assignment5

March 13, 2021

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
```

1 Loading the data

[2]:
```
df = pd.read_csv('/home/hiraditya/Desktop/HomeWork/SJSU/cs156/jupiter/
↪JupyterBooks/homework5_input_data.csv')

df.head()
```

[2]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	…
0	p	x	s	n	t	p	f	…
1	e	x	s	y	t	a	f	…
2	e	b	s	w	t	l	f	…
3	p	x	y	w	t	p	f	…
4	e	x	s	g	f	n	f	…

	gill-spacing	gill-size	gill-color	…	stalk-surface-below-ring	…
0	c	n	k	…	s	…
1	c	b	k	…	s	…
2	c	b	n	…	s	…
3	c	n	n	…	s	…
4	w	b	k	…	s	…

	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	…
0	w	w	p	w	…
1	w	w	p	w	…
2	w	w	p	w	…
3	w	w	p	w	…
4	w	w	p	w	…

	ring-number	ring-type	spore-print-color	population	habitat
0	o	p	k	s	u

```
1          o         p                    n         n       g
2          o         p                    n         n       m
3          o         p                    k         s       u
4          o         e                    n         a       g

[5 rows x 23 columns]
```

[3]:
```
df['cap-color'].value_counts()
```

[3]:
```
n    2284
g    1840
e    1500
y    1072
w    1040
b     168
p     144
c      44
u      16
r      16
Name: cap-color, dtype: int64
```

[4]:
```
df['class']
```

[4]:
```
0       p
1       e
2       e
3       p
4       e
       ..
8119    e
8120    e
8121    e
8122    p
8123    e
Name: class, Length: 8124, dtype: object
```

[5]:
```
df['class'].value_counts()
```

[5]:
```
e    4208
p    3916
Name: class, dtype: int64
```

[7]:
```
#df.info(verbose = True)
```

2 Converting categorical variable to numeric

[8]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
y = le.fit_transform(df['class'])

df = df.drop('class', axis = 1)
X = pd.get_dummies(df, columns = df.columns, prefix = df.columns)
```

[9]:
```
X
```

[9]:

	cap-shape_b	cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	\
0	0	0	0	0	0	
1	0	0	0	0	0	
2	1	0	0	0	0	
3	0	0	0	0	0	
4	0	0	0	0	0	
…	…	…	…	…	…	
8119	0	0	0	1	0	
8120	0	0	0	0	0	
8121	0	0	1	0	0	
8122	0	0	0	1	0	
8123	0	0	0	0	0	

	cap-shape_x	cap-surface_f	cap-surface_g	cap-surface_s	cap-surface_y	\
0	1	0	0	1	0	
1	1	0	0	1	0	
2	0	0	0	1	0	
3	1	0	0	0	1	
4	1	0	0	1	0	
…	…	…	…	…	…	
8119	0	0	0	1	0	
8120	1	0	0	1	0	
8121	0	0	0	1	0	
8122	0	0	0	0	1	
8123	1	0	0	1	0	

	…	population_s	population_v	population_y	habitat_d	habitat_g	\
0	…	1	0	0	0	0	
1	…	0	0	0	0	1	
2	…	0	0	0	0	0	
3	…	1	0	0	0	0	
4	…	0	0	0	0	1	
…	…	…	…	…	…	…	
8119	…	0	0	0	0	0	

```
8120  …              0             1              0          0          0
8121  …              0             0              0          0          0
8122  …              0             1              0          0          0
8123  …              0             0              0          0          0

      habitat_l  habitat_m  habitat_p  habitat_u  habitat_w
0             0          0          0          1          0
1             0          0          0          0          0
2             0          1          0          0          0
3             0          0          0          1          0
4             0          0          0          0          0
…           …          …          …          …          …
8119          1          0          0          0          0
8120          1          0          0          0          0
8121          1          0          0          0          0
8122          1          0          0          0          0
8123          1          0          0          0          0

[8124 rows x 117 columns]
```

3 Breaking the data into the training and test datasets

[10]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

4 Training a decision tree model (DecisionTreeClassifier) and Reporting 5-fold cross-validation accuracies

[11]:
```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()
```

[12]:
```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(clf, X_train, y_train, cv=5)
print(" Individual cross-validation accuracies:", scores,"\n Mean␣
 ↪Cross-Validation Accuracies:", np.mean(scores))
```

```
Individual cross-validation accuracies: [1. 1. 1. 1. 1.]
Mean Cross-Validation Accuracies: 1.0
```

5 Training a decision tree model on all the training data

[13]:
```
from sklearn.metrics import accuracy_score

clf.fit(X_train, y_train)
preds = clf.predict(X_test)

# accuracy
accuracy_score(preds, y_test)
```

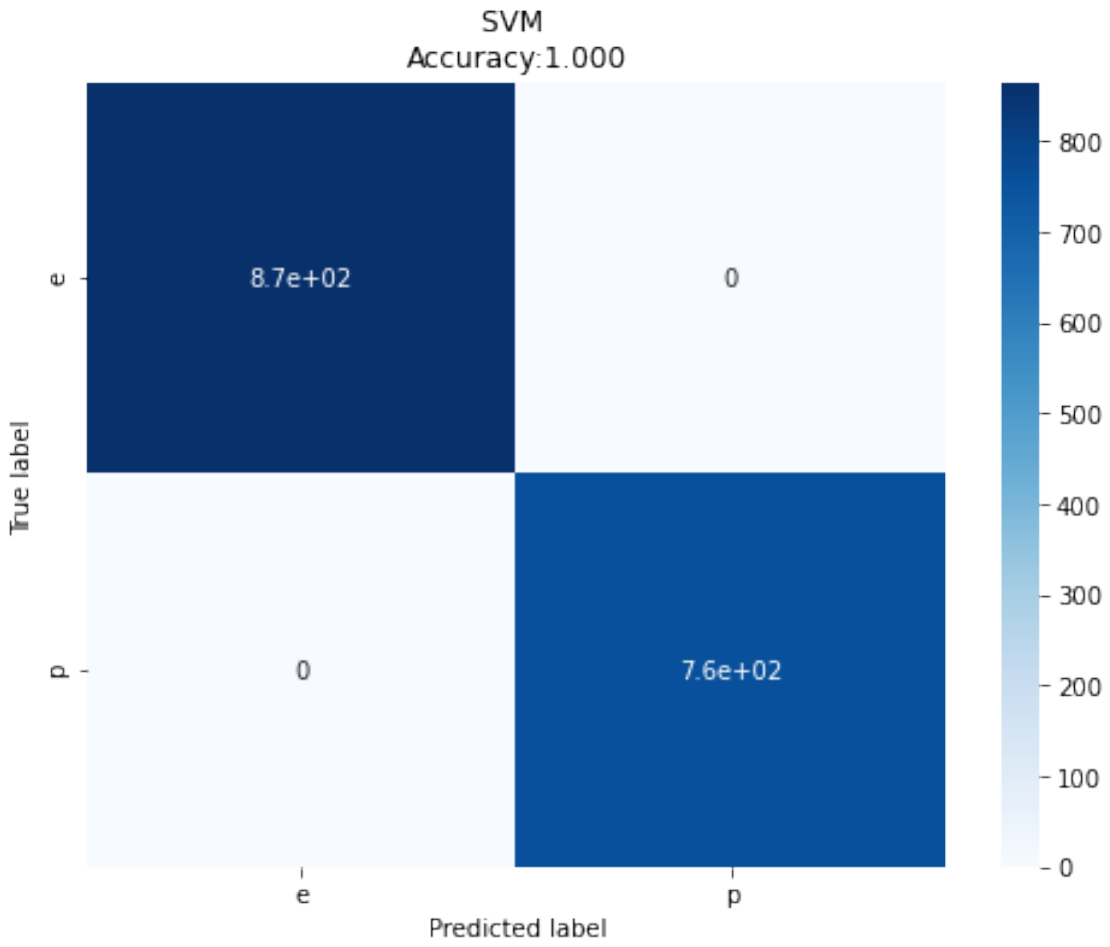
[13]:
```
1.0
```

6 confusion matrix (non-normalized)

[14]:
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, preds)
cm_df = pd.DataFrame(cm, index = le.classes_, columns = le.classes_)

plt.figure(figsize=(8, 6))
sb.heatmap(cm_df, annot=True, cmap=plt.cm.Blues)
plt.title('SVM \nAccuracy:{0:.3f}'.format(accuracy_score(y_test, preds)))
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```



[15]:
```
# StandardScaler =  (actual - mean) / standard dev
```

[ ]:
```

```

7 confusion matrix (normalized)

[17]:
```
from sklearn.preprocessing import StandardScaler # preprocessing stuff
scaler = StandardScaler()

X_norm = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_norm, y, test_size = 0.2)

clf.fit(X_train, y_train)
preds = clf.predict(X_test)
```

```
# accuracy
accuracy_score(preds, y_test)
```

