Final examination 2(a) SVM

May 8, 2020

[18]:
```
#Suja Basnet
#Machine Learning Fundamentals
#Final Examinatiom

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
```

[19]:
```
# read the dataset
train_data = pd.read_csv("train 2.csv")
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42000 entries, 0 to 41999
Columns: 785 entries, label to pixel783
dtypes: int64(785)
memory usage: 251.5 MB
```

[20]:
```
# head
train_data.head()
```

[20]:

	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]
```

[21]:
```
# read the dataset
test_data = pd.read_csv("test 2.csv")
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28000 entries, 0 to 27999
Columns: 784 entries, pixel0 to pixel783
dtypes: int64(784)
memory usage: 167.5 MB
```

[22]:
```
# head
test_data.head()
```

[22]:
```
   pixel0  pixel1  pixel2  pixel3  pixel4  pixel5  pixel6  pixel7  pixel8  \
0       0       0       0       0       0       0       0       0       0
1       0       0       0       0       0       0       0       0       0
2       0       0       0       0       0       0       0       0       0
3       0       0       0       0       0       0       0       0       0
4       0       0       0       0       0       0       0       0       0

   pixel9  …  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 784 columns]
```

[23]:
```
train_data.isnull().sum().head(10)
```

[23]:
```
label     0
pixel0    0
pixel1    0
pixel2    0
pixel3    0
pixel4    0
pixel5    0
pixel6    0
pixel7    0
pixel8    0
dtype: int64
```

[24]:
```
test_data.isnull().sum().head(10)
```

[24]:
```
pixel0    0
pixel1    0
pixel2    0
pixel3    0
pixel4    0
pixel5    0
pixel6    0
pixel7    0
pixel8    0
pixel9    0
dtype: int64
```

[25]:
```
test_data.describe()
```

[25]:

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	\
count	28000.0	28000.0	28000.0	28000.0	28000.0	28000.0	28000.0	28000.0	
mean	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
std	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
min	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
25%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
50%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
75%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
max	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	pixel8	pixel9	…	pixel774	pixel775	pixel776	\
count	28000.0	28000.0	…	28000.000000	28000.000000	28000.000000	
mean	0.0	0.0	…	0.164607	0.073214	0.028036	
std	0.0	0.0	…	5.473293	3.616811	1.813602	

```
min        0.0      0.0  …      0.000000      0.000000      0.000000
25%        0.0      0.0  …      0.000000      0.000000      0.000000
50%        0.0      0.0  …      0.000000      0.000000      0.000000
75%        0.0      0.0  …      0.000000      0.000000      0.000000
max        0.0      0.0  …    253.000000    254.000000    193.000000

           pixel777      pixel778  pixel779  pixel780  pixel781  pixel782  \
count  28000.000000  28000.000000   28000.0   28000.0   28000.0   28000.0
mean       0.011250      0.006536       0.0       0.0       0.0       0.0
std        1.205211      0.807475       0.0       0.0       0.0       0.0
min        0.000000      0.000000       0.0       0.0       0.0       0.0
25%        0.000000      0.000000       0.0       0.0       0.0       0.0
50%        0.000000      0.000000       0.0       0.0       0.0       0.0
75%        0.000000      0.000000       0.0       0.0       0.0       0.0
max      187.000000    119.000000       0.0       0.0       0.0       0.0

       pixel783
count   28000.0
mean        0.0
std         0.0
min         0.0
25%         0.0
50%         0.0
75%         0.0
max         0.0

[8 rows x 784 columns]
```

[26]:
```
train_data.describe()
```

[26]:
```
              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]
```

[27]:
```
print(train_data.columns)
print(test_data.columns)
```

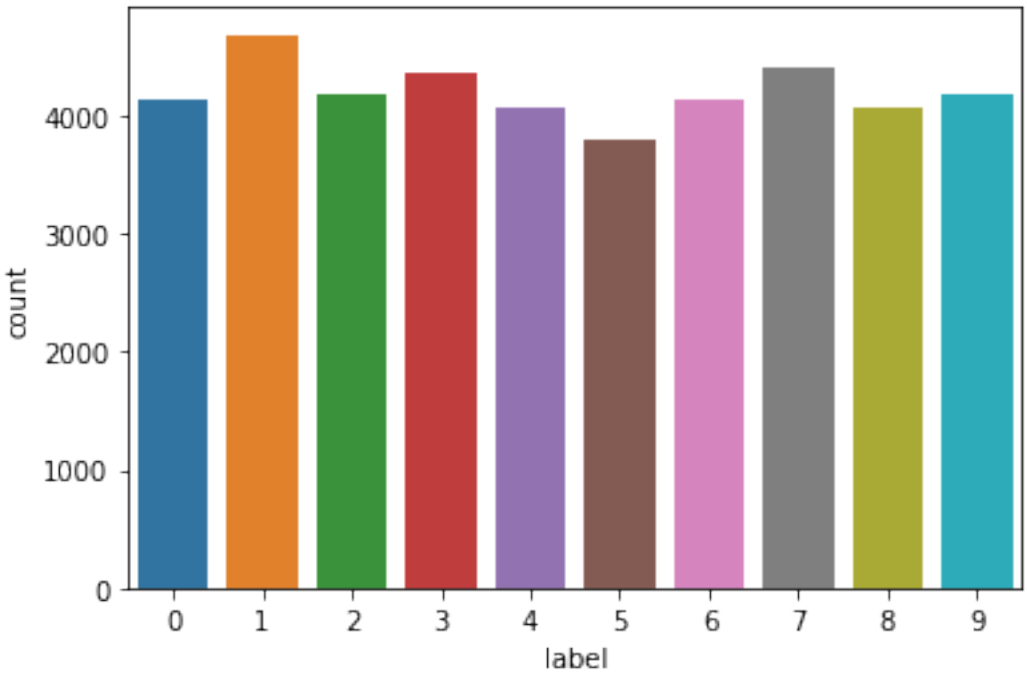
```
Index(['label', 'pixel0', 'pixel1', 'pixel2', 'pixel3', 'pixel4', 'pixel5',
       'pixel6', 'pixel7', 'pixel8',
       …
       'pixel774', 'pixel775', 'pixel776', 'pixel777', 'pixel778', 'pixel779',
       'pixel780', 'pixel781', 'pixel782', 'pixel783'],
      dtype='object', length=785)
Index(['pixel0', 'pixel1', 'pixel2', 'pixel3', 'pixel4', 'pixel5', 'pixel6',
       'pixel7', 'pixel8', 'pixel9',
       …
       'pixel774', 'pixel775', 'pixel776', 'pixel777', 'pixel778', 'pixel779',
       'pixel780', 'pixel781', 'pixel782', 'pixel783'],
      dtype='object', length=784)
```

[28]:
```
order = list(np.sort(train_data['label'].unique()))
print(order)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

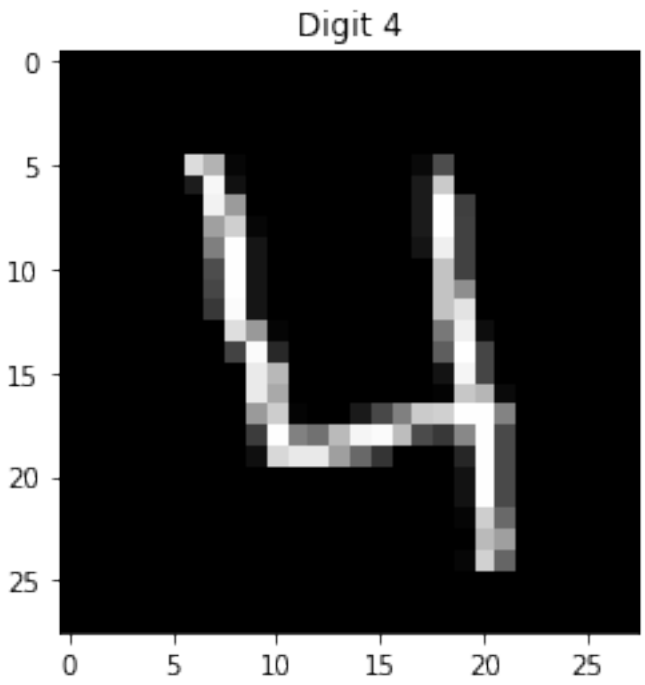
[29]:
```
sns.countplot(train_data["label"])
```

[29]: <matplotlib.axes._subplots.AxesSubplot at 0x121f35978>



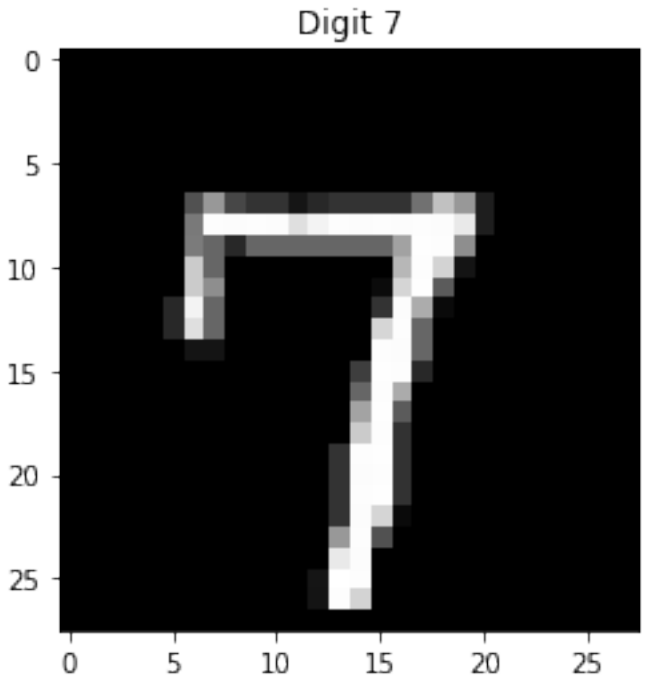
[30]:
```
# Plotting some samples as well as converting into matrix

four = train_data.iloc[3, 1:]
four.shape
four = four.values.reshape(28,28)
plt.imshow(four, cmap='gray')
plt.title("Digit 4")
```

[30]: Text(0.5, 1.0, 'Digit 4')



[31]:
```
seven = train_data.iloc[6, 1:]
seven.shape
seven = seven.values.reshape(28, 28)
plt.imshow(seven, cmap='gray')
plt.title("Digit 7")
```

[31]: Text(0.5, 1.0, 'Digit 7')



[32]:
```
## Separating the X and Y variable

y = train_data['label']

## Dropping the variable 'label' from X variable
X = train_data.drop(columns = 'label')

## Printing the size of data
print(train_data.shape)
```

```
(42000, 785)
```

[33]:
```
# scaling the features
from sklearn.preprocessing import scale
X_scaled = scale(X)

# train test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size = 0.
↪3, train_size = 0.2 ,random_state = 10)
```

[34]:
```
# creating a KFold object with 5 splits
folds = KFold(n_splits = 5, shuffle = True, random_state = 10)

# specify range of hyperparameters
# Set the parameters by cross-validation
```

```
hyper_params = [ {'gamma': [1e-2, 1e-3, 1e-4],
                     'C': [5,10]}]


# specify model
model = SVC(kernel="rbf")

# set up GridSearchCV()
model_cv = GridSearchCV(estimator = model,
                        param_grid = hyper_params,
                        scoring= 'accuracy',
                        cv = folds,
                        verbose = 1,
                        return_train_score=True)

# fit the model
model_cv.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed: 44.4min finished
```

[34]:
```
GridSearchCV(cv=KFold(n_splits=5, random_state=10, shuffle=True),
             error_score=nan,
             estimator=SVC(C=1.0, break_ties=False, cache_size=200,
                           class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='scale', kernel='rbf', max_iter=-1,
                           probability=False, random_state=None, shrinking=True,
                           tol=0.001, verbose=False),
             iid='deprecated', n_jobs=None,
             param_grid=[{'C': [5, 10], 'gamma': [0.01, 0.001, 0.0001]}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='accuracy', verbose=1)
```

[36]:
```
#Accuracy and Confusion Matrix
from sklearn import metrics
from sklearn.metrics import confusion_matrix

# model
model = SVC(C=10, gamma=0.001, kernel="rbf")

model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# metrics
print("accuracy", metrics.accuracy_score(y_test, y_pred), "\n")
```

```
print(metrics.confusion_matrix(y_test, y_pred), "\n")
```

```
accuracy 0.9438888888888889

[[1163    0    4    1    1    2    8    6    3    0]
 [   0 1389    4    2    4    0    1    9    4    0]
 [   1    4 1184   14    5    1    9   30    7    5]
 [   0    3   15 1263    0   14    2   23    8    3]
 [   1    2   20    3 1149    0   10   10    2   21]
 [   2    8    3   30    4 1064   15    9   11    3]
 [   8    1    3    0    3   13 1167   23    1    0]
 [   4    9   10    8   12    0    0 1255    2   30]
 [   5   18   17   23    8   20    5   13 1098   10]
 [   5    3    2   27   21    1    1   51    3 1161]]
```

[ ]:

[ ]: