Siddharth Patel

19BCP130

```
import numpy as np
import pandas as pd

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

import matplotlib.pyplot as plt
```

```
dataset = pd.read_csv("./train.csv")
X = dataset.drop(['label'], axis=1).values
y = dataset['label'].values
dataset
```

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pix
0	1	0	0	0	0	0	0	0	0	0	...	
1	0	0	0	0	0	0	0	0	0	0	...	
2	1	0	0	0	0	0	0	0	0	0	...	
3	4	0	0	0	0	0	0	0	0	0	...	
4	0	0	0	0	0	0	0	0	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
41995	0	0	0	0	0	0	0	0	0	0	...	
41996	1	0	0	0	0	0	0	0	0	0	...	
41997	7	0	0	0	0	0	0	0	0	0	...	
41998	6	0	0	0	0	0	0	0	0	0	...	
41999	9	0	0	0	0	0	0	0	0	0	...	

42000 rows × 785 columns

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)

X_train_scaled
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
```

```
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

```
import warnings
warnings.filterwarnings("ignore")

parameters = {'C': [.001, .01, .1, 1, 10, 100]}

grid = GridSearchCV(estimator=LogisticRegression(), param_grid=parameters, cv=4, verbose=2, return_t
grid.fit(X_train_scaled, y_train)
```

```
Fitting 4 folds for each of 6 candidates, totalling 24 fits
[CV] END .....................................C=0.001; total time=  11.4s
[CV] END .....................................C=0.001; total time=  11.7s
[CV] END .....................................C=0.001; total time=  11.5s
[CV] END .....................................C=0.001; total time=  15.1s
[CV] END ......................................C=0.01; total time=   9.9s
[CV] END ......................................C=0.01; total time=   9.7s
[CV] END ......................................C=0.01; total time=   9.8s
[CV] END ......................................C=0.01; total time=   9.6s
[CV] END .......................................C=0.1; total time=   9.7s
[CV] END .......................................C=0.1; total time=   9.6s
[CV] END .......................................C=0.1; total time=   9.4s
[CV] END .......................................C=0.1; total time=   9.6s
[CV] END .........................................C=1; total time=   9.9s
[CV] END .........................................C=1; total time=   9.5s
[CV] END .........................................C=1; total time=   9.6s
[CV] END .........................................C=1; total time=   9.5s
[CV] END ........................................C=10; total time=   9.6s
[CV] END ........................................C=10; total time=   9.5s
[CV] END ........................................C=10; total time=   9.3s
[CV] END ........................................C=10; total time=   9.3s
[CV] END .......................................C=100; total time=   9.4s
[CV] END .......................................C=100; total time=   9.6s
[CV] END .......................................C=100; total time=   9.5s
[CV] END .......................................C=100; total time=   9.6s
GridSearchCV(cv=4, estimator=LogisticRegression(),
             param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100]},
             return_train_score=True, verbose=2)
```

```
grid.cv_results_
```

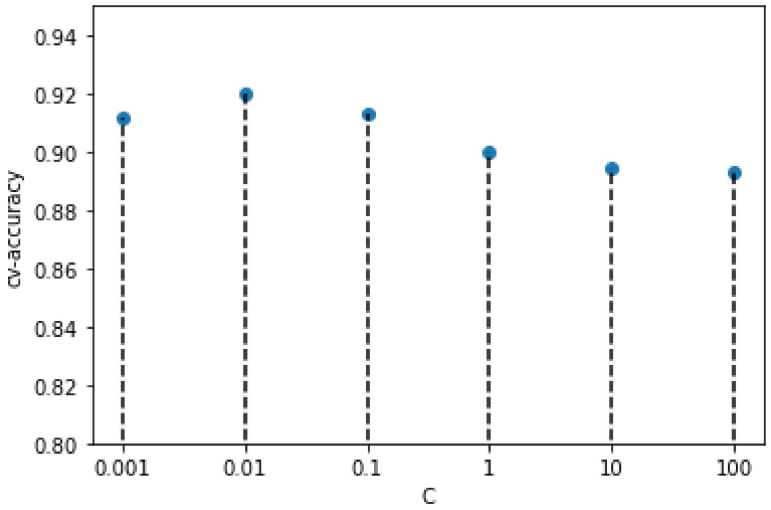
```
{'mean_fit_time': array([12.47271502,  9.81256151,  9.6483801 ,  9.69212663,  9.49873281,
         9.59169298]),
 'std_fit_time': array([1.58217727, 0.13810486, 0.10798535, 0.15484578, 0.09669219,
        0.05857961]),
 'mean_score_time': array([0.03489959, 0.02898633, 0.02822733, 0.02880573, 0.02705884,
        0.0275991 ]),
 'std_score_time': array([0.00275724, 0.00388277, 0.0005116 , 0.00140212, 0.00077676,
        0.00078495]),
 'param_C': masked_array(data=[0.001, 0.01, 0.1, 1, 10, 100],
              mask=[False, False, False, False, False, False],
        fill_value='?',
             dtype=object),
 'params': [{'C': 0.001},
  {'C': 0.01},
  {'C': 0.1},
  {'C': 1},
```

```
 {'C': 10},
 {'C': 100}],
'split0_test_score': array([0.9125    , 0.92202381, 0.915     , 0.90154762, 0.89547619,
       0.89559524]),
'split1_test_score': array([0.90797619, 0.91642857, 0.90928571, 0.89738095, 0.89261905,
       0.89011905]),
'split2_test_score': array([0.91083333, 0.91964286, 0.91404762, 0.89988095, 0.89428571,
       0.89297619]),
'split3_test_score': array([0.91511905, 0.92083333, 0.91357143, 0.90011905, 0.89464286,
       0.89297619]),
'mean_test_score': array([0.91160714, 0.91973214, 0.91297619, 0.89973214, 0.89425595,
       0.89291667]),
'std_test_score': array([0.0025939 , 0.00208482, 0.0021919 , 0.00149966, 0.00103911,
       0.00193704]),
'rank_test_score': array([3, 1, 2, 4, 5, 6]),
'split0_train_score': array([0.92043651, 0.94099206, 0.95535714, 0.96253968, 0.96384921,
       0.96353175]),
'split1_train_score': array([0.92186508, 0.94293651, 0.95805556, 0.96488095, 0.96619048,
       0.96686508]),
'split2_train_score': array([0.92103175, 0.94305556, 0.95746032, 0.96563492, 0.96607143,
       0.96626984]),
'split3_train_score': array([0.91964286, 0.94170635, 0.95619048, 0.96436508, 0.965     ,
       0.96519841]),
'mean_train_score': array([0.92074405, 0.94217262, 0.95676587, 0.96435516, 0.96527778,
       0.96546627]),
'std_train_score': array([0.00081343, 0.0008623 , 0.00105607, 0.0011413 , 0.00094616,
       0.00126654])}
```

```
C_vals = [str(x) for x in parameters['C']]

plt.xlabel("C")
plt.ylabel("cv-accuracy")
plt.scatter(C_vals, grid.cv_results_['mean_test_score'])
plt.vlines(C_vals, 0, grid.cv_results_['mean_test_score'], linestyle="dashed")
plt.ylim(0.80,.95)
plt.xticks(C_vals)
plt.show()

plt.xlabel("C")
plt.ylabel("train-accuracy")
plt.scatter(C_vals, grid.cv_results_['mean_train_score'])
plt.vlines(C_vals, 0, grid.cv_results_['mean_train_score'], linestyle="dashed")
plt.ylim(0.85,1.00)
plt.xticks(C_vals)
plt.show()
```




```
print("Best parameters:", grid.best_params_)
print("Best score:", grid.best_score_)
```

```
model = LogisticRegression(C=.01)

model.fit(X_train_scaled, y_train)
y_pred = model.predict(X_test_scaled)

accuracy_score(y_test, y_pred)
```

```
0.9195238095238095
```

```
for c in parameters['C']:
    model = LogisticRegression(C=c)

    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)

    print(c, accuracy_score(y_test, y_pred))
```

```
0.001 0.9125
0.01 0.9195238095238095
0.1 0.916547619047619
1 0.9089285714285714
10 0.9060714285714285
100 0.9058333333333334
```