Use the data that you used for week 7 assignment, use PCA to reduce the dataset's dimensionality with an explained variance ratio of 95%.

Train a new Random Forest classifier on the reduced dataset and see how long it takes. Was training much faster?

Evaluate the classifier on the test set. How does it compare to the previous classifier that you built in Week 7 assignment?

Load the MNIST dataset

In [ ]:
```
from sklearn.datasets import fetch_openml
import time
```

Get the MNIST data using the below code

In [ ]:
```
mnist = fetch_openml('mnist_784', version=1, as_frame=False)
mnist.keys()
```

Out[ ]:
```
dict_keys(['data', 'target', 'frame', 'categories', 'feature_names', 'target_name
s', 'DESCR', 'details', 'url'])
```

define train and test data

In [ ]:
```
train, test = mnist["data"], mnist["target"]
```

Train Test data details

In [ ]:
```
print(train.shape)
print(test.shape)
```

```
(70000, 784)
(70000,)
```

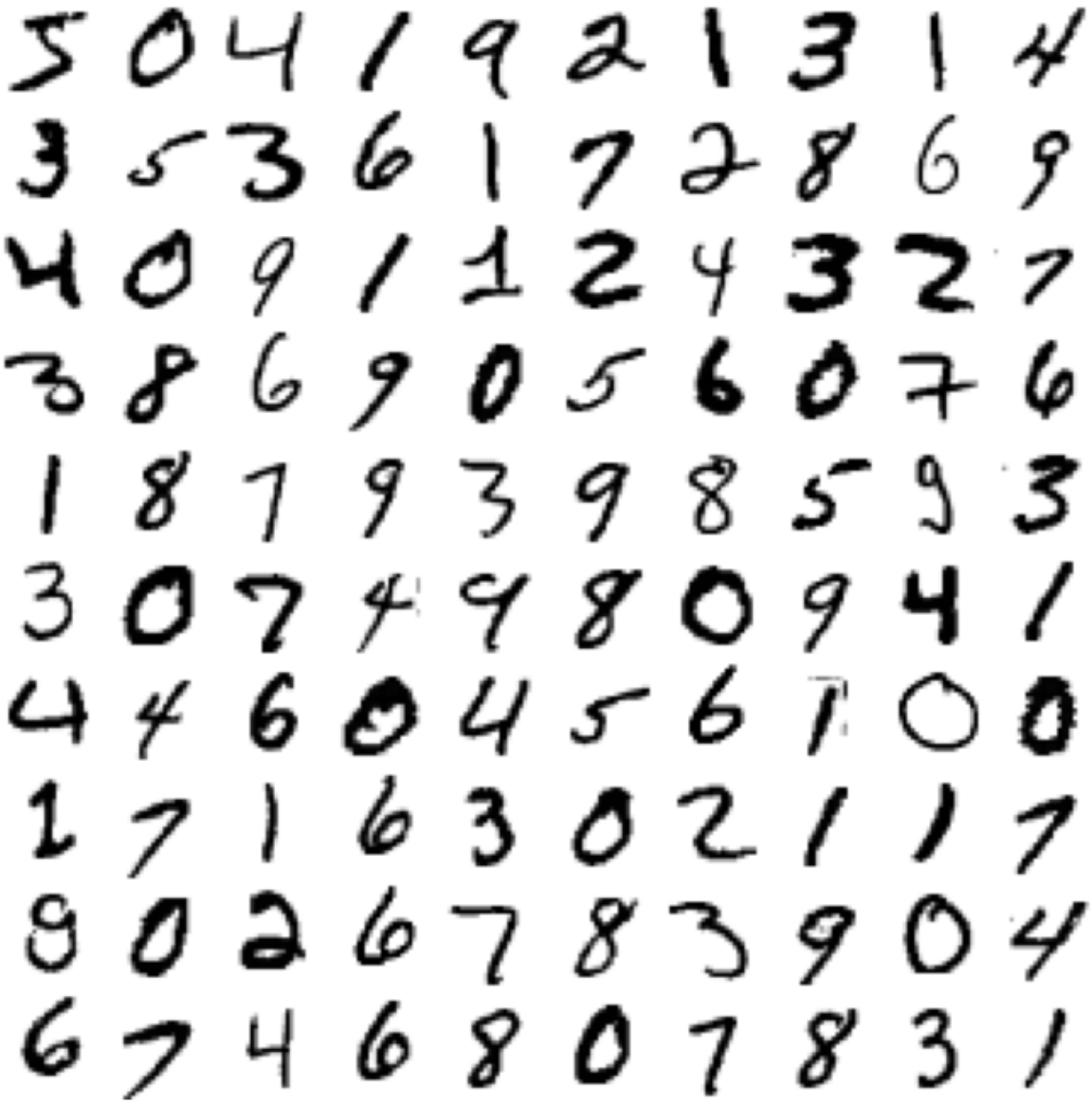
In [ ]:
```
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt

img = train[10000]
image_shape = img.reshape(28, 28)
plt.imshow(image_shape, cmap=mpl.cm.binary)
plt.axis("off")

plt.show()
```

In [ ]:
```
test[10000]
```

Out[ ]:
```
'3'
```

In [ ]:
```
def plot_digits(instances, images_per_row=10, **options):
    size = 28
    images_per_row = min(len(instances), images_per_row)
    n_rows = (len(instances) - 1) // images_per_row + 1
    n_empty = n_rows * images_per_row - len(instances)
    padded_instances = np.concatenate([instances, np.zeros((n_empty, size * size))
    image_grid = padded_instances.reshape((n_rows, images_per_row, size, size))
    big_image = image_grid.transpose(0, 2, 1, 3).reshape(n_rows * size,images_per_r
    plt.imshow(big_image, cmap = mpl.cm.binary, **options)
    plt.axis("off")
```

In [ ]:
```
import numpy as np
plt.figure(figsize=(12,12))
example_images = train[:100]
plot_digits(example_images, images_per_row=10)
plt.show()
```



split data into train(60000) and test (10000)

In [ ]:
```
X_train = mnist['data'][:60000]
y_train = mnist['target'][:60000]

X_test = mnist['data'][60000:]
y_test = mnist['target'][60000:]
```

Train a Random Forest classifier on the dataset and time how long it takes, then evaluate the resulting model on the test set.

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [ ]:
```
time_one_100 = time.time()
rnd_clf = RandomForestClassifier(n_estimators=100, max_leaf_nodes=16, random_state
random_forest_clf=rnd_clf.fit(X_train, y_train)
time_two_100 = time.time()
```

Train a Random Forest classifier on the dataset

In [ ]:
```
print ( "Training time {:.2f}s ".format(time_two_100 - time_one_100))
```

```
Training time 17.88s
```

In [ ]:
```
from sklearn.metrics import accuracy_score
```

evaluate the resulting model on the test set.

In [ ]:
```
y_pred_one = rnd_clf.predict(X_test)
pred_score_100 = accuracy_score(y_test, y_pred_one)
print(pred_score_100)
```

```
0.8245
```

In [ ]:
```
rnd_clf = RandomForestClassifier(n_estimators=200, random_state=42)
```

In [ ]:
```
time_one_200 = time.time()
rnd_clf.fit(X_train, y_train)
time_two_200 = time.time()
```

In [ ]:
```
print("Training time {:.2f}s".format(time_two_200 - time_one_200))
```

```
Training time 128.99s
```

In [ ]:
```
y_pred_two = rnd_clf.predict(X_test)
pred_score_200 = accuracy_score(y_test, y_pred_two)
print(pred_score_200)
```

```
0.9707
```

In [ ]:
```
rnd_clf = RandomForestClassifier(n_estimators=400, random_state=42)
```

In [ ]:
```
time_one_400 = time.time()
rnd_clf.fit(X_train, y_train)
time_two_400 = time.time()
```

In [ ]:
```
print("Training time {:.2f}s".format(time_two_400 - time_one_400))
```

```
Training time 253.97s
```

In [ ]:
```
y_pred_three = rnd_clf.predict(X_test)
pred_score_400 = accuracy_score(y_test, y_pred_three)
print(pred_score_400)
```

```
0.9712
```

In [ ]:
```
rnd_clf7 = RandomForestClassifier(n_estimators=700, max_leaf_nodes=16, random_state
```

In [ ]:
```
time_one_700 = time.time()
rnd_clf.fit(X_train, y_train)
time_two_700 = time.time()
```

In [ ]:
```
print("Training time {:.2f}s".format(time_two_700 - time_one_700))
```

```
Training time 256.29s
```

In [ ]:
```
y_pred_four = rnd_clf.predict(X_test)
pred_score_700 = accuracy_score(y_test, y_pred_four)
print(pred_score_700)
```

```
0.9712
```

In [ ]:
```
print ( "Training time for 100 n_estimator {:.2f}s ".format(time_two_100 - time_one
print("Prediction Score for 100 n_estimator ", pred_score_100)
print ( "\n Training time for 200 n_estimator {:.2f}s ".format(time_two_200 - time_
print("Prediction Score for 200 n_estimator ", pred_score_200)
print ( "\n Training time for 400 n_estimator {:.2f}s ".format(time_two_400 - time_
```

```
print("Prediction Score for 400 n_estimator ", pred_score_400)
print ( "\n Training time for 700 n_estimator {:.2f}s ".format(time_two_700 - time_
print("Prediction Score for 700 n_estimator ", pred_score_700)
```

```
Training time for 100 n_estimator 17.88s
Prediction Score for 100 n_estimator  0.8245

 Training time for 200 n_estimator 128.99s
Prediction Score for 200 n_estimator  0.9707

 Training time for 400 n_estimator 253.97s
Prediction Score for 400 n_estimator  0.9712

 Training time for 700 n_estimator 256.29s
Prediction Score for 700 n_estimator  0.9712
```

its look like increasing n_estimator value increase the run time. but other side its also increase the prediction score.

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

pca = PCA(n_components=0.95)
X_train_reduced = pca.fit_transform(X_train)
```

In [ ]:
```
rnd_clf8 = RandomForestClassifier(n_estimators=100, random_state=42)
```

In [ ]:
```
pca_start_time = time.time()
rnd_clf8.fit(X_train_reduced, y_train)
pca_end_time = time.time()
```

In [ ]:
```
print("Training time {:.2f}s".format(pca_end_time - pca_start_time))
```

```
Training time 163.32s
```

In [ ]:
```
X_test_reduced = pca.transform(X_test)

y_pred_pca = rnd_clf8.predict(X_test_reduced)
pred_score_pca = accuracy_score(y_test, y_pred_pca)
print(pred_score_pca)
```

```
0.9481
```

In [ ]:
```
print ( "Training time for 100 n_estimator {:.2f}s ".format(time_two_100 - time_one
print("Prediction Score for 100 n_estimator ", pred_score_100)
print("\n checking with PCA")
print ( "\n Training time for 100 n_estimator {:.2f}s ".format(pca_end_time - pca_s
print("Prediction Score for 100 n_estimator ", pred_score_pca)
```

```
Training time for 100 n_estimator 17.88s
Prediction Score for 100 n_estimator  0.8245

 checking with PCA

 Training time for 100 n_estimator 163.32s
Prediction Score for 100 n_estimator  0.9481
```

# Here as per result i can say like traing time with 100 estimator is more.

# prediction score with test dataset is increase compare to privios.

In [ ]: