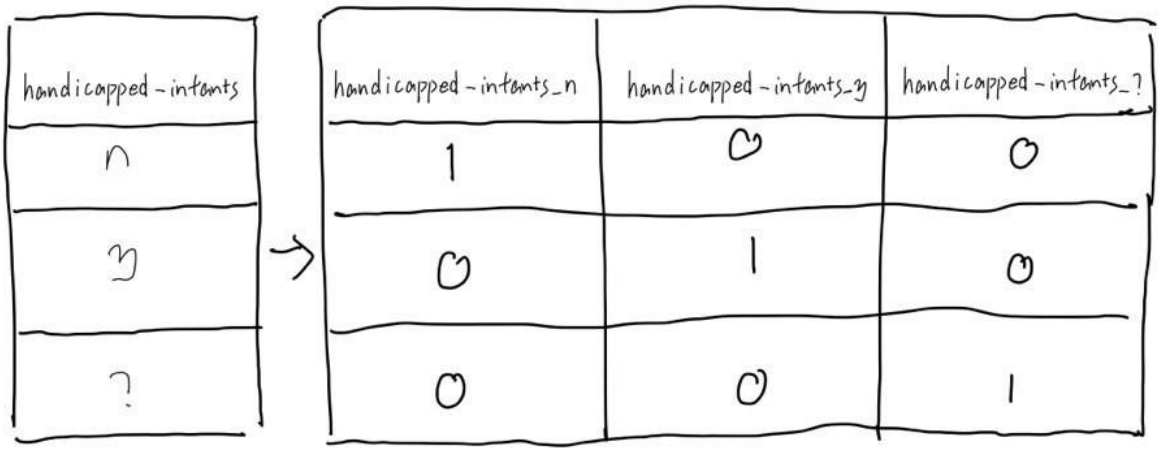
(1)In the dataset of congressional voting records, We have two class: democrat, republican and other 16 attributes:

```
C:\Users\30639\PycharmProjects\osda_hw\venv\Scripts\python.exe C:\Users\30639\Pycharm
Index(['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', ' synfuels-corporation-cutback', ' education-spending',
       ' superfund-right-to-sue', ' crime', ' duty-free-exports',
       ' export-administration-act-south-africa'],
      dtype='object')
```

Every attributes have three states y, n and ? . So we should do One-hot encoding:

```
Index([' handicapped-infants_?', ' handicapped-infants_n',
       ' handicapped-infants_y', ' water-project-cost-sharing_?',
       ' water-project-cost-sharing_n', ' water-project-cost-sharing_y',
       ' adoption-of-the-budget-resolution_?',
       ' adoption-of-the-budget-resolution_n',
       ' adoption-of-the-budget-resolution_y', ' physician-fee-freeze_?',
       ' physician-fee-freeze_n', ' physician-fee-freeze_y',
       ' el-salvador-aid_?', ' el-salvador-aid_n', ' el-salvador-aid_y',
       ' religious-groups-in-schools_?', ' religious-groups-in-schools_n',
       ' religious-groups-in-schools_y', ' anti-satellite-test-ban_?',
       ' anti-satellite-test-ban_n', ' anti-satellite-test-ban_y',
       ' aid-to-nicaraguan-contras_?', ' aid-to-nicaraguan-contras_n',
       ' aid-to-nicaraguan-contras_y', ' mx-missile_?', ' mx-missile_n',
       ' mx-missile_y', ' immigration_?', ' immigration_n', ' immigration_y',
       ' synfuels-corporation-cutback_?', ' synfuels-corporation-cutback_n',
       ' synfuels-corporation-cutback_y', ' education-spending_?',
       ' education-spending_n', ' education-spending_y',
       ' superfund-right-to-sue_?', ' superfund-right-to-sue_n',
       ' superfund-right-to-sue_y', ' crime_?', ' crime_n', ' crime_y',
       ' duty-free-exports_?', ' duty-free-exports_n', ' duty-free-exports_y'],
      dtype='object')
```

For example, let us see the picture below:



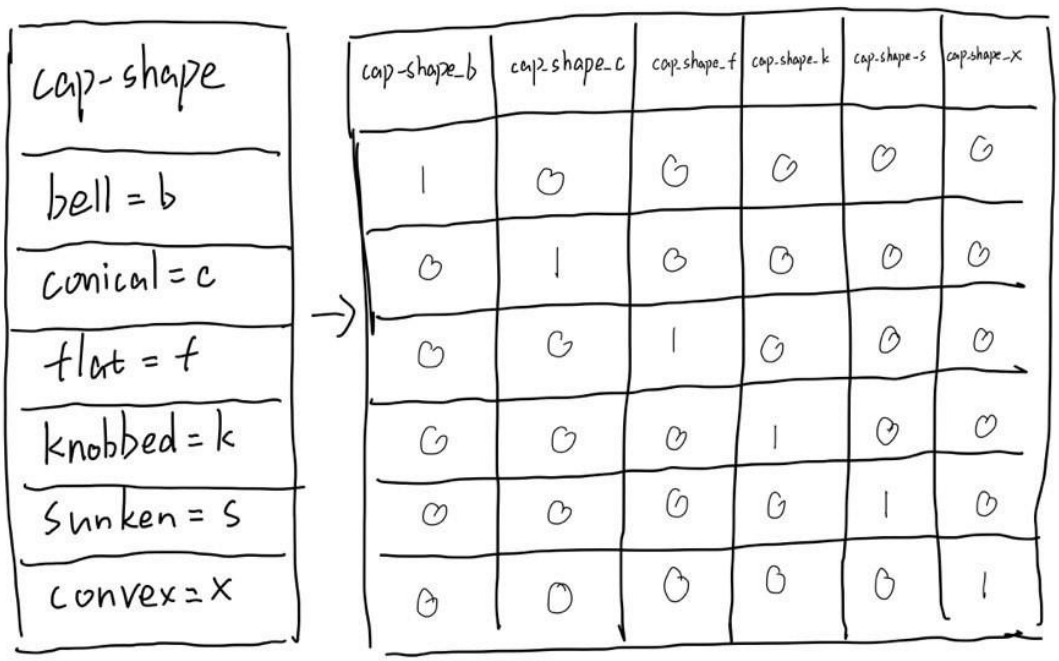
(2)In the dataset of mushroom classification, We have two class: edible=e, poisonous=p and other 22 attributes:

```
Index(['class', 'cap-shape', 'cap-surface', 'cap-color', 'bruises', 'odor',
       'gill-attachment', 'gill-spacing', 'gill-size', 'gill-color',
       'stalk-shape', 'stalk-root', 'stalk-surface-above-ring',
       'stalk-surface-below-ring', 'stalk-color-above-ring',
       'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number',
       'ring-type', 'spore-print-color', 'population', 'habitat'],
      dtype='object')
```

Every attributes are categorical, we should do One-hot encoding:

```
Index(['cap-shape_b', 'cap-shape_c', 'cap-shape_f', 'cap-shape_k',
       'cap-shape_s', 'cap-shape_x', 'cap-surface_f', 'cap-surface_g',
       'cap-surface_s', 'cap-surface_y',
       ...
       'spore-print-color_r', 'spore-print-color_u', 'spore-print-color_w',
       'spore-print-color_y', 'population_a', 'population_c', 'population_n',
       'population_s', 'population_v', 'population_y'],
      dtype='object', length=110)
```

For example, Let us see the picture below:



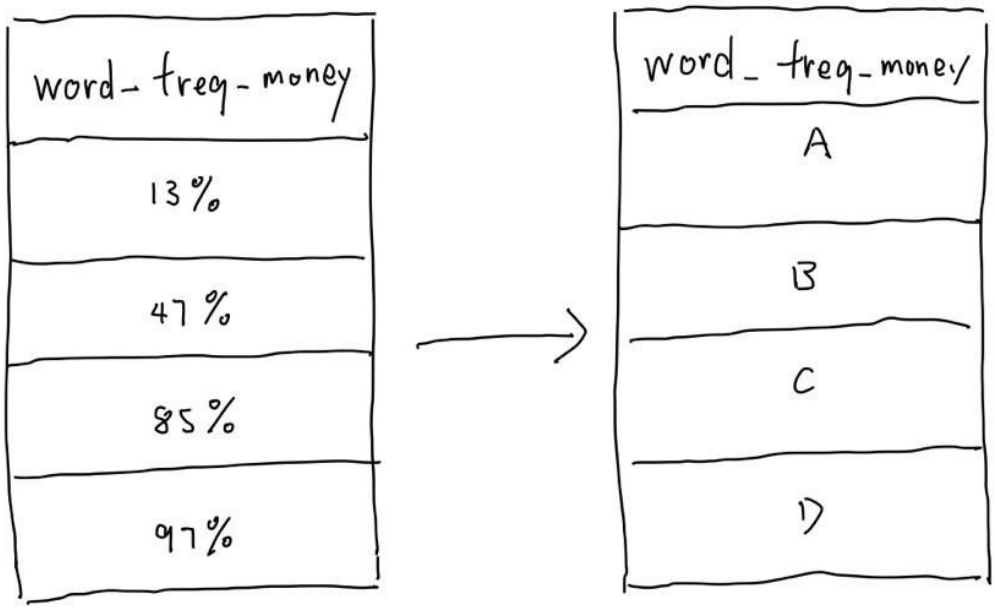
(3) In the dataset of spambase, we have two class: spam (1) or not spam (0) and other 57 attributes:

```
Index(['word_freq_make', 'word_freq_address', 'word_freq_all', 'word_freq_3d',
       'word_freq_our', 'word_freq_over', 'word_freq_remove',
       'word_freq_internet', 'word_freq_order', 'word_freq_mail',
       'word_freq_receive', 'word_freq_will', 'word_freq_people',
       'word_freq_report', 'word_freq_addresses', 'word_freq_free',
       'word_freq_business', 'word_freq_email', 'word_freq_you',
       'word_freq_credit', 'word_freq_your', 'word_freq_font', 'word_freq_000',
       'word_freq_money', 'word_freq_hp', 'word_freq_hpl', 'word_freq_george',
       'word_freq_650', 'word_freq_lab', 'word_freq_labs', 'word_freq_telnet',
       'word_freq_857', 'word_freq_data', 'word_freq_415', 'word_freq_85',
       'word_freq_technology', 'word_freq_1999', 'word_freq_parts',
       'word_freq_pm', 'word_freq_direct', 'word_freq_cs', 'word_freq_meeting',
       'word_freq_original', 'word_freq_project', 'word_freq_re',
       'word_freq_edu', 'word_freq_table', 'word_freq_conference',
       'char_freq_;', 'char_freq_(', 'char_freq_[', 'char_freq_!',
       'char_freq_$', 'char_freq_#', 'capital_run_length_average',
       'capital_run_length_longest', 'capital_run_length_total', 'spam'],
      dtype='object')
```

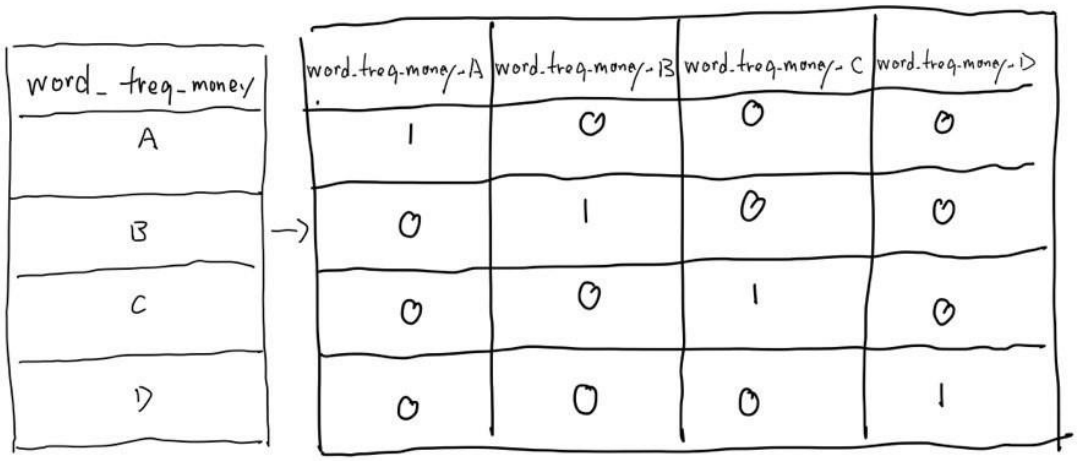
Every attributes are numerical:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	word_freq_	word_freq_	word_freq_	word_freq_	word_freq_	word_freq_	word_freq_	word_freq_	word_freq_	word_freq_	word_freq_	word_freq_	word_freq_
2	0	0.64	0.64	0	0.32	0	0	0	0	0	0	0.64	0
3	0.21	0.28	0.5	0	0.14	0.28	0.21	0.07	0	0.94	0.21	0.79	0.65
4	0.06	0	0.71	0	1.23	0.19	0.19	0.12	0.64	0.25	0.38	0.45	0.12
5	0	0	0	0	0.63	0	0.31	0.63	0.31	0.63	0.31	0.31	0.31
6	0	0	0	0	0.63	0	0.31	0.63	0.31	0.63	0.31	0.31	0.31

Firstly, the entire range of numeric feature values is divided into 4 intervals and the corresponding 4 Categorical features are created. Then we use One-hot encoding.
For example, let us see the picture below:



A 0%-25% B 25%-50% C 50%-75% D 75%-100%




```
C:\Users\30639\PycharmProjects\osda_hw\venv\Scripts\python.exe C:\Users\30639\PycharmProjects\osda_hw\venv\spambase_binarized_data.py
Index(['word_freq_make_D', 'word_freq_address_D', 'word_freq_all_C',
       'word_freq_all_D', 'word_freq_3d_D', 'word_freq_our_C',
       'word_freq_our_D', 'word_freq_over_D', 'word_freq_remove_D',
       'word_freq_internet_D', 'word_freq_order_D', 'word_freq_mail_C',
       'word_freq_mail_D', 'word_freq_receive_D', 'word_freq_will_B',
       'word_freq_will_C', 'word_freq_will_D', 'word_freq_people_D',
       'word_freq_report_D', 'word_freq_addresses_D', 'word_freq_free_C',
       'word_freq_free_D', 'word_freq_business_D', 'word_freq_email_D',
       'word_freq_you_B', 'word_freq_you_C', 'word_freq_you_D',
       'word_freq_credit_D', 'word_freq_your_B', 'word_freq_your_C',
       'word_freq_your_D', 'word_freq_font_D', 'word_freq_000_D',
```

2. Perform classification using standard ML tools:

(1) congressional voting records: naive bayes

Select and Train a Gaussian Naive Bayes Model

+ Code + Markdown

```
clf = GaussianNB()
clf.fit(voting_cat_encoded,voting_labels)
```

```
GaussianNB()
```

```
from sklearn.metrics import accuracy_score

pred = clf.predict(voting_cat_encoded)
accuracy_score(pred,voting_labels)
```

```
0.9655172413793104
```

(2) mushroom classification: decision tree and random forest

Decision Tree Model

+ Code + Markdown

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(random_state = 0 , max_depth = 5)
dt.fit(x_train , y_train)
```

```
DecisionTreeClassifier(max_depth=5, random_state=0)
```

```
predictions = dt.predict(x_test)
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
print(accuracy_score(y_test , predictions))
print(f1_score(y_test , predictions))
```

```
0.9827727645611156
0.9820971867007673
```

Random Forest Model

+ Code + Markdown

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(max_depth = 5)
```

```
rf.fit(x_train , y_train)
```

```
RandomForestClassifier(max_depth=5)
```

+ Code + Markdown

```
predictions = rf.predict(x_test)
```

```
print(accuracy_score(y_test , predictions))
print(f1_score(y_test , predictions))
```

```
0.9922067268252666
0.9917855598789451
```

(3) spambase: k-NN, logistic regression and naive bayes

K-Nearest Neighbor(KNN) Classification Model

+ Code + Markdown

Default metric (Euclidean)

```
k = 5
spam_clf = KNeighborsClassifier(n_neighbors = k)
spam_clf.fit(X, y)
```

```
KNeighborsClassifier()
```

Evaluation Model

```
from sklearn.metrics import accuracy_score, f1_score
train_pred_1 = spam_clf.predict(X)

cm = confusion_matrix(y_true = y, y_pred = train_pred_1)

print(accuracy_score(y_true = y, y_pred = train_pred_1))
print(f1_score(y_true = y, y_pred = train_pred_1))
```

```
0.8680124223602484
0.8292486942547206
```

6. logistic Regression Classifier

```
LogisticRegression = LogisticRegression(solver='liblinear', penalty='l1')
#fit the model with the training data
LogisticRegression.fit(x_train,y_train)
# predict the target(spam or not) on the test dataset
pred = LogisticRegression.predict(x_test)
# Accuracy and f1 Score
accuracy_test_LR = accuracy_score(y_test,pred)
f1_test_LR=f1_score(y_test,pred)
print(accuracy_test_LR)
print(f1_test_LR)
```

```
0.9335088874259381
0.9140425531914894
```

5. Naive Bayes Classifier

+ Code + Markdown

```
GaussNB = GaussianNB()
#fit the model with the training data
GaussNB.fit(x_train,y_train)
# predict the target(spam or not) on the test dataset
predict_test = GaussNB.predict(x_test)
# Accuracy and f1 Score
accuracy_test_NB = accuracy_score(y_test,predict_test)
f1_test_NB=f1_score(y_test,predict_test)
print(accuracy_test_NB)
print(f1_test_NB)
```

```
0.8209348255431205
0.808450704225352
```

3. Write 5 cross-validation procedure to tune the parameters of decision function

At first, let me use an example to introduce the way I worte the 5 cross-validation procedure and tune the parameters of decision function.

Let us see the code of congressional _voting_binarized_data_kfold.py.

```
26  #kFold
27  kf = KFold(n_splits=5, random_state=None)
28  accuracy=[]
29  f1=[]
30  alpha_num=[]
31
32  for c in range(500):
33    a = 0
34    f = 0
35    for i, (train_index, test_index) in enumerate(kf.split(X,y)):...
49
50    accuracy.append(a/5)
51    f1.append(f/5)
52    alpha_num.append(c/1e+4)
53    print("average accuracy:", a/5)
54    print("average f1 score:", f/5)
55
56  max_index, max_number = max(enumerate(accuracy), key=operator.itemgetter(1))
57
58  print("\n")
59  print("the biggest accuracy:",max_number)
60  print("alpha:",alpha_num[max_index])
61
62  plt.title('congressional_voting_binarized')
63  plt.ylabel('accuracy')
64  plt.xlabel('alpha')
65  plt.plot( *args: alpha_num,accuracy)
66  plt.show()
```

We have wrote two for loops.

The out loop is going to iterate through the alpha to find the best accuracy and its alpha.

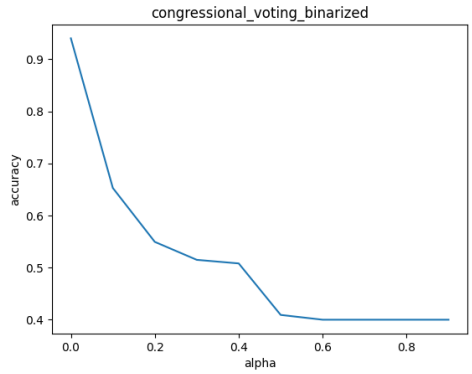
The internal loop is going to do a 5 cross-validation in the current alpha , and we will take the average of the accuracy results to the accuracy[].

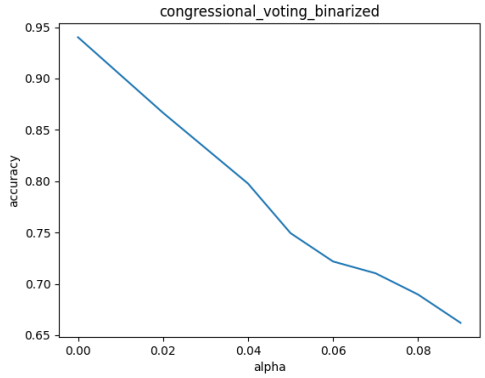
```
35  for i, (train_index, test_index) in enumerate(kf.split(X,y)):
36      print(f"Fold {i}:")
37      X_train=X.iloc[train_index]
38      y_train=y.iloc[train_index]
39      X_test=X.iloc[test_index]
40      y_test = y.iloc[test_index]
41
42      bin_cls = fcalc.classifier.BinarizedBinaryClassifier(X_train.values, y_train.to_numpy(), method="standard-support",alpha=c/1e+4)
43      bin_cls.predict(X_test.values)
44      a+=accuracy_score(y_test, bin_cls.predictions)
45      f+=f1_score(y_test, bin_cls.predictions)
46
47      print(accuracy_score(y_test, bin_cls.predictions))
48      print(f1_score(y_test, bin_cls.predictions))
```

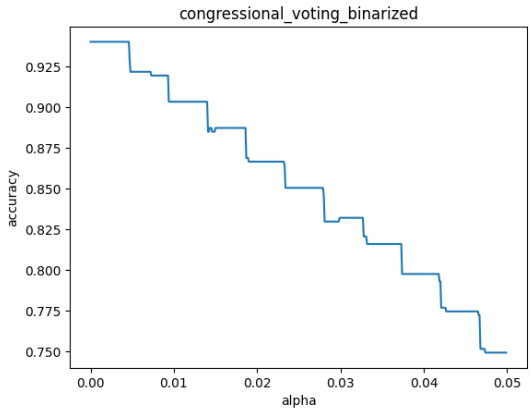
After finishing two loops, we will draw the picture and show the best accuracy and its alpha.

After I introduce the way I wrote the 5 cross_validation procedure and turn the paramenters of decision function. I will not only show the best accuracy and its paramenters, but also show the picture.

(1) congressional voting records
BinarizedBinaryClassifier:





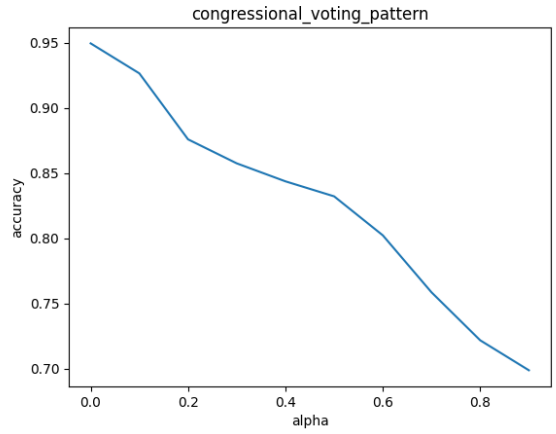


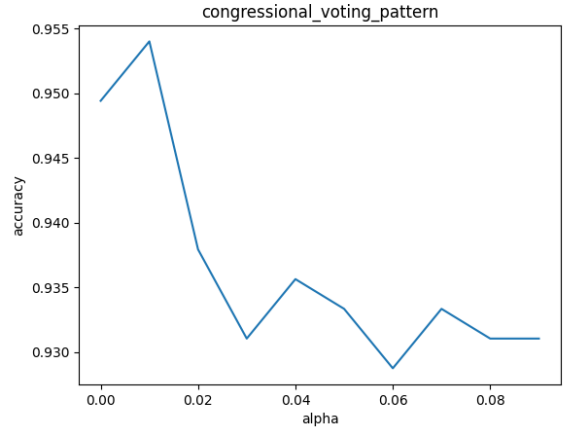
Obvioulsy, we can find that the best accuracy is 0.9402298850574713 and its alpha is 0.0

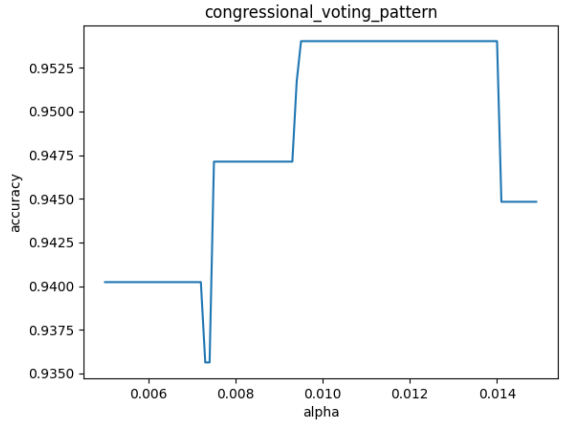
```
the biggest accuracy: 0.9402298850574713
alpha: 0.0

进程已结束，退出代码为 0
```

PatternBinaryClassifier:





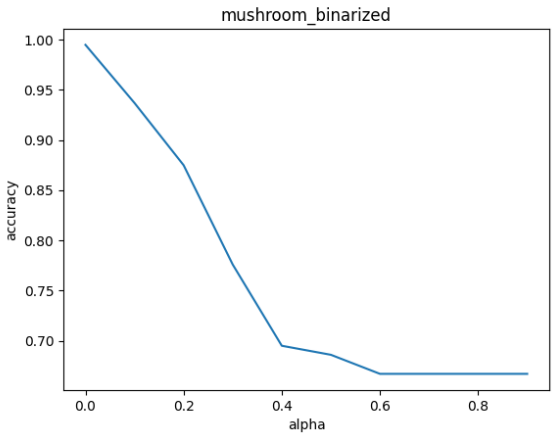


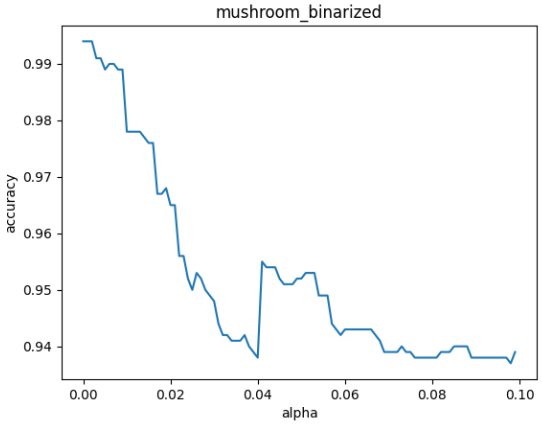
Obvioulsy, we can find that the best accuracy is 0.9540229885057471 and its alpha is 0.0095

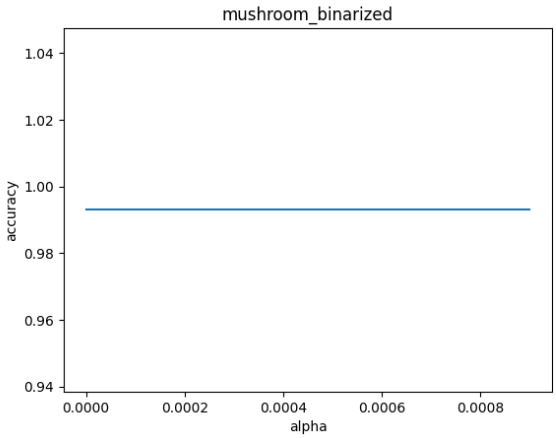
```
the biggest accuracy: 0.9540229885057471
alpha: 0.0095

进程已结束，退出代码为 0
```

(2) mushroom classification
BinarizedBinaryClassifier:





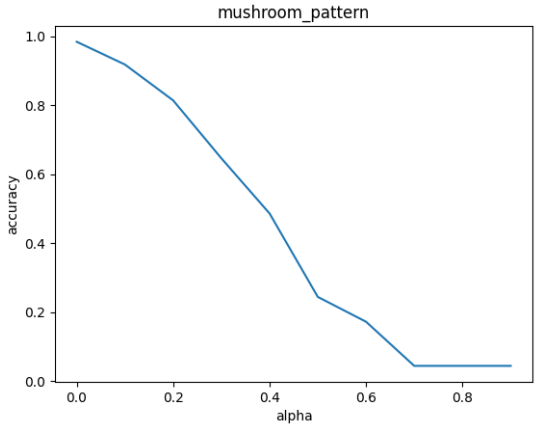


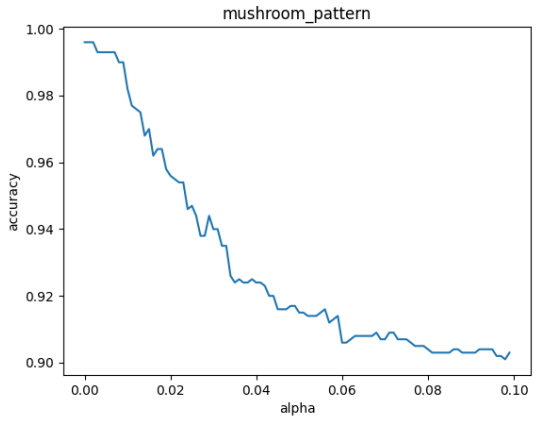
Obvioulsy, we can find that the best accuracy is 0.993 and its alpha is 0.0

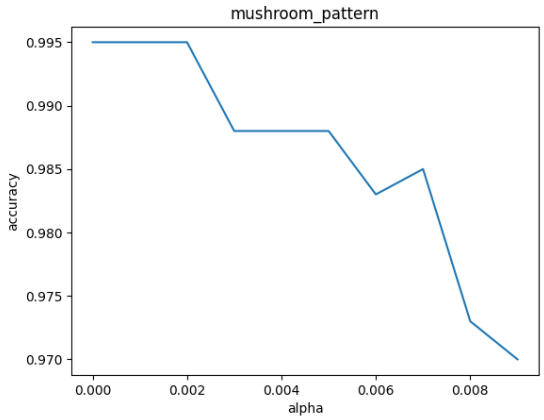
```
the biggest accuracy: 0.993
alpha: 0.0

进程已结束，退出代码为 0
```

PatternBinaryClassifier:





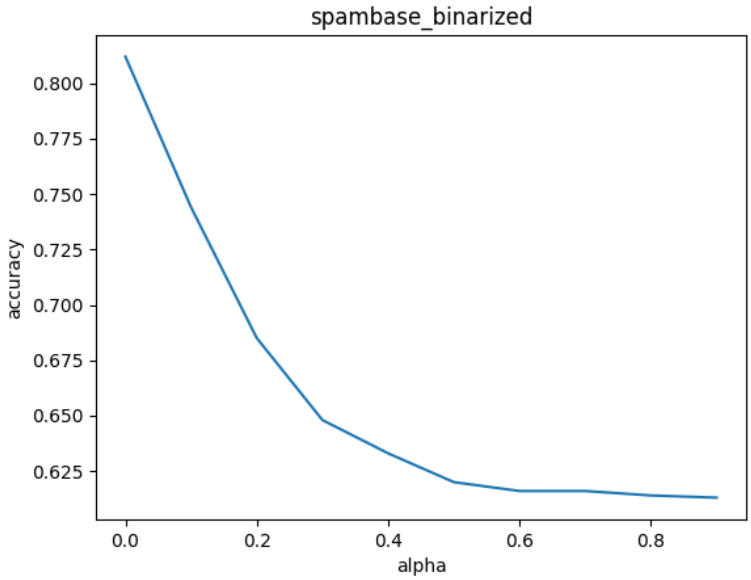


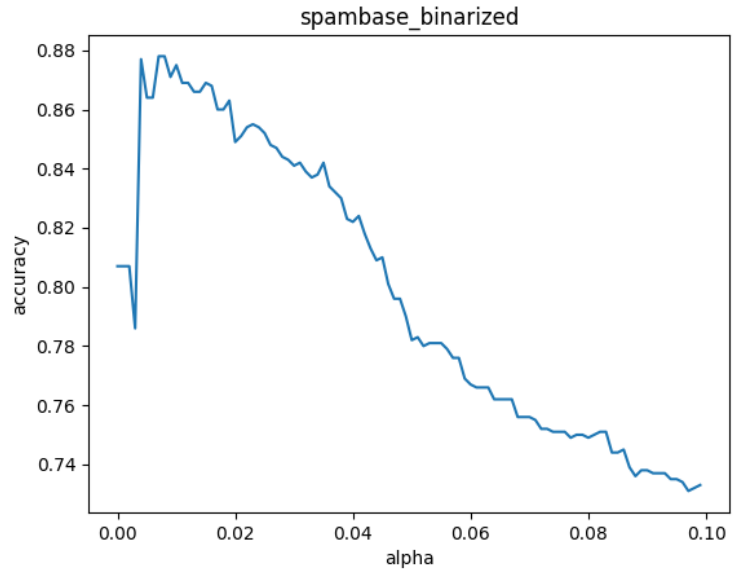
Obvioulsy, we can find that the best accuracy is 0.9949999999999999 and its alpha is 0.0

```
the biggest accuracy: 0.9949999999999999
alpha: 0.0

进程已结束，退出代码为 0
```

(3) spambase:
BinarizedBinaryClassifier:



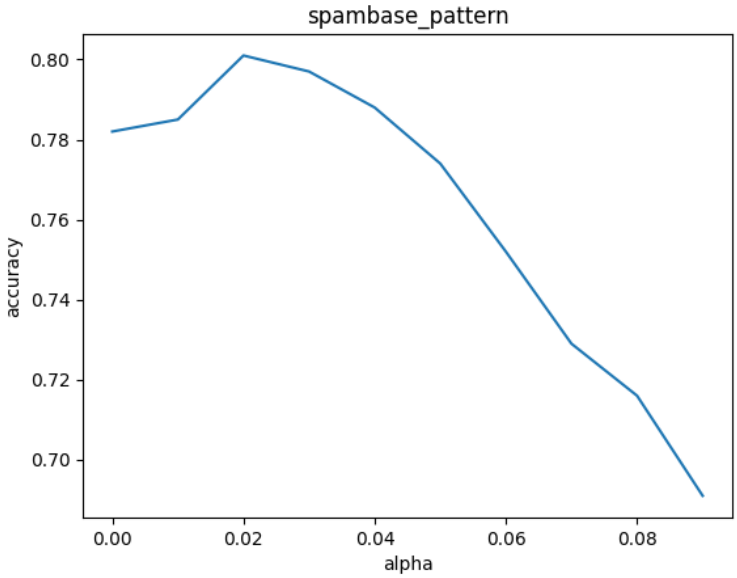


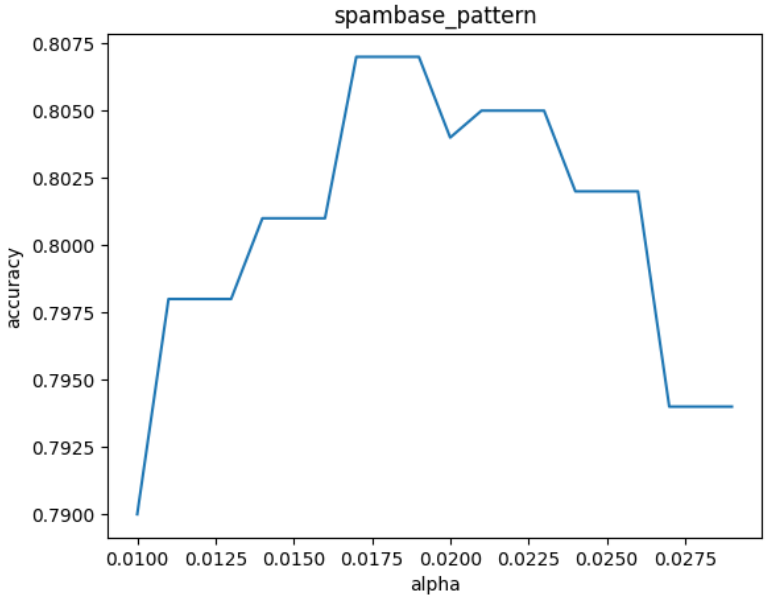
Obvioulsy, we can find that the best accuracy is 0.8779999999999999 and its alpha is 0.007

```
the biggest accuracy: 0.8779999999999999
alpha: 0.007

进程已结束，退出代码为 0
```

PatternBinaryClassifier:





Obvioulsy, we can find that the best accuracy is 0.8069999999999998 and its alpha is 0.017

```
the biggest accuracy: 0.8069999999999998
alpha: 0.017

进程已结束，退出代码为 0
```