Mushroom Classification

CSc 44700 – Introduction to Machine Learning

By Yevheniya Solomyana

- Predict based on the mushroom features if it is edible or poisonous
- https://www.kaggle.com/datasets/uciml/mushroom-classification

Mushroom Classification

Safe to eat or deadly poison?

Data

- 8124 rows x 23 columns
- Logistic Regression
- Support Vector Machine
- Random Forest
- Needs preprocessing

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...
0	p	x	s	n	t	p	f	c	n	k	...
1	e	x	s	y	t	a	f	c	b	k	...
2	e	b	s	w	t	l	f	c	b	n	...
3	p	x	y	w	t	p	f	c	n	n	...
4	e	x	s	g	f	n	f	w	b	k	...
...	...	...	...	...	...	...	...	...	...	...	...
8119	e	k	s	n	f	n	a	c	b	y	...
8120	e	x	s	n	f	n	a	c	b	y	...
8121	e	f	s	n	f	n	a	c	b	n	...
8122	p	k	y	n	f	y	f	c	n	b	...
8123	e	x	s	n	f	n	a	c	b	y	...

8124 rows x 23 columns

Libraries Imported

```
import matplotlib.pyplot as plt
import pandas as pd

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

LabelEncoder

```
encoder = LabelEncoder()

for column in range(len(data.columns)):
  data[data.columns[column]] = encoder.fit_transform(data[data.columns[column]])
```

- Transform non–numerical labels to numerical labels

LabelEncoder

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...
0	1	5	2	4	1	6	1	0	1	4	...
1	0	5	2	9	1	0	1	0	0	4	...
2	0	0	2	8	1	3	1	0	0	5	...
3	1	5	3	8	1	6	1	0	1	5	...
4	0	5	2	3	0	5	1	1	0	4	...
...	...	...	...	...	...	...	...	...	...	...	...
8119	0	3	2	4	0	5	0	0	0	11	...
8120	0	5	2	4	0	5	0	0	0	11	...
8121	0	2	2	4	0	5	0	0	0	5	...
8122	1	3	3	4	0	8	1	0	1	0	...
8123	0	5	2	4	0	5	0	0	0	11	...

8124 rows × 23 columns

StandardScaler

```
y = data['class']
X = data.drop('class', axis=1)

scaler = StandardScaler()

X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
```

- Scale each feature to unit variance to increase the accuracy of the models

StandardScaler

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...
0	1.029712	0.140128	-0.198250	1.185917	0.881938	0.162896	-0.438864	1.494683	-0.228998	-1.144806	...
1	1.029712	0.140128	1.765874	1.185917	-1.970316	0.162896	-0.438864	-0.669038	-0.228998	-1.144806	...
2	-2.087047	0.140128	1.373049	1.185917	-0.544189	0.162896	-0.438864	-0.669038	0.053477	-1.144806	...
3	1.029712	0.953270	1.373049	1.185917	0.881938	0.162896	-0.438864	1.494683	0.053477	-1.144806	...
4	1.029712	0.140128	-0.591075	-0.843230	0.406562	0.162896	2.278612	-0.669038	-0.228998	0.873511	...
...	...	...	...	...	...	...	...	...	...	...	...
8119	-0.216992	0.140128	-0.198250	-0.843230	0.406562	-6.138869	-0.438864	-0.669038	1.748325	-1.144806	...
8120	1.029712	0.140128	-0.198250	-0.843230	0.406562	-6.138869	-0.438864	-0.669038	1.748325	-1.144806	...
8121	-0.840343	0.140128	-0.198250	-0.843230	0.406562	-6.138869	-0.438864	-0.669038	0.053477	-1.144806	...
8122	-0.216992	0.953270	-0.198250	-0.843230	1.832689	0.162896	-0.438864	1.494683	-1.358896	0.873511	...
8123	1.029712	0.140128	-0.198250	-0.843230	0.406562	-6.138869	-0.438864	-0.669038	1.748325	-1.144806	...

8124 rows × 22 columns

Select & Train Models

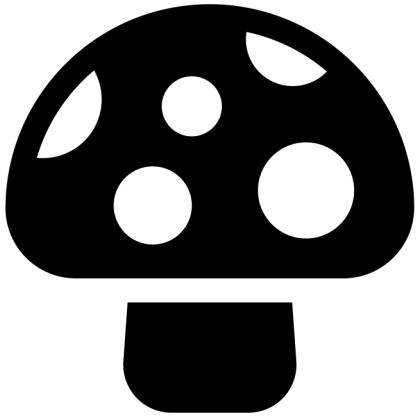
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7)

log_model = LogisticRegression()
svm_model = SVC(kernel='rbf')
rf_model = RandomForestClassifier(n_estimators=500, random_state=42)

log_model.fit(X_train, y_train)
svm_model.fit(X_train, y_train)
rf_model.fit(X_train, y_train)

print(f"Logistic Regression: {log_model.score(X_test, y_test)}")
print(f"Support Vector Machine: {svm_model.score(X_test, y_test)}")
print(f"Random Forest: {rf_model.score(X_test, y_test)}")
```

```
Logistic Regression: 0.9561115668580804
Support Vector Machine: 1.0
Random Forest: 1.0
```



Thank you