# Assignment_6

February 9, 2022

## 1 MSDS 422 Assignment 6: Digit Recognizer

**Kay Quiballo | 02/04/2022**

### 1.1 Management/Research Question

The classic MNIST Digit Recognizer problem is a competition on Kaggle.com, and you will compete in this competition. For this assignment, you will develop a classifier that may be used to predict which of the 10 digits is being written.

### 1.2 Requirements

1. Fit a random forest classifier using the full set of explanatory variables and the model training set (csv).
2. Record the time it takes to fit the model and then evaluate the model on the csvdata by submitting to Kaggle.com. Provide your Kaggle.com score and user ID.
3. Execute principal components analysis (PCA) on the combined training and test set data together, generating principal components that represent 95 percent of the variability in the explanatory variables. The number of principal components in the solution should be substantially fewer than the explanatory variables.
4. Record the time it takes to identify the principal components.
5. Using the identified principal components from step (2), use thecsvto build another random forest classifier.
6. Record the time it takes to fit the model and to evaluate the model on the csvdata by submitting to Kaggle.com. Provide your Kaggle.com score and user ID.
7. Use k-means clustering to group MNIST observations into 1 of 10 categories and then assign labels. (Follow the example here if needed: kmeans mnist.pdf Download kmeans mnist.pdf).kmeans mnist-2.pdf Download kmeans mnist-2.pdf
8. Submit the RF Classifier, the PCA RF, and k-means estimations to Kaggle.com, and provide screen snapshots of your scores as well as your Kaggle.com user name.
9. The experiment we have proposed has a major design flaw. Identify the flaw. Fix it. Rerun the experiment in a way that is consistent with a training-and-test regimen, and submit this to Kaggle.com. Report total elapsed time measures for the training set analysis. It is sufficient to run a single time-elapsed test for this assignment. In practice, we might consider the possibility of repeated executions of the relevant portions of the programs, much as the

Benchmark Example programs do. Some code that might help you with reporting elapsed total time follows.

start=datetime.now() rf2.fit(trainimages,labels) end=datetime.now() print(end-start)

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

from sklearn.metrics import classification_report, confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
```

[2]:
```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

[3]:
```
train.head()
```

[3]:

| | 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]

[4]:
```
test.head()
```

[4]:

| | 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]
```

[5]:
```
X_train = train.drop('label', axis=1)
y_train = train['label']
```

**1. Fit a random forest classifier using the full set of explanatory variables and the model training set (csv).**

[6]:
```
start = time.process_time()

#create model
model1 = RandomForestClassifier()
#sm = SMOTE(sampling_strategy='minority', random_state=42)
#Xsm_train, ysm_train = sm.fit_resample(x_train, y_train)
model1 = model1.fit(X_train, y_train)
#prediction = model1.predict(X_train)
#print(classification_report(y_train, prediction))

run_time = time.process_time() - start

#eval
model = 'Random Forest Classifier'
kaggle_score = 0.96503
print(model, ",", run_time, "seconds, Score:", kaggle_score)
```

```
Random Forest Classifier , 28.621914 seconds, Score: 0.96503
```

[7]:
```
#output file
output = pd.DataFrame({'ImageId': np.linspace(1, 28000, 28000).astype(int),␣
 ↪'Label': model1.predict(test)})
output.to_csv('output_rfc.csv', index=False)
```

**2. Record the time it takes to fit the model and then evaluate the model on the csvdata by submitting to Kaggle.com. Provide your Kaggle.com score and user ID.** It took ~29 seconds to execute. The score for this model is 0.96503.

**3. Execute principal components analysis (PCA) on the combined training and test set data together, generating principal components that represent 95 percent of the variability in the explanatory variables. The number of principal components in the solution should be substantially fewer than the explanatory variables.**

[8]:
```
from sklearn.decomposition import PCA

X_combined = pd.concat([X_train, test])

start = time.process_time()
pca = PCA(n_components=10)
principalComponents = pca.fit_transform(X_combined)
print(time.process_time() - start)

principalDf_train = pd.DataFrame(data = principalComponents).head(42000)
principalDf_test = pd.DataFrame(data = principalComponents).tail(28000)

#confirming principal components represent 95 percent of the variability in the␣
 ↪explanatory variables
ex_variance=np.var(principalComponents,axis=0)
ex_variance_ratio = ex_variance/np.sum(ex_variance)
print(ex_variance_ratio)
0.19923771+0.14627718+0.12571351+0.1104602+0.099943+0.08801081+0.06701678+0.
 ↪0590723+0.05638858+0.04787952
```

```
10.700187
[0.19923766 0.14627715 0.12571348 0.11046018 0.09994337 0.08801079
 0.06701676 0.05907229 0.05638857 0.04787975]
```

[8]:
```
0.9999995899999999
```

**4. Record the time it takes to identify the principal components.** It took ~4 seconds to execute.

**5. Using the identified principal components from step (2), use the csv to build another random forest classifier.**

[9]:
```
start = time.process_time()

#create model
model2 = RandomForestClassifier()
#sm = SMOTE(sampling_strategy='minority', random_state=42)
#Xsm_train, ysm_train = sm.fit_resample(x_train, y_train)
```

```
model2 = model2.fit(principalDf_train, y_train)
#prediction = model2.predict(X_train)
#print(classification_report(y_train, prediction))

run_time = time.process_time() - start

#eval
model = 'Random Forest Classifier (PCA)'
kaggle_score = 0.90885
print(model, ",", run_time, "seconds, Score:", kaggle_score)
```

```
Random Forest Classifier (PCA) , 14.842851000000003 seconds, Score: 0.90885
```

[10]:
```
#output file
output = pd.DataFrame({'ImageId': np.linspace(1, 28000, 28000).astype(int),␣
↪'Label': model2.predict(principalDf_test)})
output.to_csv('output_pca.csv', index=False)
```

**6. Record the time it takes to fit the model and to evaluate the model on the csvdata by submitting to Kaggle.com. Provide your Kaggle.com score and user ID.** It took ~15 seconds to execute. The score for this model is 0.90885.

**7. Use k-means clustering to group MNIST observations into 1 of 10 categories and then assign labels.**

[42]:
```
#X_combined = pd.concat([X_train, test])
#pca = PCA(n_components=10)
#principalComponents = pca.fit_transform(X_combined)
#principalDf_train = pd.DataFrame(data = principalComponents).head(42000)
#principalDf_test = pd.DataFrame(data = principalComponents).tail(28000)


from sklearn.preprocessing import StandardScaler
X_std = StandardScaler().fit_transform(X_combined.values)

pca = PCA(n_components=10)
principalComponents = pca.fit_transform(X_std)
principalDf = pd.DataFrame(data = principalComponents)
#principalDf_train = pd.DataFrame(data = principalComponents).head(42000)
#principalDf_test = pd.DataFrame(data = principalComponents).tail(28000)

from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=10)
X_clustered = kmeans.fit_predict(principalDf)
X_clustered_train = X_clustered[:42000]
X_clustered_test = X_clustered[42000:]
```

```
len(X_clustered_train)
len(X_clustered_test)
len(y_train)
```

[42]: 42000

[45]:
```
start = time.process_time()

model3 = RandomForestClassifier()
model3 = model3.fit(X_clustered_train.reshape(-1, 1), y_train)

run_time = time.process_time() - start

#eval
model = 'KMeans'
kaggle_score = 0
print(model, ",", run_time, "seconds, Score:", kaggle_score)
```

```
KMeans , 0.8913180000000125 seconds, Score: 0
[1 0 1 … 3 4 2]
0        1
1        0
2        1
3        4
4        0
        ..
41995    0
41996    1
41997    7
41998    6
41999    9
Name: label, Length: 42000, dtype: int64
```

[46]:
```
#output file
output = pd.DataFrame({'ImageId': np.linspace(1, 28000, 28000).astype(int),␣
 ↪'Label': model3.predict(X_clustered_test.reshape(-1, 1))})
output.to_csv('output_kmeans.csv', index=False)
```

**8. Submit the RF Classifier, the PCA RF, and k-means estimations to Kaggle.com, and provide screen snapshots of your scores as well as your Kaggle.com user name.** It took ~1 second to execute. The score for this model is 0.53025.

**9. The experiment we have proposed has a major design flaw. Identify the flaw. Fix it. Rerun the experiment in a way that is consistent with a training-and-test regimen, and submit this to Kaggle.com. Report total elapsed time measures for the training set analysis. It is sufficient to run a single time-elapsed test for this assignment.**

**In practice, we might consider the possibility of repeated executions of the relevant portions of the programs, much as the Benchmark Example programs do. Some code that might help you with reporting elapsed total time follows.**

[ ]:
```
#start=datetime.now()
#rf2.fit(trainimages,labels)
#end=datetime.now()
#print(end-start)
```

To implement consistent training-and-testing regimen we need to split the training data into training/testing subset and us cross validation to check validity of models.

[52]:
```
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_train, y_train,␣
 ↪test_size=0.20, shuffle=False)

start = time.process_time()
model1 = RandomForestClassifier()
model1 = model1.fit(X_train, y_train)
run_time = time.process_time() - start

prediction = model1.predict(X_test)
print(classification_report(y_test, prediction))

#eval
model = 'Random Forest + CV'
kaggle_score = 0
print(model, ",", run_time, "seconds, Score:", kaggle_score)
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       298
           1       0.97      0.99      0.98       402
           2       0.93      0.93      0.93       326
           3       0.95      0.93      0.94       367
           4       0.95      0.95      0.95       358
           5       0.96      0.93      0.94       296
           6       0.98      0.99      0.98       342
           7       0.97      0.95      0.96       333
           8       0.94      0.95      0.95       342
           9       0.95      0.94      0.95       377

    accuracy                           0.96      3441
   macro avg       0.96      0.96      0.96      3441
weighted avg       0.96      0.96      0.96      3441

Random Forest + CV , 9.370975000000044 seconds, Score: 0
```

[53]:
```
#output file
output = pd.DataFrame({'ImageId': np.linspace(1, 28000, 28000).astype(int),␣
 ↪'Label': model1.predict(test)})
output.to_csv('output_rfc_cv.csv', index=False)
```

It took ~10 second to execute. The score for this model is 0.95414.

[ ]: