PRACTICAL - 5 & 6

NAME: ARGHYADIP GHOSH

UID- 18BCS6081

CLASS- AIML-1

GROUP- A

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv('/content/train.csv')
df
```

Out[ ]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11	pixel12	pixel13	pixel14	pixel1
0	1	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	
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	4	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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
564	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
565	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
566	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
567	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
568	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

569 rows × 785 columns

In [ ]:

```
df.shape
```

Out[ ]:

```
(569, 785)
```

In [ ]:

```
df = df.fillna(0)
```

In [ ]:

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Columns: 785 entries, label to pixel783
dtypes: float64(88), int64(697)
```

```
memory usage: 3.4 MB
None
```

In [ ]:

```
df.columns
```

Out[ ]:

```
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)
```

In [ ]:

```
order = list(np.sort(df['label'].unique()))
print(order)
```

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

In [ ]:

```
num_mean = df.groupby('label').mean()
num_mean.head()
```

Out[ ]:

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11	pixel12	pixel13	pixel14	pixel15	pi
label																	
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	
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	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	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	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	0.0	0.0	

5 rows × 784 columns

In [ ]:

```
round(df.drop('label',axis = 1).mean(),2)
```

Out[ ]:

```
pixel0      0.0
pixel1      0.0
pixel2      0.0
pixel3      0.0
pixel4      0.0
           ...
pixel779    0.0
pixel780    0.0
pixel781    0.0
pixel782    0.0
pixel783    0.0
Length: 784, dtype: float64
```

In [ ]:

```
df = df.reset_index()
```

In [ ]:

```
X = df.iloc[:,2:].values
y = df.iloc[:,0].values
```

In [ ]:

```
X.shape
```

Out[ ]:

```
(569, 784)
```

In [ ]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
```

In [ ]:

```
from sklearn.model_selection import KFold
fold = KFold(n_splits = 5, shuffle = True )
hyper_param = [{'gamma':[1e-2,1e-3,1e-4],
                'C':[1,10,100,1000]}]
```

In [ ]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
svc_rbf = SVC(kernel = 'rbf')
model_cv = GridSearchCV(estimator = svc_rbf, param_grid = hyper_param , scoring = 'accuracy', cv =
fold , verbose = 1 , return_train_score= True)
model_cv.fit(X,y)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  60 out of  60 | elapsed:  2.3min finished
```

Out[ ]:

```
GridSearchCV(cv=KFold(n_splits=5, random_state=None, 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': [1, 10, 100, 1000],
                          'gamma': [0.01, 0.001, 0.0001]}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='accuracy', verbose=1)
```

In [ ]:

```
cv_results = pd.DataFrame(model_cv.cv_results_)
cv_results
```

Out[ ]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_C	param_gamma	params	split0_test_score	split1_test_sco
0	1.210020	0.028390	0.235375	0.016800	1	0.01	{'C': 1, 'gamma': 0.01}	0.0	0
1	1.181412	0.008549	0.239847	0.025961	1	0.001	{'C': 1, 'gamma': 0.001}	0.0	0

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_C	param_gamma	params	split0_test_score	split1_test_sco
2	1.184769	0.025793	0.236276	0.035164	1	0.0001	{'C': 1, 'gamma': 0.0001}	0.0	0
3	1.183616	0.013374	0.219993	0.020165	10	0.01	{'C': 10, 'gamma': 0.01}	0.0	0
4	1.170103	0.014160	0.213069	0.012774	10	0.001	{'C': 10, 'gamma': 0.001}	0.0	0
5	1.193024	0.008776	0.208243	0.008646	10	0.0001	{'C': 10, 'gamma': 0.0001}	0.0	0
6	1.171226	0.007729	0.216625	0.016449	100	0.01	{'C': 100, 'gamma': 0.01}	0.0	0
7	1.162791	0.009975	0.214288	0.011952	100	0.001	{'C': 100, 'gamma': 0.001}	0.0	0
8	1.187706	0.004397	0.219455	0.012277	100	0.0001	{'C': 100, 'gamma': 0.0001}	0.0	0
9	1.228700	0.018532	0.261499	0.024407	1000	0.01	{'C': 1000, 'gamma': 0.01}	0.0	0
10	1.240289	0.033943	0.260133	0.026525	1000	0.001	{'C': 1000, 'gamma': 0.001}	0.0	0
11	1.211648	0.009252	0.225420	0.010397	1000	0.0001	{'C': 1000, 'gamma': 0.0001}	0.0	0

In [ ]:

```
y_pred = model_cv.predict(X)
```

In [ ]:

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y,y_pred))
```

```
1.0
```

In [ ]:

```
df2 = pd.read_csv('/content/sample_data/test.csv')
df2
```

Out[ ]:

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11	pixel12	pixel13	pixel14	pixel15	p
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	
2	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	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
27995	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
27996	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
27997	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
27998	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
27999	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

28000 rows × 784 columns

28000 rows × 784 columns

In [ ]:

```
df2.columns
```

Out[ ]:

```
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)
```

In [ ]:

```
df2 = df2.fillna(0)
```

In [ ]:

```
X_test = df2.iloc[:,:].values
```

In [ ]:

```
y_pred_test = model_cv.predict(X_test)
```

In [ ]:

```
df3 = pd.read_csv('/content/sample_submission.csv')
```

In [ ]:

```
y_test = df3.iloc[:,-1].values
```

In [ ]:

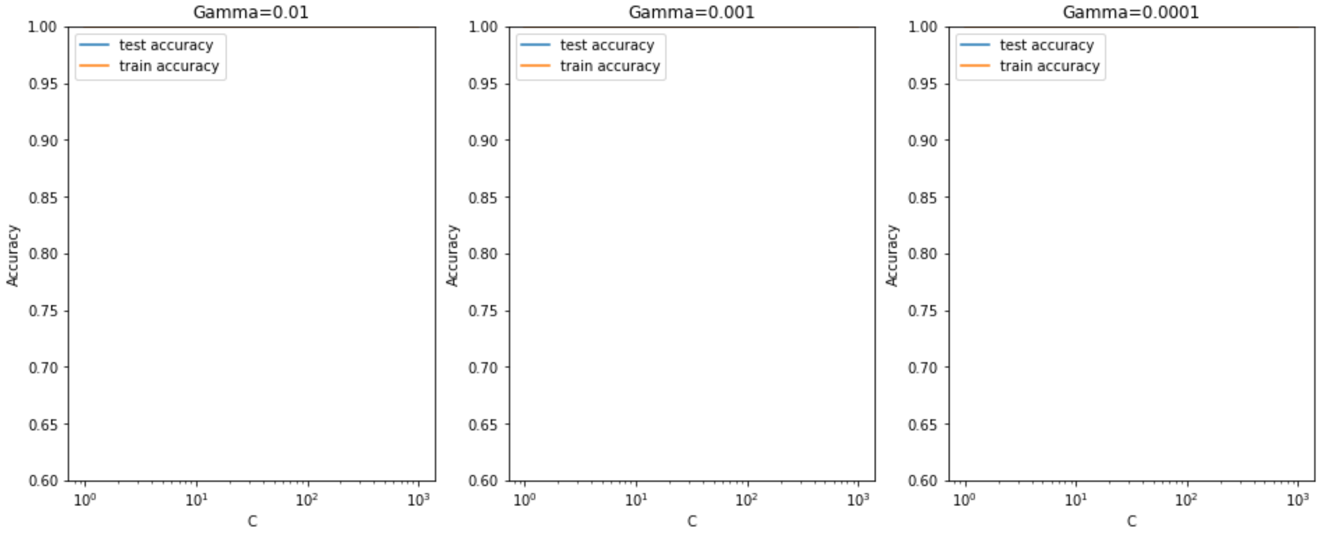
```
print(accuracy_score(y_test,y_pred_test))
```

```
0.0
```

In [ ]:

```
cv_results['param_C'] = cv_results['param_C'].astype('int')
# # plotting
plt.figure(figsize=(16,6))
# subplot 1/3
plt.subplot(131)
gamma_01 = cv_results[cv_results['param_gamma']==0.01]
plt.plot(gamma_01["param_C"], gamma_01["mean_test_score"])
plt.plot(gamma_01["param_C"], gamma_01["mean_train_score"])
plt.xlabel('C')
plt.ylabel('Accuracy')
plt.title("Gamma=0.01")
plt.ylim([0.60, 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']==0.001]
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=0.001")
plt.ylim([0.60, 1])

plt.legend(['test accuracy', 'train accuracy'], loc='upper left')
plt.xscale('log')
plt.xscale('log')
# subplot 3/3
plt.subplot(133)
gamma_0001 = cv_results[cv_results['param_gamma']==0.0001]
plt.plot(gamma_0001["param_C"], gamma_0001["mean_test_score"])
plt.plot(gamma_0001["param_C"], gamma_0001["mean_train_score"])
plt.xlabel('C')
plt.ylabel('Accuracy')
plt.title("Gamma=0.0001")
plt.ylim([0.60, 1])
plt.legend(['test accuracy', 'train accuracy'], loc='upper left')
plt.xscale('log')
```



In [ ]: