Data Preparation, Exploration and Visualization

The purpose of assignment 5 is to use the random forest learning method in order to build a multiclass prediction for the MNIST data set. The overall goal is to build a mode that is able to correlate a digit to a handwritten digit. While there are variations of handwritten digits, we want to be able to build a model that can still assign a value to this digit as well. In addition to dealing with this challenge, we also want to build a model that can do this in time since the dataset is pretty large. The data set has 70,000 handwritten digits, 785 columns and 784 columns are grey scale. My first visualization was in lines 42-43 where I first created a confusion matrix of the data set then I wanted to see the plotted data for a sample set in line 43. After doing the initial analysis I dove into the modeling process.

Research Design and Modeling Methods

The steps for this assignment where to use a random forest model for each variable (784 of them), then use PCA to reduce the data, then fit another random forest model using the reduced set and then compared the scores and the timing of the models using the F1 and precision recall scores. This was done from lines 48-70. The first step was to split the data into training and test sets, then randomize them. I then used a confusion matrix to compare the effectiveness of the models.

Review Results, Evaluate Models

For the first model with all the estimators (784), I found that the run time was 21.359 minutes, the f1 score was 0.997 (lines 53-55) . The second model with the first reduced PCA with 332 estimators ran at around 3.8 minutes with an f1 of 0.9439 (lines 61-66). Lastly, the third reduced

PCA on training data with 332 estimators, ran at 7 minutes with a f1 score of 0.984 (lines 66-68). In the last model, we fixed the major flaw that the PCA was running on both test and training data by running a model that runs PCA on just training data and applying that to testing and training data. This reduced the run time to around 3 minutes while keeping the accuracy the relatively high.

Kaggle Submission and Scores

Username: Ankita Avadhani and the scores for RF Classifier 0.96945, PCA RFC was 0.94932 and for the PCA RFC 2 (reduced) submission score was 0.94992.

https://www.kaggle.com/c/digit-recognizer/leaderboard#score

Exposition, Management Problem and Recommendations

If I were a manager of a team, I would recommend using PCA for a preliminary ML classification, specifically the third model from this assignments that utilizes PCA on training data and then applying that to testing and training data. The reason why is because this model had a f1 score of 0.94 and a test score of around 0.9 (lines 73-74). This shows an estimators of 320 that explains that 95% variance. The run time is also the fastest of the three models at 3 minutes and yields high accuracy. Even though running the PCA on all estimators gave the highest accuracy, it also caused the highest time. In order to fully navigate a data science project we have to make sure that the time is optimized as well as accurate, which is what PCA does. PCA ultimately maximizes the variance of each dimension in a space and by utilizing this technique, we will be able calculate further covariances and optimize large data sets for computer vision purposes.

In [37]:

```
%matplotlib inline

import os

# Core
import pandas as pd
import numpy as np

# Visuals
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

plt.rcParams['axes.labelsize'] = 14
plt.rcParams['xtick.labelsize'] = 12
plt.rcParams['ytick.labelsize'] = 12


# IPython display
from IPython.display import display

from scipy.io import loadmat # for loading .MAT files
import urllib.request


from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV, cross_validate, cross_val_predict
from sklearn.metrics import accuracy_score, classification_report
from sklearn import metrics
from sklearn.decomposition import PCA
#making sure floats output to 3 decimal points only
pd.set_option('display.float_format', lambda x: '{:.3f}'.format(x))
#Timer for run time
from timeit import default_timer as timer
#Ignore warning from sklearn and seaborn
import warnings
def ignore_warn(*args, **kwargs):
    pass
warnings.warn = ignore_warn
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib

/Users/avadhani/anaconda3/lib/python3.7/site-packages/IPython/core/magi
cs/pylab.py:160: UserWarning: pylab import has clobbered these variable
s: ['test']
`%matplotlib` prevents importing * from pylab and numpy
  "\n`%matplotlib` prevents importing * from pylab and numpy"
```

In [38]:

```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

In [39]:

```
def cross_val(model, train, target, fold, scoring):
    scores = cross_validate(model, train, target, cv = fold, scoring = {scoring})
    #Cross validation scores
    scores_df = pd.DataFrame(scores).iloc[:,2:]
    scores_df.columns.names=['cross_validation']
    return scores_df
```

In [40]:

```
train.head()
```

Out[40]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775
0	1	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	1	0	0	0	0	0	0	0	0	0	...	0	0
3	4	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 × 785 columns

In [41]:

```
# Using unscaled data for evaluation as Random Forests are scale invariant
y_train = train['label'].values
x_train = train.drop(columns=['label'])
x_train = x_train.to_numpy()
x_test = test.to_numpy()
```

In [42]:

```
rnd_clf = RandomForestClassifier(n_estimators=10, max_features='sqrt', bootstrap=Tr
ue, random_state=1, n_jobs=-1)
start = timer()
rnd_clf.fit(x_train, y_train)

y_pred = cross_val_predict(rnd_clf, x_train, y_train, cv=10)
print(train.keys())
print('Classification Report with No Decomposition:')
print(metrics.classification_report(y_train, y_pred), '\n')
print('Confusion Matrix:')
print(metrics.confusion_matrix(y_train, y_pred))
```

```
Index(['label', 'pixel0', 'pixel1', 'pixel2', 'pixel3', 'pixel4', 'pixe
l5',
       'pixel6', 'pixel7', 'pixel8',
       ...
       'pixel774', 'pixel775', 'pixel776', 'pixel777', 'pixel778', 'pix
el779',
       'pixel780', 'pixel781', 'pixel782', 'pixel783'],
      dtype='object', length=785)
Classification Report with No Decomposition:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      4132
           1       0.97      0.98      0.98      4684
           2       0.92      0.95      0.93      4177
           3       0.91      0.92      0.92      4351
           4       0.92      0.95      0.94      4072
           5       0.93      0.91      0.92      3795
           6       0.97      0.96      0.96      4137
           7       0.96      0.94      0.95      4401
           8       0.94      0.90      0.92      4063
           9       0.93      0.90      0.92      4188

    accuracy                           0.94     42000
   macro avg       0.94      0.94      0.94     42000
weighted avg       0.94      0.94      0.94     42000


Confusion Matrix:
[[4044    0   13    8    6    9   24    3   23    2]
 [   1 4605   21   11   11    8    5   10    9    3]
 [  32   18 3956   40   24    9   15   38   34   11]
 [  14   13  101 4021    6   86    8   32   52   18]
 [   5    8   20    9 3883    7   20   10   11   99]
 [  28   12   15  154   24 3463   33    1   46   19]
 [  45   10   14    4   30   49 3965    2   17    1]
 [  10   24   76   18   38    4    0 4152   16   63]
 [  11   33   57   75   42   73   31   13 3660   68]
 [  27    6   22   64  141   31    5   67   35 3790]]
```

In [43]:

```
def plot_digits(instances, images_per_row=10, **options):
    size = 28
    images_per_row = min(len(instances), images_per_row)
    images = [instance.reshape(size,size) for instance in instances]
    n_rows = (len(instances) - 1) // images_per_row + 1
    row_images = []
    n_empty = n_rows * images_per_row - len(instances)
    images.append(np.zeros((size, size * n_empty)))
    for row in range(n_rows):
        rimages = images[row * images_per_row : (row + 1) * images_per_row]
        row_images.append(np.concatenate(rimages, axis=1))
    image = np.concatenate(row_images, axis=0)
    plt.imshow(image, cmap = matplotlib.cm.binary, **options)
    plt.axis("off")

plt.figure(figsize=(9,9))
example_images = np.r_[x_train[:12000:600], x_train[13000:30600:600], x_train[30600
:60000:590]]
plot_digits(example_images, images_per_row=10)
plt.show()
```

In [44]:

```
y_train = train["label"]
x_test = test
```

Random Classifier and Recording Time (Part 1)

In [45]:

```
rfc_grid = {
    'max_depth' : [5,10,20],
    'min_samples_split': [2,5,10],
    'min_samples_leaf': [1,3,7]
}
```

In [46]:

```
rfc = RandomForestClassifier(max_features='sqrt', bootstrap=True, n_estimators=10,
random_state=42, n_jobs=-1)
rfc_grid = GridSearchCV(estimator=rfc, param_grid=rfc_grid, cv=5)
rfc_grid.fit(x_train, y_train)
```

Out[46]:

```
GridSearchCV(cv=5,
             estimator=RandomForestClassifier(max_features='sqrt',
                                              n_estimators=10, n_jobs=-
1,
                                              random_state=42),
             param_grid={'max_depth': [5, 10, 20],
                         'min_samples_leaf': [1, 3, 7],
                         'min_samples_split': [2, 5, 10]})
```

In [53]:

```
from time import time
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
rfc = RandomForestClassifier(max_features='sqrt', bootstrap=True, max_depth=20, min
_samples_leaf=1,
                             min_samples_split=10, n_estimators=500, random_state=42, n_
jobs=-1)

# fit model and record time elapsed - record time to fit model
rf_start_time = time()
rfc.fit(x_train, y_train)
rf_elapsed_time = time() - rf_start_time
print("Time elapsed: {:.3f}".format(rf_elapsed_time))

# run 5 fold cv
cv_rfc = cross_val_score(rfc, x_train, y_train, cv=5, scoring="accuracy")
cv_rfc
```

```
Time elapsed: 21.359
```

Out[53]:

```
array([0.96119048, 0.96285714, 0.9602381 , 0.96369048, 0.96535714])
```

F1 Score

In [55]:

```
from sklearn.metrics import f1_score
y_train_pred = rfc.predict(x_train)
f1_score(y_train, y_train_pred, average="macro")
```

Out[55]:

```
0.9972935365349821
```

Test data model run

In [56]:

```
y_test_pred = rfc.predict(x_test)
print(y_test_pred[:10])

df_rfc=pd.DataFrame({'ImageID' : pd.Series(range(1,28001)), 'Label': y_test_pred})
df_rfc.to_csv('rfc.csv', index=False)
```

```
[2 0 9 9 3 7 0 3 0 3]
```

Kaggle Score 0.96935 , username: Ankita Avadhani (RF Classifier)

PCA on Testing and training data part 2 and 3

In [58]:

```
all_data = pd.concat([train, test])
all_data.shape

ss = StandardScaler()
x_pca = all_data.drop(columns=['label'])
x_pca_scaled = ss.fit_transform(x_pca)
y_pca = pd.DataFrame(all_data['label'])
```

In [59]:

```
PCA_start_time = time()

pca = PCA(n_components=0.95, random_state=42)
pca.fit(x_pca_scaled)

PCA_elapsed_time = time() - PCA_start_time
print("Time elapsed: {:.3f}".format(PCA_elapsed_time))
print("Number of principal components:", pca.n_components_)
```

```
Time elapsed: 7.281
Number of principal components: 332
```

In [61]:

```
x_train_reduced = pca.transform(x_train)
x_test_reduced = pca.transform(x_test)
rfc_pca = RandomForestClassifier(max_features='sqrt', bootstrap=True, max_depth=20,
min_samples_leaf=3,
                         min_samples_split=10, n_estimators=500, random_state=23, n_
jobs=-1)

# fit model and record time elapsed
#Record time to fit model
rf_pca_start_time = time()
rfc_pca.fit(x_train_reduced, y_train)
rf_pca_elapsed_time = time() - rf_pca_start_time
print("Time elapsed: {:.3f}".format(rf_pca_elapsed_time))

# run 5 fold cv
cv_rfc_pca = cross_val_score(rfc_pca, x_train_reduced, y_train, cv=5, scoring="accu
racy")
cv_rfc_pca
```

```
Time elapsed: 57.716
```

Out[61]:

```
array([0.94904762, 0.94940476, 0.9447619 , 0.94869048, 0.95238095])
```

In [66]:

```
from sklearn.metrics import confusion_matrix
y_train_pred_rf_pca = cross_val_predict(rfc_pca, x_train_reduced, y_train, cv=5)

y_train_pred = rfc.predict(x_train)
confusion_matrix(y_train, y_train_pred_rf_pca)
```

Out[66]:

```
array([[4054,    0,   11,    6,    3,    5,   28,    2,   21,    2],
       [   1, 4585,   28,   21,    7,    4,   16,    8,   10,    4],
       [  25,   14, 3956,   37,   31,    1,   22,   43,   43,    5],
       [  10,    2,   90, 4041,    2,   53,   13,   37,   67,   36],
       [   3,   16,   24,    2, 3894,    1,   25,    8,    9,   90],
       [  26,    1,   16,   99,   17, 3535,   52,    0,   29,   20],
       [  33,    5,    3,    0,   10,   33, 4045,    0,    8,    0],
       [   7,   28,   53,    8,   40,    2,    1, 4176,    8,   78],
       [  12,   24,   38,  129,   32,   59,   23,   13, 3695,   38],
       [  26,    6,   14,   72,   89,   14,    5,   69,   22, 3871]])
```

In [67]:

```
f1_score(y_train, y_train_pred_rf_pca, average="macro")
```

Out[67]:

```
0.9484398549189338
```

In [68]:

```
y_test_pred_rf_pca = rfc_pca.predict(x_test_reduced)
print(y_test_pred_rf_pca[:10])

df_rfc_pca=pd.DataFrame({'ImageID' : pd.Series(range(1,28001)), 'Label': y_test_pre
d_rf_pca})
df_rfc_pca.to_csv('rfc_pca.csv', index=False)
```

```
[2 0 9 7 3 7 0 3 0 3]
```

Kaggle username: Ankita Avadhani, score 0.94932 (PCA RF Classifier)

Major Flaw and Fix part 5

The major flaw was that the PCA was running on both the test and training data sets. The way to fix this is to have a model that runs PCA on just the training data and applies that transformation to both the training and test data.

In [70]:

```
x_train_scaled = ss.fit_transform(x_train)

PCA_start_time = time()

pca2 = PCA(n_components=0.95)
pca2.fit(x_train_scaled)

PCA_elapsed_time = time() - PCA_start_time
print("Time elapsed: {:.3f}".format(PCA_elapsed_time))
print("Number of principal components:", pca2.n_components_)
x_train_reduced2 = pca2.transform(x_train)
x_test_reduced2 = pca2.transform(x_test)
```

```
Time elapsed: 3.809
Number of principal components: 320
```

In [71]:

```
rfc_pca_param_grid2 = {
    'max_depth' : [5,10,20],
    'min_samples_split': [2,5,10],
    'min_samples_leaf': [1,3,7]
}

start_time = time()

rfc_pca2 = RandomForestClassifier(max_features='sqrt', bootstrap=True, n_estimators
=10, random_state=42, n_jobs=-1)
CV_rfc_pca2 = GridSearchCV(estimator=rfc_pca2, param_grid=rfc_pca_param_grid2, cv=5
)
CV_rfc_pca2.fit(x_train_reduced2, y_train)

elapsed_time = time() - start_time
rfc_pca2 = RandomForestClassifier(max_features='sqrt', bootstrap=True, max_depth=20
, min_samples_leaf=3,
                        min_samples_split=10, n_estimators=500, random_state=42, n_
jobs=-1)

# fit model and record time elapsed
rf_pca_start_time = time()
rfc_pca2.fit(x_train_reduced2, y_train)
rf_pca_elapsed_time = time() - rf_pca_start_time
print("Time elapsed: {:.3f}".format(rf_pca_elapsed_time))

# run 5 fold cv
cv_rfc_pca2 = cross_val_score(rfc_pca2, x_train_reduced2, y_train, cv=5, scoring="a
ccuracy")
cv_rfc_pca2
```

```
Time elapsed: 71.652
```

Out[71]:

```
array([0.9477381 , 0.94869048, 0.94261905, 0.94833333, 0.95369048])
```

In [73]:

```
y_train_pred_rf_pca2 = cross_val_predict(rfc_pca2, x_train_reduced2, y_train, cv=5)
f1_score(y_train, y_train_pred_rf_pca2, average="macro")
```

Out[73]:

```
0.9478253305961306
```

In [74]:

```
y_test_pred_rf_pca2 = rfc_pca2.predict(x_test_reduced2)
print(y_test_pred_rf_pca2[:10])

df_rfc_pca2=pd.DataFrame({'ImageID' : pd.Series(range(1,28001)), 'Label': y_test_pr
ed_rf_pca2})
df_rfc_pca2.to_csv('rfc_pca2.csv', index=False)
```

```
[2 0 9 7 3 7 0 3 0 3]
```

kaggle username: Ankita Avadhani, kaggle score 0.94992