experiments

May 4, 2021

In [1]:
```
import numpy as np
import pandas as pd

from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from lightgbm import LGBMClassifier
```

In [2]:
```
raw_data = pd.read_csv('../data/raw/heart.csv')
non_numerical_types = [t for t in raw_data.dtypes if t not in ('int64', 'float64')]
if non_numerical_types:
    print('non-numerical types:', *non_numerical_types)
raw_data
```

Out[2]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
..	...	...	..	...	...	...	...	...	...	...	...	..	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

```
299      1   0     3       0
300      1   2     3       0
301      1   1     3       0
302      1   1     2       0

[303 rows x 14 columns]
```

```
In [3]: X = raw_data.drop('target', axis=1)
        y = raw_data['target']
```

```
In [4]: for C in [10 ** x for x in range(-3, 3)]:
            model = LogisticRegression(C=C, random_state=42, max_iter=2000)
            accuracy = cross_val_score(model, X, y, cv=10)
            print(f'C = {C},  \tval accuracy score: {np.mean(accuracy):.3f}')
```

```
C = 0.001,          val accuracy score: 0.703
C = 0.01,          val accuracy score: 0.766
C = 0.1,          val accuracy score: 0.838
C = 1,          val accuracy score: 0.818
C = 10,          val accuracy score: 0.812
C = 100,          val accuracy score: 0.812
```

```
In [5]: for C in [10 ** x for x in range(-1, 6)]:
            model = SVC(C=C, random_state=42)
            accuracy = cross_val_score(model, X, y, cv=10)
            print(f'C = {C},  \tval accuracy score: {np.mean(accuracy):.3f}')
```

```
C = 0.1,          val accuracy score: 0.545
C = 1,          val accuracy score: 0.660
C = 10,          val accuracy score: 0.690
C = 100,          val accuracy score: 0.742
C = 1000,          val accuracy score: 0.831
C = 10000,          val accuracy score: 0.785
C = 100000,          val accuracy score: 0.755
```

```
In [6]: model = GaussianNB()
        accuracy = cross_val_score(model, X, y, cv=10)
        print(f'val accuracy score: {np.mean(accuracy):.3f}')
```

```
val accuracy score: 0.805
```

```
In [7]: for learning_rate in [0.02, 0.05, 0.1]:
            for n_estimators in range(10, 71, 10):
                model = LGBMClassifier(learning_rate=learning_rate, n_estimators=n_estimators)
                accuracy = cross_val_score(model, X, y, cv=10)
                print(f'learning_rate = {learning_rate}, n_estimators = {n_estimators}:\n'
                      f'\tval accuracy score: {np.mean(accuracy):.3f}')
            print()
```

```
learning_rate = 0.02, n_estimators = 10:
        val accuracy score: 0.821
learning_rate = 0.02, n_estimators = 20:
        val accuracy score: 0.825
learning_rate = 0.02, n_estimators = 30:
        val accuracy score: 0.821
learning_rate = 0.02, n_estimators = 40:
        val accuracy score: 0.821
learning_rate = 0.02, n_estimators = 50:
        val accuracy score: 0.825
learning_rate = 0.02, n_estimators = 60:
        val accuracy score: 0.821
learning_rate = 0.02, n_estimators = 70:
        val accuracy score: 0.818

learning_rate = 0.05, n_estimators = 10:
        val accuracy score: 0.818
learning_rate = 0.05, n_estimators = 20:
        val accuracy score: 0.825
learning_rate = 0.05, n_estimators = 30:
        val accuracy score: 0.822
learning_rate = 0.05, n_estimators = 40:
        val accuracy score: 0.828
learning_rate = 0.05, n_estimators = 50:
        val accuracy score: 0.835
learning_rate = 0.05, n_estimators = 60:
        val accuracy score: 0.831
learning_rate = 0.05, n_estimators = 70:
        val accuracy score: 0.831

learning_rate = 0.1, n_estimators = 10:
        val accuracy score: 0.825
learning_rate = 0.1, n_estimators = 20:
        val accuracy score: 0.815
learning_rate = 0.1, n_estimators = 30:
        val accuracy score: 0.831
learning_rate = 0.1, n_estimators = 40:
        val accuracy score: 0.832
learning_rate = 0.1, n_estimators = 50:
        val accuracy score: 0.815
learning_rate = 0.1, n_estimators = 60:
        val accuracy score: 0.802
learning_rate = 0.1, n_estimators = 70:
        val accuracy score: 0.788
```

```
In [10]: for C in [0.04, 0.08, 0.1, 0.2]:
```

```
            model = LogisticRegression(C=C, random_state=42, max_iter=2000)
            accuracy = cross_val_score(model, X, y, cv=10)
            print(f'C = {C},  \tval accuracy score: {np.mean(accuracy):.3f}')
```

```
C = 0.04,          val accuracy score: 0.835
C = 0.08,          val accuracy score: 0.832
C = 0.1,          val accuracy score: 0.838
C = 0.2,          val accuracy score: 0.832
```

In [17]:
```
best_model = LogisticRegression(C=0.1, random_state=42, max_iter=1000)
np.mean(cross_val_score(best_model, X, y, cv=7))
```

Out[17]:
```
0.8514799154334037
```

In [ ]: