Dataset 1

December 9, 2022

[ ]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.utils import resample
from sklearn.preprocessing  import  scale
from sklearn.decomposition  import PCA
from sklearn import metrics
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.model_selection import validation_curve
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt
import matplotlib.colors as colors
import os
```

[ ]:
```
data = pd.read_csv("/home/linux-partition/Desktop/SVM/dataset1.csv") #reading␣
↪the csv files using pandas
```

[ ]:
```
data.head()
```

[ ]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	\
0	1	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	
3	4	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	

	pixel8	…	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	\
0	0	…	0	0	0	0	0	0	
1	0	…	0	0	0	0	0	0	
2	0	…	0	0	0	0	0	0	
3	0	…	0	0	0	0	0	0	
4	0	…	0	0	0	0	0	0	

	pixel780	pixel781	pixel782	pixel783

```
0         0         0         0         0
1         0         0         0         0
2         0         0         0         0
3         0         0         0         0
4         0         0         0         0

[5 rows x 785 columns]
```

[ ]:
```
data.isnull().sum().head
```

[ ]:
```
<bound method NDFrame.head of label       0
pixel0      0
pixel1      0
pixel2      0
pixel3      0
           ..
pixel779    0
pixel780    0
pixel781    0
pixel782    0
pixel783    0
Length: 785, dtype: int64>
```

[ ]:
```
data.describe()
```

[ ]:
```
              label   pixel0   pixel1   pixel2   pixel3   pixel4   pixel5  \
count  42000.000000  42000.0  42000.0  42000.0  42000.0  42000.0  42000.0
mean       4.456643      0.0      0.0      0.0      0.0      0.0      0.0
std        2.887730      0.0      0.0      0.0      0.0      0.0      0.0
min        0.000000      0.0      0.0      0.0      0.0      0.0      0.0
25%        2.000000      0.0      0.0      0.0      0.0      0.0      0.0
50%        4.000000      0.0      0.0      0.0      0.0      0.0      0.0
75%        7.000000      0.0      0.0      0.0      0.0      0.0      0.0
max        9.000000      0.0      0.0      0.0      0.0      0.0      0.0

        pixel6   pixel7   pixel8  …      pixel774      pixel775  \
count  42000.0  42000.0  42000.0  …  42000.000000  42000.000000
mean       0.0      0.0      0.0  …      0.219286      0.117095
std        0.0      0.0      0.0  …      6.312890      4.633819
min        0.0      0.0      0.0  …      0.000000      0.000000
25%        0.0      0.0      0.0  …      0.000000      0.000000
50%        0.0      0.0      0.0  …      0.000000      0.000000
75%        0.0      0.0      0.0  …      0.000000      0.000000
max        0.0      0.0      0.0  …    254.000000    254.000000

           pixel776     pixel777      pixel778      pixel779  pixel780  \
count  42000.000000  42000.00000  42000.000000  42000.000000   42000.0
```

```
mean       0.059024      0.02019      0.017238      0.002857       0.0
std        3.274488      1.75987      1.894498      0.414264       0.0
min        0.000000      0.00000      0.000000      0.000000       0.0
25%        0.000000      0.00000      0.000000      0.000000       0.0
50%        0.000000      0.00000      0.000000      0.000000       0.0
75%        0.000000      0.00000      0.000000      0.000000       0.0
max      253.000000    253.00000    254.000000     62.000000       0.0

       pixel781  pixel782  pixel783
count   42000.0   42000.0   42000.0
mean        0.0       0.0       0.0
std         0.0       0.0       0.0
min         0.0       0.0       0.0
25%         0.0       0.0       0.0
50%         0.0       0.0       0.0
75%         0.0       0.0       0.0
max         0.0       0.0       0.0

[8 rows x 785 columns]
```

[ ]:
```
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import Normalizer,normalize
y = data['label']
X = data.drop(columns='label')

scaler = MinMaxScaler(feature_range=(0, 1))
X = scaler.fit_transform(X)
X =pd.DataFrame(X)
```

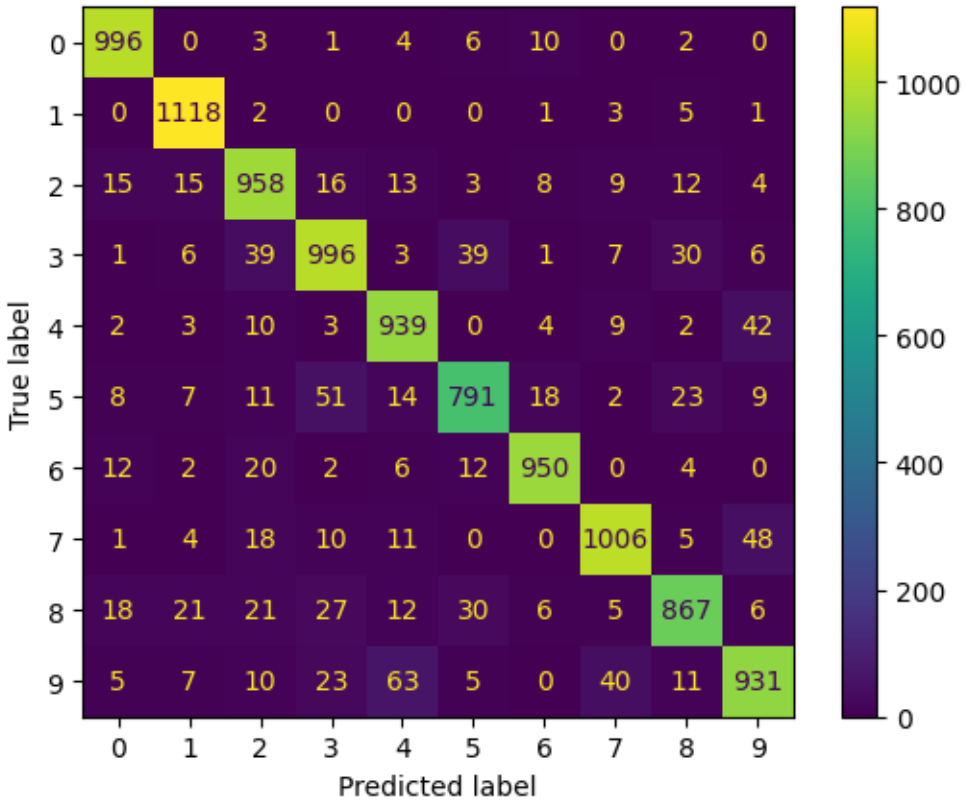
[ ]:
```
X.head()
```

[ ]:
```
   0    1    2    3    4    5    6    7    8    9    …  774  775  776  777  \
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
1  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
2  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
3  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
4  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

   778  779  780  781  782  783
0  0.0  0.0  0.0  0.0  0.0  0.0
1  0.0  0.0  0.0  0.0  0.0  0.0
2  0.0  0.0  0.0  0.0  0.0  0.0
3  0.0  0.0  0.0  0.0  0.0  0.0
4  0.0  0.0  0.0  0.0  0.0  0.0

[5 rows x 784 columns]
```

[ ]:
```
X_scaled = scale(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled,y,random_state=40)
```

[ ]:
```
linear_svm = SVC(kernel='linear')
linear_svm.fit(X_train, y_train)
y_pred_test = linear_svm.predict(X_test)
y_pred_train = linear_svm.predict(X_train)
print("train accuracy: ",metrics.accuracy_score(y_true=y_train,␣
 ↪y_pred=y_pred_train),"\n")
print("test accuracy:", metrics.accuracy_score(y_true=y_test,␣
 ↪y_pred=y_pred_test), "\n")
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_test)
plt.show()
```

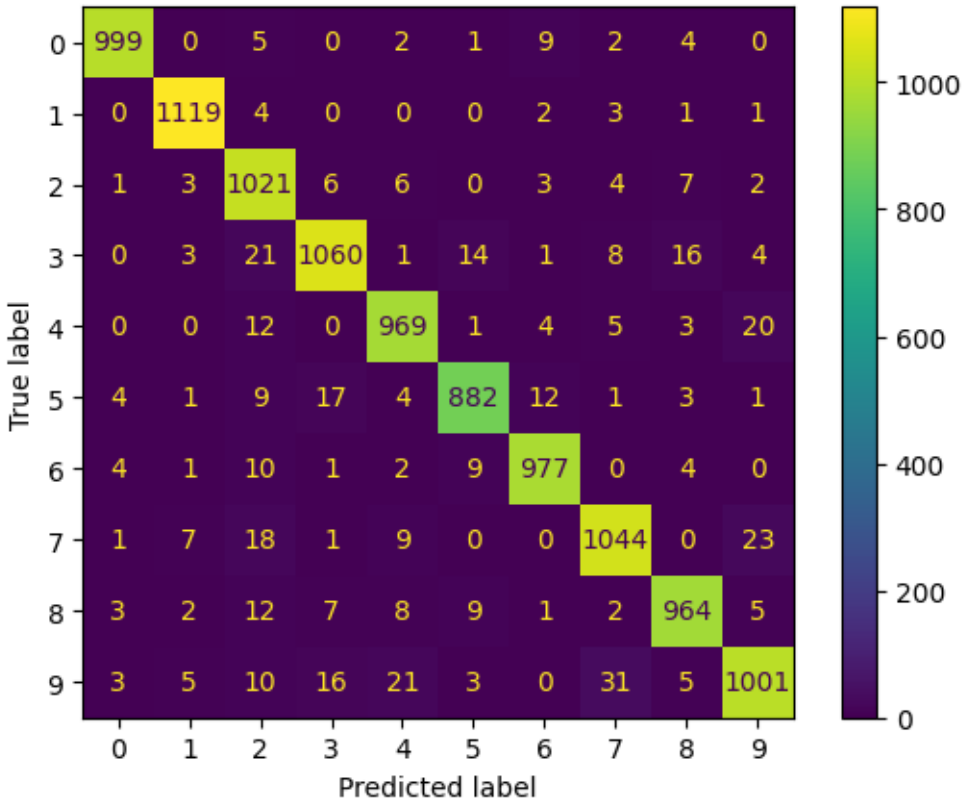
```
train accuracy:  0.9942857142857143

test accuracy: 0.9097142857142857
```



[ ]:
```
rbf_svm = SVC(kernel='rbf')
rbf_svm.fit(X_train, y_train)
```

```
y_pred_test = rbf_svm.predict(X_test)
y_pred_train = rbf_svm.predict(X_train)
print("train accuracy: ",metrics.accuracy_score(y_true=y_train,␣
 ↪y_pred=y_pred_train),"\n")
print(" test accuracy:", metrics.accuracy_score(y_true=y_test,␣
 ↪y_pred=y_pred_test), "\n")
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_test)
plt.show()
```

```
train accuracy:  0.9849523809523809

 test accuracy: 0.9558095238095238
```



[ ]:
```
param_grid = [ {
    'C' :[0.5 , 1 ,5],
    "gamma" : ['scale',0.5,0.001],
    'kernel' :['rbf']},
                ]

model = SVC(kernel="rbf")
```

```
optimal_parameters =GridSearchCV(
    model, param_grid,
    cv=5,
    scoring= 'accuracy',
    verbose=10,
    n_jobs= -1,
    return_train_score=True
)
optimal_parameters.fit(X_train,y_train)

print(optimal_parameters.best_params_)
print(optimal_parameters.best_score_)
```

```
Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV 3/5; 1/9] START C=0.5, gamma=scale, kernel=rbf…
[CV 3/5; 2/9] START C=0.5, gamma=0.5, kernel=rbf…
[CV 4/5; 1/9] START C=0.5, gamma=scale, kernel=rbf…
[CV 2/5; 2/9] START C=0.5, gamma=0.5, kernel=rbf…
[CV 1/5; 1/9] START C=0.5, gamma=scale,
kernel=rbf…[CV 5/5; 2/9] START C=0.5, gamma=0.5,
kernel=rbf…

[CV 1/5; 4/9] START C=1, gamma=scale, kernel=rbf…
[CV 4/5; 2/9] START C=0.5, gamma=0.5, kernel=rbf…
[CV 5/5; 1/9] START C=0.5, gamma=scale, kernel=rbf…
[CV 1/5; 3/9] START C=0.5, gamma=0.001, kernel=rbf…
[CV 4/5; 3/9] START C=0.5, gamma=0.001, kernel=rbf…
[CV 5/5; 3/9] START C=0.5, gamma=0.001, kernel=rbf…
[CV 2/5; 3/9] START C=0.5, gamma=0.001, kernel=rbf…
[CV 3/5; 3/9] START C=0.5, gamma=0.001, kernel=rbf…
[CV 2/5; 1/9] START C=0.5, gamma=scale, kernel=rbf…
[CV 1/5; 2/9] START C=0.5, gamma=0.5, kernel=rbf…
[CV 1/5; 4/9] END C=1, gamma=scale, kernel=rbf;, score=(train=0.985, test=0.954)
total time=18.9min
[CV 2/5; 4/9] START C=1, gamma=scale, kernel=rbf…
[CV 5/5; 3/9] END C=0.5, gamma=0.001, kernel=rbf;, score=(train=0.964,
test=0.946) total time=19.5min
[CV 3/5; 4/9] START C=1, gamma=scale, kernel=rbf…
[CV 1/5; 3/9] END C=0.5, gamma=0.001, kernel=rbf;, score=(train=0.965,
test=0.945) total time=19.7min
[CV 4/5; 4/9] START C=1, gamma=scale, kernel=rbf…
[CV 4/5; 3/9] END C=0.5, gamma=0.001, kernel=rbf;, score=(train=0.964,
test=0.945) total time=19.4min
[CV 5/5; 4/9] START C=1, gamma=scale, kernel=rbf…
[CV 1/5; 1/9] END C=0.5, gamma=scale, kernel=rbf;, score=(train=0.970,
test=0.944) total time=20.4min
[CV 1/5; 5/9] START C=1, gamma=0.5, kernel=rbf…
```

```
[CV 4/5; 1/9] END C=0.5, gamma=scale, kernel=rbf;, score=(train=0.969,
test=0.945) total time=20.4min
[CV 2/5; 5/9] START C=1, gamma=0.5, kernel=rbf…
[CV 2/5; 3/9] END C=0.5, gamma=0.001, kernel=rbf;, score=(train=0.964,
test=0.941) total time=20.2min
[CV 3/5; 5/9] START C=1, gamma=0.5, kernel=rbf…
[CV 2/5; 1/9] END C=0.5, gamma=scale, kernel=rbf;, score=(train=0.970,
test=0.943) total time=21.2min
[CV 4/5; 5/9] START C=1, gamma=0.5, kernel=rbf…
[CV 5/5; 1/9] END C=0.5, gamma=scale, kernel=rbf;, score=(train=0.969,
test=0.945) total time=21.6min
[CV 5/5; 5/9] START C=1, gamma=0.5, kernel=rbf…
[CV 3/5; 1/9] END C=0.5, gamma=scale, kernel=rbf;, score=(train=0.968,
test=0.951) total time=21.0min
[CV 1/5; 6/9] START C=1, gamma=0.001, kernel=rbf…
[CV 3/5; 3/9] END C=0.5, gamma=0.001, kernel=rbf;, score=(train=0.963,
test=0.951) total time=20.8min
[CV 2/5; 6/9] START C=1, gamma=0.001, kernel=rbf…
[CV 2/5; 4/9] END C=1, gamma=scale, kernel=rbf;, score=(train=0.984, test=0.953)
total time=19.0min
[CV 3/5; 6/9] START C=1, gamma=0.001, kernel=rbf…
[CV 4/5; 4/9] END C=1, gamma=scale, kernel=rbf;, score=(train=0.985, test=0.956)
total time=19.0min
[CV 4/5; 6/9] START C=1, gamma=0.001, kernel=rbf…
[CV 5/5; 4/9] END C=1, gamma=scale, kernel=rbf;, score=(train=0.984, test=0.955)
total time=19.0min
[CV 5/5; 6/9] START C=1, gamma=0.001, kernel=rbf…
[CV 3/5; 4/9] END C=1, gamma=scale, kernel=rbf;, score=(train=0.984, test=0.960)
total time=19.1min
[CV 1/5; 7/9] START C=5, gamma=scale, kernel=rbf…
[CV 1/5; 6/9] END C=1, gamma=0.001, kernel=rbf;, score=(train=0.979, test=0.953)
total time=17.8min
[CV 2/5; 7/9] START C=5, gamma=scale, kernel=rbf…
[CV 2/5; 6/9] END C=1, gamma=0.001, kernel=rbf;, score=(train=0.978, test=0.952)
total time=17.7min
[CV 3/5; 7/9] START C=5, gamma=scale, kernel=rbf…
[CV 3/5; 6/9] END C=1, gamma=0.001, kernel=rbf;, score=(train=0.978, test=0.959)
total time=18.6min
[CV 4/5; 7/9] START C=5, gamma=scale, kernel=rbf…
[CV 4/5; 6/9] END C=1, gamma=0.001, kernel=rbf;, score=(train=0.979, test=0.952)
total time=18.0min
[CV 5/5; 7/9] START C=5, gamma=scale, kernel=rbf…
[CV 1/5; 7/9] END C=5, gamma=scale, kernel=rbf;, score=(train=0.998, test=0.964)
total time=18.7min
[CV 1/5; 8/9] START C=5, gamma=0.5, kernel=rbf…
[CV 5/5; 6/9] END C=1, gamma=0.001, kernel=rbf;, score=(train=0.978, test=0.954)
total time=18.2min
[CV 2/5; 8/9] START C=5, gamma=0.5, kernel=rbf…
```

```
[CV 2/5; 7/9] END C=5, gamma=scale, kernel=rbf;, score=(train=0.998, test=0.960)
total time=18.0min
[CV 3/5; 8/9] START C=5, gamma=0.5, kernel=rbf…
[CV 3/5; 7/9] END C=5, gamma=scale, kernel=rbf;, score=(train=0.998, test=0.967)
total time=17.7min
[CV 4/5; 8/9] START C=5, gamma=0.5, kernel=rbf…
[CV 4/5; 7/9] END C=5, gamma=scale, kernel=rbf;, score=(train=0.998, test=0.963)
total time=16.8min
[CV 5/5; 8/9] START C=5, gamma=0.5, kernel=rbf…
[CV 5/5; 7/9] END C=5, gamma=scale, kernel=rbf;, score=(train=0.998, test=0.962)
total time=17.0min
[CV 1/5; 9/9] START C=5, gamma=0.001, kernel=rbf…
[CV 3/5; 2/9] END C=0.5, gamma=0.5, kernel=rbf;, score=(train=0.113, test=0.113)
total time=112.5min
[CV 2/5; 9/9] START C=5, gamma=0.001, kernel=rbf…
[CV 4/5; 2/9] END C=0.5, gamma=0.5, kernel=rbf;, score=(train=0.113, test=0.113)
total time=114.5min
[CV 3/5; 9/9] START C=5, gamma=0.001, kernel=rbf…
[CV 5/5; 2/9] END C=0.5, gamma=0.5, kernel=rbf;, score=(train=0.113, test=0.113)
total time=117.6min
[CV 4/5; 9/9] START C=5, gamma=0.001, kernel=rbf…
[CV 1/5; 2/9] END C=0.5, gamma=0.5, kernel=rbf;, score=(train=0.113, test=0.113)
total time=118.7min
[CV 5/5; 9/9] START C=5, gamma=0.001, kernel=rbf…
[CV 2/5; 2/9] END C=0.5, gamma=0.5, kernel=rbf;, score=(train=0.113, test=0.113)
total time=118.5min
[CV 1/5; 9/9] END C=5, gamma=0.001, kernel=rbf;, score=(train=0.996, test=0.964)
total time=13.6min
[CV 2/5; 9/9] END C=5, gamma=0.001, kernel=rbf;, score=(train=0.996, test=0.962)
total time=13.3min
[CV 3/5; 9/9] END C=5, gamma=0.001, kernel=rbf;, score=(train=0.995, test=0.966)
total time=13.0min
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[5], line 16
      7 model = SVC(kernel="rbf")
      8 optimal_parameters =GridSearchCV(
      9     model, param_grid,
     10     cv=5,
   (…)
     14     return_train_score=True
     15 )
---> 16 optimal_parameters.fit(X_train,y_train)
     18 print(optimal_parameters.best_params_)
     19 print(optimal_parameters.best_score_)
```

```
File ~/.local/lib/python3.10/site-packages/sklearn/model_selection/_search.py:
 ↪875, in BaseSearchCV.fit(self, X, y, groups, **fit_params)
    869     results = self._format_results(
    870         all_candidate_params, n_splits, all_out, all_more_results
    871     )
    873     return results
--> 875 self._run_search(evaluate_candidates)
    877 # multimetric is determined here because in the case of a callable
    878 # self.scoring the return type is only known after calling
    879 first_test_score = all_out[0]["test_scores"]

File ~/.local/lib/python3.10/site-packages/sklearn/model_selection/_search.py:
 ↪1379, in GridSearchCV._run_search(self, evaluate_candidates)
   1377 def _run_search(self, evaluate_candidates):
   1378     """Search all candidates in param_grid"""
-> 1379     evaluate_candidates(ParameterGrid(self.param_grid))

File ~/.local/lib/python3.10/site-packages/sklearn/model_selection/_search.py:
 ↪822, in BaseSearchCV.fit.<locals>.evaluate_candidates(candidate_params, cv,␣
 ↪more_results)
    814 if self.verbose > 0:
    815     print(
    816         "Fitting {0} folds for each of {1} candidates,"
    817         " totalling {2} fits".format(
    818             n_splits, n_candidates, n_candidates * n_splits
    819         )
    820     )
--> 822 out = parallel(
    823     delayed(_fit_and_score)(
    824         clone(base_estimator),
    825         X,
    826         y,
    827         train=train,
    828         test=test,
    829         parameters=parameters,
    830         split_progress=(split_idx, n_splits),
    831         candidate_progress=(cand_idx, n_candidates),
    832         **fit_and_score_kwargs,
    833     )
    834     for (cand_idx, parameters), (split_idx, (train, test)) in product(
    835         enumerate(candidate_params), enumerate(cv.split(X, y, groups))
    836     )
    837 )
    839 if len(out) < 1:
    840     raise ValueError(
    841         "No fits were performed. "
    842         "Was the CV iterator empty? "
    843         "Were there no candidates?"
```

```
    844     )

File ~/.local/lib/python3.10/site-packages/joblib/parallel.py:1098, in Parallel
 ↪__call__(self, iterable)
   1095     self._iterating = False
   1097 with self._backend.retrieval_context():
-> 1098     self.retrieve()
   1099 # Make sure that we get a last message telling us we are done
   1100 elapsed_time = time.time() - self._start_time

File ~/.local/lib/python3.10/site-packages/joblib/parallel.py:975, in Parallel.
 ↪retrieve(self)
    973 try:
    974     if getattr(self._backend, 'supports_timeout', False):
--> 975         self._output.extend(job.get(timeout=self.timeout))
    976     else:
    977         self._output.extend(job.get())

File ~/.local/lib/python3.10/site-packages/joblib/_parallel_backends.py:567, in
 ↪LokyBackend.wrap_future_result(future, timeout)
    564 """Wrapper for Future.result to implement the same behaviour as
    565 AsyncResults.get from multiprocessing."""
    566 try:
--> 567     return future.result(timeout=timeout)
    568 except CfTimeoutError as e:
    569     raise TimeoutError from e

File /usr/lib/python3.10/concurrent/futures/_base.py:453, in Future.result(self
 ↪timeout)
    450 elif self._state == FINISHED:
    451     return self.__get_result()
--> 453 self._condition.wait(timeout)
    455 if self._state in [CANCELLED, CANCELLED_AND_NOTIFIED]:
    456     raise CancelledError()

File /usr/lib/python3.10/threading.py:320, in Condition.wait(self, timeout)
    318 try:    # restore state no matter what (e.g., KeyboardInterrupt)
    319     if timeout is None:
--> 320         waiter.acquire()
    321         gotit = True
    322     else:

KeyboardInterrupt:
```

[ ]:
```
cv_results = pd.DataFrame(optimal_parameters.cv_results_)

# converting C to numeric type for plotting on x-axis
```

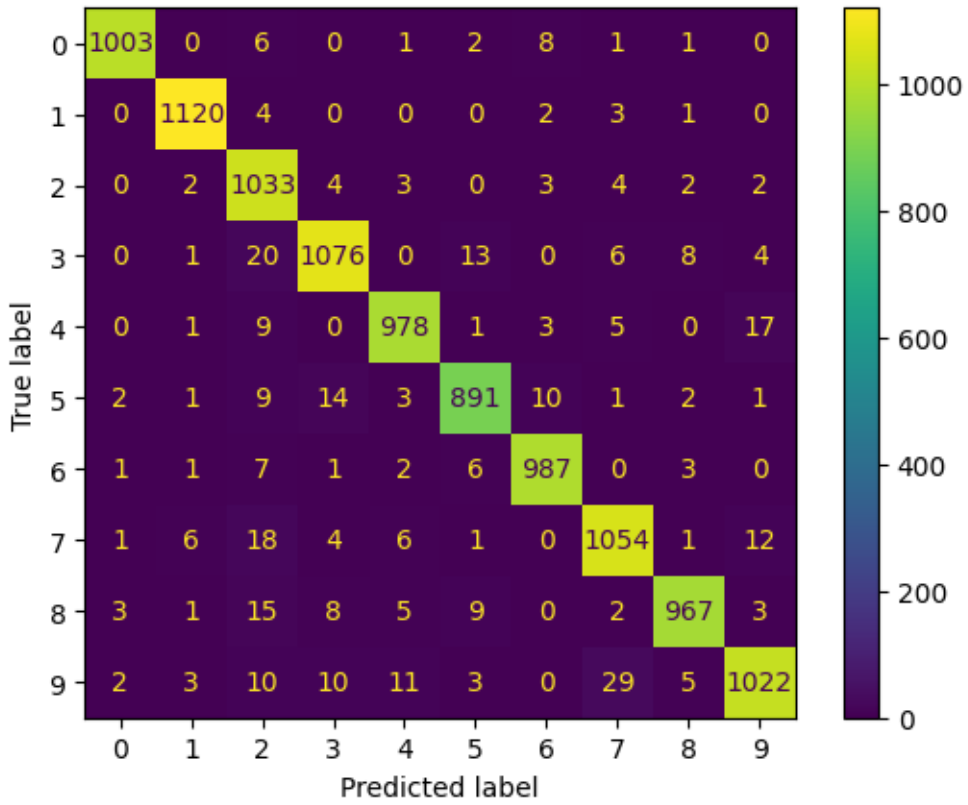
```
cv_results['param_C'] = cv_results['param_C'].astype('int')

# # plotting
plt.figure(figsize=(16,8))

# subplot 1/3
plt.subplot(131)
gamma_scale = cv_results[cv_results['param_gamma']=='scale']

plt.plot(gamma_scale["param_C"], gamma_scale["mean_test_score"])
plt.plot(gamma_scale["param_C"], gamma_scale["mean_train_score"])
plt.xlabel('C')
plt.ylabel('Accuracy')
plt.title("Gamma=scale")
plt.ylim([0.40, 1])
plt.legend(['test accuracy', 'train accuracy'], loc='upper left')
plt.xscale('log')

# subplot 2/3
plt.subplot(132)
gamma_001 = cv_results[cv_results['param_gamma']==0.5]

plt.plot(gamma_001["param_C"], gamma_001["mean_test_score"])
plt.plot(gamma_001["param_C"], gamma_001["mean_train_score"])
plt.xlabel('C')
plt.ylabel('Accuracy')
plt.title("Gamma=0.5")
plt.ylim([0.40, 1])
plt.legend(['test accuracy', 'train accuracy'], loc='upper left')
plt.xscale('log')

# subplot 3/3
plt.subplot(133)
gamma_5 = cv_results[cv_results['param_gamma']==0.001]

plt.plot(gamma_5["param_C"], gamma_5["mean_test_score"])
plt.plot(gamma_5["param_C"], gamma_5["mean_train_score"])
plt.xlabel('C')
plt.ylabel('Accuracy')
plt.title("Gamma=0.001")
plt.ylim([0.40, 1])
plt.legend(['test accuracy', 'train accuracy'], loc='upper left')
plt.xscale('log')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[26], line 1
```

```
----> 1 cv_results = pd.DataFrame(optimal_parameters.cv_results_)
      3 # converting C to numeric type for plotting on x-axis
      4 cv_results['param_C'] = cv_results['param_C'].astype('int')

AttributeError: 'GridSearchCV' object has no attribute 'cv_results_'
```

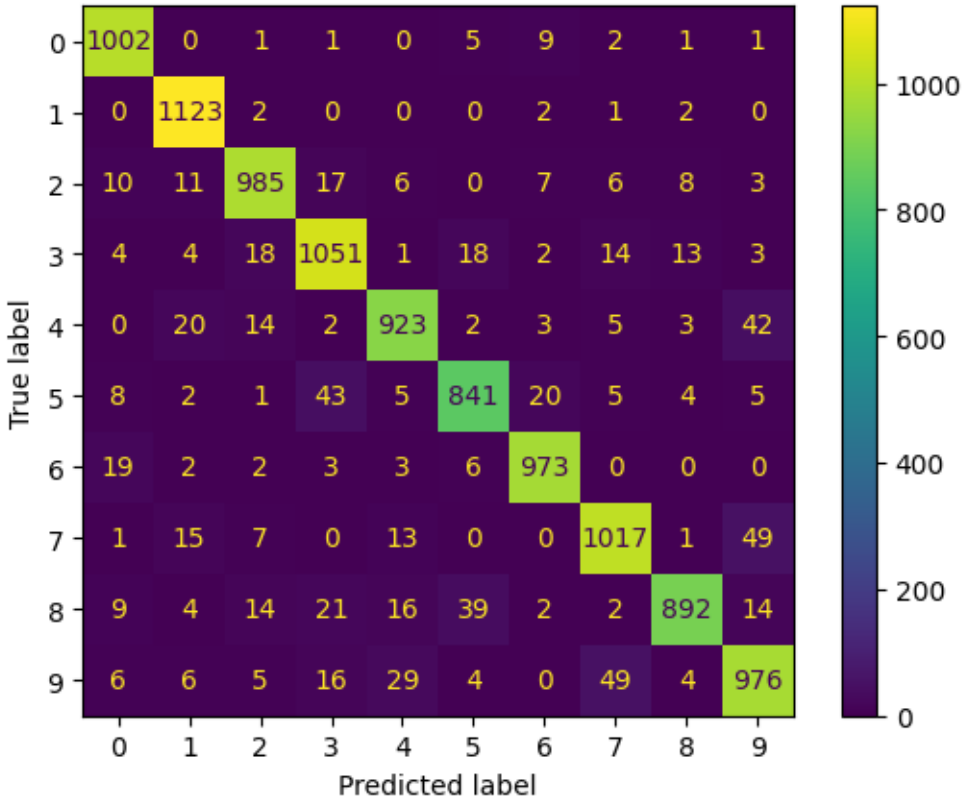
[ ]:
```
rbf_svm_better = SVC(C=5, gamma='scale',kernel='rbf')
rbf_svm_better.fit(X_train, y_train)
y_pred_test = rbf_svm_better.predict(X_test)
y_pred_train = rbf_svm_better.predict(X_train)
print("train accuracy: ",metrics.accuracy_score(y_true=y_train,␣
 ↪y_pred=y_pred_train),"\n")
print("accuracy:", metrics.accuracy_score(y_true=y_test, y_pred=y_pred_test),␣
 ↪"\n")
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_test)
plt.show()
```

```
train accuracy:  0.9978412698412699

accuracy: 0.9648571428571429
```



[ ]:
```
from sklearn.neighbors import KNeighborsClassifier
knn_model =KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train, y_train)
y_pred_test = knn_model.predict(X_test)
y_pred_train = knn_model.predict(X_train)
print("Train accuracy: ",metrics.accuracy_score(y_true=y_train,␣
 ↪y_pred=y_pred_train),"\n")
print("Test accuracy:", metrics.accuracy_score(y_true=y_test,␣
 ↪y_pred=y_pred_test), "\n")
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_test)
plt.show()
```

```
Train accuracy:  0.9674920634920635

Test accuracy: 0.9317142857142857
```



[ ]:
```
model= KNeighborsClassifier()
param_grid = [{'n_neighbors': np.arange(1, 20)}]
optimal_parameters =GridSearchCV(
    model, param_grid,
```

```
    cv=5,
    scoring= 'accuracy',
    verbose=10,
    n_jobs= -1,
    return_train_score=True
)
optimal_parameters.fit(X_train,y_train)

print(optimal_parameters.best_params_)
print(optimal_parameters.best_score_)
```

```
Fitting 5 folds for each of 19 candidates, totalling 95 fits
[CV 1/5; 1/19] START n_neighbors=1…
[CV 3/5; 1/19] START n_neighbors=1…
[CV 2/5; 1/19] START n_neighbors=1…
[CV 3/5; 2/19] START n_neighbors=2…
[CV 2/5; 2/19] START n_neighbors=2…
[CV 4/5; 1/19] START n_neighbors=1…
[CV 1/5; 4/19] START n_neighbors=4…
[CV 3/5; 3/19] START n_neighbors=3…
[CV 2/5; 3/19] START n_neighbors=3…
[CV 1/5; 2/19] START n_neighbors=2…
[CV 5/5; 3/19] START n_neighbors=3…
[CV 5/5; 1/19] START n_neighbors=1…
[CV 1/5; 3/19] START n_neighbors=3…
[CV 5/5; 2/19] START n_neighbors=2…
[CV 4/5; 2/19] START n_neighbors=2…
[CV 4/5; 3/19] START n_neighbors=3…
[CV 1/5; 1/19] END n_neighbors=1;, score=(train=1.000, test=0.934) total time=
22.0s
[CV 2/5; 4/19] START n_neighbors=4…
[CV 2/5; 2/19] END n_neighbors=2;, score=(train=0.965, test=0.920) total time=
21.5s
[CV 3/5; 4/19] START n_neighbors=4…
[CV 3/5; 1/19] END n_neighbors=1;, score=(train=1.000, test=0.935) total time=
22.0s
[CV 4/5; 4/19] START n_neighbors=4…
[CV 1/5; 3/19] END n_neighbors=3;, score=(train=0.966, test=0.934) total time=
21.9s
[CV 3/5; 3/19] END n_neighbors=3;, score=(train=0.964, test=0.939) total time=
22.2s
[CV 5/5; 4/19] START n_neighbors=4…
[CV 1/5; 5/19] START n_neighbors=5…
[CV 4/5; 1/19] END n_neighbors=1;, score=(train=1.000, test=0.933) total time=
21.5s
[CV 3/5; 2/19] END n_neighbors=2;, score=(train=0.965, test=0.925) total time=
21.6s
```

```
[CV 4/5; 3/19] END n_neighbors=3;, score=(train=0.965, test=0.934) total time=
21.3s
[CV 2/5; 5/19] START n_neighbors=5…
[CV 3/5; 5/19] START n_neighbors=5…
[CV 4/5; 5/19] START n_neighbors=5…
[CV 2/5; 1/19] END n_neighbors=1;, score=(train=1.000, test=0.932) total time=
21.9s
[CV 5/5; 5/19] START n_neighbors=5…
[CV 4/5; 2/19] END n_neighbors=2;, score=(train=0.964, test=0.922) total time=
21.5s
[CV 2/5; 3/19] END n_neighbors=3;, score=(train=0.965, test=0.935) total time=
21.9s
[CV 5/5; 3/19] END n_neighbors=3;, score=(train=0.965, test=0.933) total time=
22.7s
[CV 5/5; 2/19] END n_neighbors=2;, score=(train=0.964, test=0.923) total time=
21.3s
[CV 1/5; 4/19] END n_neighbors=4;, score=(train=0.958, test=0.932) total time=
23.5s
[CV 1/5; 6/19] START n_neighbors=6…
[CV 2/5; 6/19] START n_neighbors=6…
[CV 3/5; 6/19] START n_neighbors=6…
[CV 4/5; 6/19] START n_neighbors=6…
[CV 5/5; 6/19] START n_neighbors=6…
[CV 5/5; 1/19] END n_neighbors=1;, score=(train=1.000, test=0.934) total time=
22.0s
[CV 1/5; 7/19] START n_neighbors=7…
[CV 1/5; 2/19] END n_neighbors=2;, score=(train=0.964, test=0.919) total time=
22.2s
[CV 2/5; 7/19] START n_neighbors=7…
[CV 3/5; 4/19] END n_neighbors=4;, score=(train=0.956, test=0.936) total time=
21.6s
[CV 3/5; 7/19] START n_neighbors=7…
[CV 2/5; 4/19] END n_neighbors=4;, score=(train=0.957, test=0.933) total time=
21.4s
[CV 4/5; 7/19] START n_neighbors=7…
[CV 4/5; 6/19] END n_neighbors=6;, score=(train=0.951, test=0.932) total time=
22.8s
[CV 5/5; 7/19] START n_neighbors=7…
[CV 2/5; 5/19] END n_neighbors=5;, score=(train=0.955, test=0.937) total time=
22.1s
[CV 1/5; 8/19] START n_neighbors=8…
[CV 4/5; 4/19] END n_neighbors=4;, score=(train=0.957, test=0.929) total time=
22.6s
[CV 2/5; 8/19] START n_neighbors=8…
[CV 5/5; 5/19] END n_neighbors=5;, score=(train=0.957, test=0.934) total time=
22.3s
[CV 3/5; 8/19] START n_neighbors=8…
[CV 4/5; 5/19] END n_neighbors=5;, score=(train=0.955, test=0.933) total time=
```

```
23.1s
[CV 4/5; 8/19] START n_neighbors=8…
[CV 1/5; 5/19] END n_neighbors=5;, score=(train=0.955, test=0.933) total time=
22.5s
[CV 5/5; 8/19] START n_neighbors=8…
[CV 3/5; 5/19] END n_neighbors=5;, score=(train=0.955, test=0.939) total time=
22.4s
[CV 1/5; 9/19] START n_neighbors=9…
[CV 3/5; 6/19] END n_neighbors=6;, score=(train=0.951, test=0.937) total time=
22.7s
[CV 2/5; 9/19] START n_neighbors=9…
[CV 5/5; 6/19] END n_neighbors=6;, score=(train=0.950, test=0.933) total time=
21.9s
[CV 3/5; 9/19] START n_neighbors=9…
[CV 1/5; 7/19] END n_neighbors=7;, score=(train=0.949, test=0.932) total time=
22.0s
[CV 4/5; 9/19] START n_neighbors=9…
[CV 2/5; 6/19] END n_neighbors=6;, score=(train=0.951, test=0.934) total time=
22.7s
[CV 5/5; 9/19] START n_neighbors=9…
[CV 2/5; 7/19] END n_neighbors=7;, score=(train=0.949, test=0.935) total time=
23.4s
[CV 1/5; 10/19] START n_neighbors=10…
[CV 5/5; 4/19] END n_neighbors=4;, score=(train=0.958, test=0.933) total time=
22.2s
[CV 2/5; 10/19] START n_neighbors=10…
[CV 1/5; 6/19] END n_neighbors=6;, score=(train=0.950, test=0.930) total time=
23.1s
[CV 3/5; 10/19] START n_neighbors=10…
[CV 4/5; 7/19] END n_neighbors=7;, score=(train=0.949, test=0.932) total time=
23.5s
[CV 4/5; 10/19] START n_neighbors=10…
[CV 3/5; 8/19] END n_neighbors=8;, score=(train=0.947, test=0.934) total time=
21.5s
[CV 5/5; 10/19] START n_neighbors=10…
[CV 2/5; 9/19] END n_neighbors=9;, score=(train=0.944, test=0.934) total time=
23.0s
[CV 1/5; 11/19] START n_neighbors=11…
[CV 1/5; 8/19] END n_neighbors=8;, score=(train=0.946, test=0.928) total time=
24.4s
[CV 2/5; 11/19] START n_neighbors=11…
[CV 4/5; 8/19] END n_neighbors=8;, score=(train=0.947, test=0.930) total time=
23.8s
[CV 3/5; 7/19] END n_neighbors=7;, score=(train=0.949, test=0.938) total time=
23.4s
[CV 3/5; 11/19] START n_neighbors=11…
[CV 4/5; 11/19] START n_neighbors=11…
[CV 2/5; 10/19] END n_neighbors=10;, score=(train=0.941, test=0.932) total time=
```

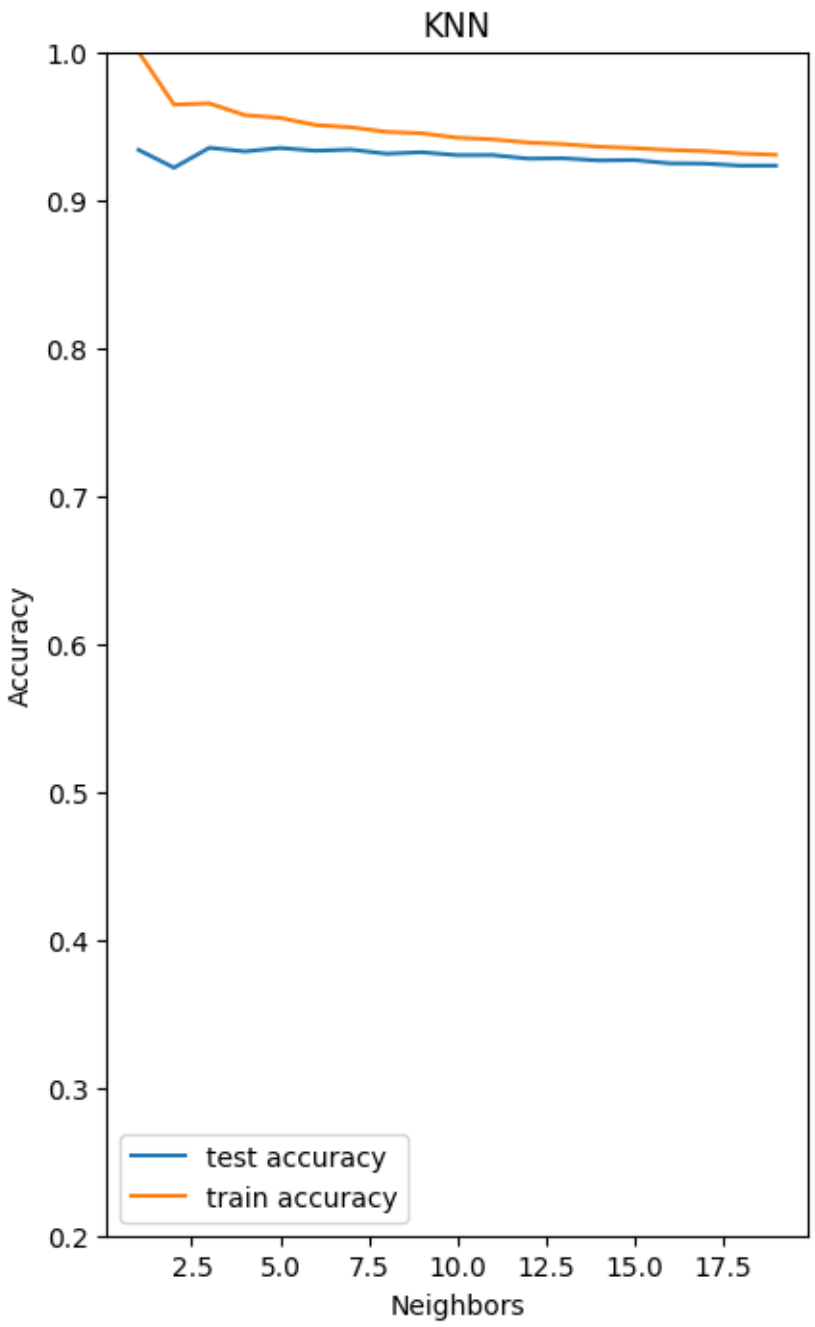
```
22.8s
[CV 5/5; 11/19] START n_neighbors=11…
[CV 5/5; 8/19] END n_neighbors=8;, score=(train=0.946, test=0.930) total time=
24.5s
[CV 1/5; 12/19] START n_neighbors=12…
[CV 5/5; 7/19] END n_neighbors=7;, score=(train=0.949, test=0.933) total time=
22.7s
[CV 2/5; 12/19] START n_neighbors=12…
[CV 2/5; 8/19] END n_neighbors=8;, score=(train=0.946, test=0.933) total time=
23.4s
[CV 3/5; 12/19] START n_neighbors=12…
[CV 3/5; 10/19] END n_neighbors=10;, score=(train=0.943, test=0.934) total time=
23.1s
[CV 4/5; 12/19] START n_neighbors=12…
[CV 5/5; 9/19] END n_neighbors=9;, score=(train=0.945, test=0.930) total time=
22.9s
[CV 5/5; 12/19] START n_neighbors=12…
[CV 1/5; 9/19] END n_neighbors=9;, score=(train=0.945, test=0.930) total time=
24.8s
[CV 1/5; 13/19] START n_neighbors=13…
[CV 1/5; 10/19] END n_neighbors=10;, score=(train=0.942, test=0.927) total time=
23.6s
[CV 2/5; 13/19] START n_neighbors=13…
[CV 4/5; 9/19] END n_neighbors=9;, score=(train=0.945, test=0.932) total time=
24.7s
[CV 3/5; 13/19] START n_neighbors=13…
[CV 3/5; 9/19] END n_neighbors=9;, score=(train=0.945, test=0.935) total time=
24.3s
[CV 4/5; 13/19] START n_neighbors=13…
[CV 4/5; 10/19] END n_neighbors=10;, score=(train=0.942, test=0.929) total time=
24.4s
[CV 5/5; 13/19] START n_neighbors=13…
[CV 5/5; 11/19] END n_neighbors=11;, score=(train=0.941, test=0.929) total time=
22.7s
[CV 1/5; 14/19] START n_neighbors=14…
[CV 5/5; 10/19] END n_neighbors=10;, score=(train=0.942, test=0.929) total time=
23.3s
[CV 2/5; 14/19] START n_neighbors=14…
[CV 5/5; 12/19] END n_neighbors=12;, score=(train=0.939, test=0.926) total time=
23.7s
[CV 3/5; 14/19] START n_neighbors=14…
[CV 2/5; 12/19] END n_neighbors=12;, score=(train=0.938, test=0.928) total time=
24.3s
[CV 4/5; 14/19] START n_neighbors=14…
[CV 4/5; 11/19] END n_neighbors=11;, score=(train=0.942, test=0.928) total time=
24.2s
[CV 5/5; 14/19] START n_neighbors=14…
[CV 1/5; 11/19] END n_neighbors=11;, score=(train=0.941, test=0.929) total time=
```

```
23.9s
[CV 1/5; 15/19] START n_neighbors=15…
[CV 3/5; 11/19] END n_neighbors=11;, score=(train=0.941, test=0.934) total time=
24.3s
[CV 2/5; 15/19] START n_neighbors=15…
[CV 2/5; 11/19] END n_neighbors=11;, score=(train=0.940, test=0.931) total time=
24.5s
[CV 3/5; 15/19] START n_neighbors=15…
[CV 4/5; 12/19] END n_neighbors=12;, score=(train=0.940, test=0.927) total time=
23.3s
[CV 4/5; 15/19] START n_neighbors=15…
[CV 1/5; 13/19] END n_neighbors=13;, score=(train=0.937, test=0.926) total time=
23.9s
[CV 3/5; 12/19] END n_neighbors=12;, score=(train=0.939, test=0.931) total time=
23.9s
[CV 5/5; 15/19] START n_neighbors=15…
[CV 1/5; 16/19] START n_neighbors=16…
[CV 1/5; 12/19] END n_neighbors=12;, score=(train=0.938, test=0.927) total time=
24.6s
[CV 2/5; 16/19] START n_neighbors=16…
[CV 3/5; 13/19] END n_neighbors=13;, score=(train=0.937, test=0.932) total time=
24.2s
[CV 3/5; 16/19] START n_neighbors=16…
[CV 4/5; 13/19] END n_neighbors=13;, score=(train=0.939, test=0.927) total time=
23.4s
[CV 4/5; 16/19] START n_neighbors=16…
[CV 2/5; 13/19] END n_neighbors=13;, score=(train=0.937, test=0.929) total time=
26.7s
[CV 5/5; 16/19] START n_neighbors=16…
[CV 2/5; 14/19] END n_neighbors=14;, score=(train=0.935, test=0.928) total time=
21.9s
[CV 1/5; 17/19] START n_neighbors=17…
[CV 5/5; 13/19] END n_neighbors=13;, score=(train=0.938, test=0.927) total time=
21.9s
[CV 2/5; 17/19] START n_neighbors=17…
[CV 1/5; 14/19] END n_neighbors=14;, score=(train=0.936, test=0.925) total time=
21.6s
[CV 3/5; 17/19] START n_neighbors=17…
[CV 3/5; 15/19] END n_neighbors=15;, score=(train=0.935, test=0.930) total time=
22.0s
[CV 4/5; 17/19] START n_neighbors=17…
[CV 2/5; 15/19] END n_neighbors=15;, score=(train=0.934, test=0.929) total time=
21.5s
[CV 5/5; 17/19] START n_neighbors=17…
[CV 4/5; 14/19] END n_neighbors=14;, score=(train=0.937, test=0.925) total time=
21.1s
[CV 4/5; 15/19] END n_neighbors=15;, score=(train=0.936, test=0.924) total time=
22.2s
```

```
[CV 1/5; 18/19] START n_neighbors=18…
[CV 2/5; 18/19] START n_neighbors=18…
[CV 1/5; 16/19] END n_neighbors=16;, score=(train=0.934, test=0.924) total time=
22.5s
[CV 3/5; 18/19] START n_neighbors=18…
[CV 3/5; 14/19] END n_neighbors=14;, score=(train=0.935, test=0.931) total time=
23.2s
[CV 4/5; 18/19] START n_neighbors=18…
[CV 3/5; 16/19] END n_neighbors=16;, score=(train=0.933, test=0.928) total time=
21.5s
[CV 5/5; 18/19] START n_neighbors=18…
[CV 5/5; 14/19] END n_neighbors=14;, score=(train=0.936, test=0.924) total time=
23.4s
[CV 1/5; 19/19] START n_neighbors=19…
[CV 2/5; 16/19] END n_neighbors=16;, score=(train=0.933, test=0.927) total time=
23.1s
[CV 2/5; 19/19] START n_neighbors=19…
[CV 5/5; 15/19] END n_neighbors=15;, score=(train=0.936, test=0.924) total time=
22.9s
[CV 3/5; 19/19] START n_neighbors=19…
[CV 1/5; 15/19] END n_neighbors=15;, score=(train=0.935, test=0.927) total time=
22.9s
[CV 4/5; 19/19] START n_neighbors=19…
[CV 4/5; 16/19] END n_neighbors=16;, score=(train=0.935, test=0.921) total time=
23.9s
[CV 5/5; 19/19] START n_neighbors=19…
[CV 5/5; 16/19] END n_neighbors=16;, score=(train=0.934, test=0.923) total time=
23.0s
[CV 3/5; 17/19] END n_neighbors=17;, score=(train=0.932, test=0.929) total time=
19.2s
[CV 2/5; 17/19] END n_neighbors=17;, score=(train=0.932, test=0.927) total time=
20.1s
[CV 1/5; 17/19] END n_neighbors=17;, score=(train=0.934, test=0.923) total time=
21.5s
[CV 4/5; 17/19] END n_neighbors=17;, score=(train=0.934, test=0.921) total time=
19.9s
[CV 3/5; 18/19] END n_neighbors=18;, score=(train=0.931, test=0.928) total time=
19.3s
[CV 2/5; 18/19] END n_neighbors=18;, score=(train=0.930, test=0.924) total time=
19.7s
[CV 4/5; 19/19] END n_neighbors=19;, score=(train=0.931, test=0.920) total time=
19.5s
[CV 1/5; 18/19] END n_neighbors=18;, score=(train=0.932, test=0.921) total time=
20.4s
[CV 5/5; 18/19] END n_neighbors=18;, score=(train=0.931, test=0.923) total time=
20.6s
[CV 5/5; 17/19] END n_neighbors=17;, score=(train=0.933, test=0.923) total time=
21.6s
```

```
[CV 4/5; 18/19] END n_neighbors=18;, score=(train=0.932, test=0.921) total time=
20.9s
[CV 2/5; 19/19] END n_neighbors=19;, score=(train=0.930, test=0.925) total time=
20.5s
[CV 1/5; 19/19] END n_neighbors=19;, score=(train=0.931, test=0.921) total time=
21.1s
[CV 3/5; 19/19] END n_neighbors=19;, score=(train=0.931, test=0.928) total time=
19.9s
[CV 5/5; 19/19] END n_neighbors=19;, score=(train=0.930, test=0.921) total time=
19.3s
{'n_neighbors': 3}
0.9352380952380953
```

[ ]:
```python
cv_results = pd.DataFrame(optimal_parameters.cv_results_)
# print(cv_results)
# # converting C to numeric type for plotting on x-axis
cv_results['param_n_neighbors']=cv_results['param_n_neighbors'].astype('int')
plt.figure(figsize=(16,8))

plt.subplot(131)
plt.plot(cv_results["param_n_neighbors"], cv_results["mean_test_score"])
plt.plot(cv_results["param_n_neighbors"], cv_results["mean_train_score"])
plt.xlabel('Neighbors')
plt.ylabel('Accuracy')
plt.title("KNN")
plt.ylim([0.20, 1])
plt.legend(['test accuracy', 'train accuracy'], loc='lower left')
plt.show()
```



[ ]:
```
from sklearn.neighbors import NearestCentroid

ncc_model = NearestCentroid()
ncc_model.fit(X_train, y_train)
y_pred_test = ncc_model.predict(X_test)
y_pred_train = ncc_model.predict(X_train)
```

```
print("Train accuracy: ",metrics.accuracy_score(y_true=y_train,␣
  ↪y_pred=y_pred_train),"\n")
print("Test accuracy:", metrics.accuracy_score(y_true=y_test,␣
  ↪y_pred=y_pred_test), "\n")
print(f"Test Set Score : {ncc_model.score(X_test, y_test) * 100} %")
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_test)
plt.show()
```

```
Train accuracy:  0.804031746031746

Test accuracy: 0.8000952380952381

Test Set Score : 80.00952380952381 %
```