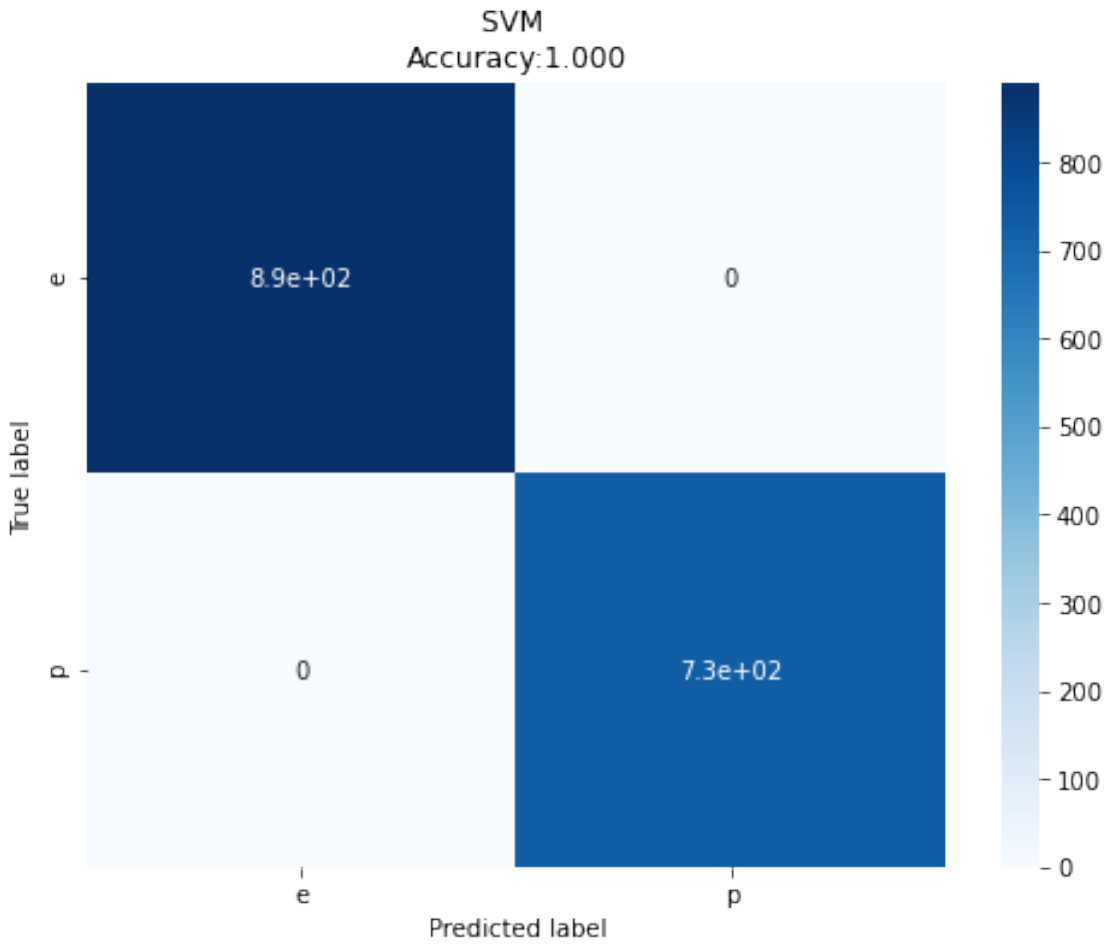
[17]:
```
1.0
```

[18]:
```
X_norm
```

[18]:
```
array([[-0.24272523, -0.02219484, -0.79620985, …, -0.40484176,
         4.59086996, -0.15558197],
       [-0.24272523, -0.02219484, -0.79620985, …, -0.40484176,
        -0.21782364, -0.15558197],
       [ 4.11988487, -0.02219484, -0.79620985, …, -0.40484176,
        -0.21782364, -0.15558197],
       …,
       [-0.24272523, -0.02219484,  1.2559503 , …, -0.40484176,
        -0.21782364, -0.15558197],
       [-0.24272523, -0.02219484, -0.79620985, …, -0.40484176,
        -0.21782364, -0.15558197],
       [-0.24272523, -0.02219484, -0.79620985, …, -0.40484176,
        -0.21782364, -0.15558197]])
```

[19]:
```
cm = confusion_matrix(y_test, preds)
cm_df = pd.DataFrame(cm, index = le.classes_, columns = le.classes_)

plt.figure(figsize=(8, 6))
sb.heatmap(cm_df, annot=True, cmap=plt.cm.Blues)
plt.title('SVM \nAccuracy:{0:.3f}'.format(accuracy_score(y_test, preds)))
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```


SVM
Accuracy:1.000
8.9e+02
0
0
7.3e+02
e
p
True label
e
p
Predicted label
800
700
600
500
400
300
200
100
0

[ ]:

[ ]: