Govardhan Seshasayee

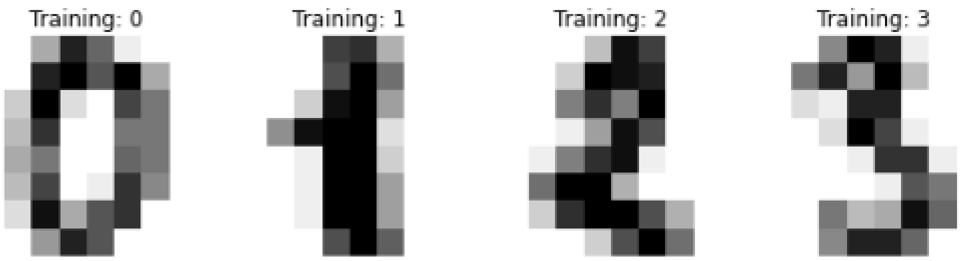
Handwritten Digit Recognition using MNIST Data set Project Report

Digit recognition system is the working of a machine to train itself for recognizing the digits from different sources like emails, bank cheque, papers, images, etc. and in different real-world scenarios for online handwriting recognition on computer tablets or system. Developing such a system includes a machine to understand and classify the images of handwritten digits as 10 digits (0–9). Handwritten digits from the MNIST database has been one of the most famous databases among the machine learning community for many recent decades.

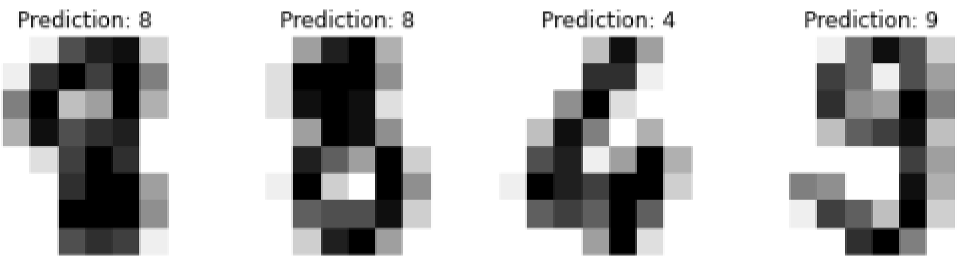
Output:

From the outputs it can be seen that the

Loading the Digits from libraries

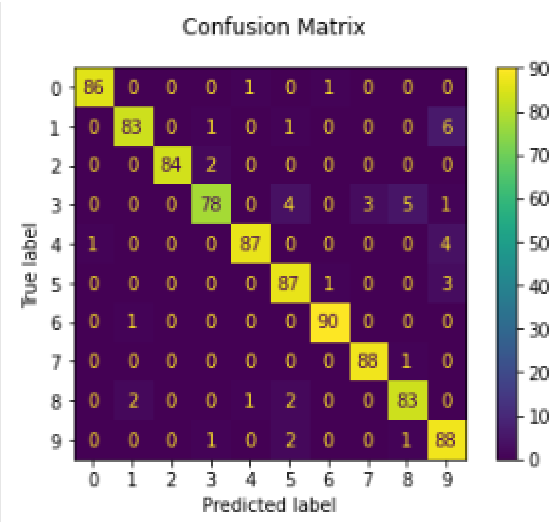


Using MLP the predictions are seen below:



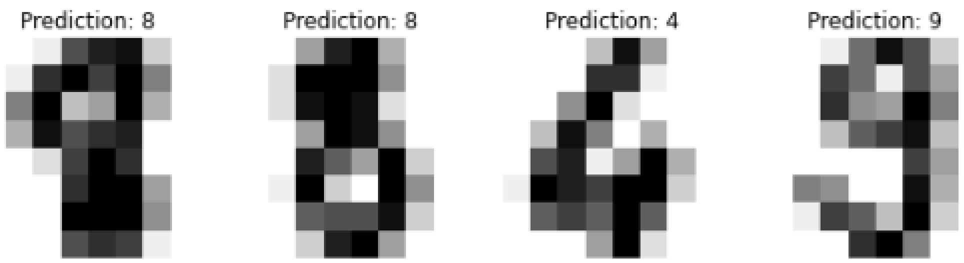
	precision	recall	f1-score	support
0	0.99	0.98	0.98	88
1	0.97	0.91	0.94	91
2	1.00	0.98	0.99	86
3	0.95	0.86	0.90	91
4	0.98	0.95	0.96	92
5	0.91	0.96	0.93	91
6	0.98	0.99	0.98	91
7	0.97	0.99	0.98	89
8	0.92	0.94	0.93	88
9	0.86	0.96	0.91	92
accuracy			0.95	899
macro avg	0.95	0.95	0.95	899
weighted avg	0.95	0.95	0.95	899

Confusion matrix after training and testing



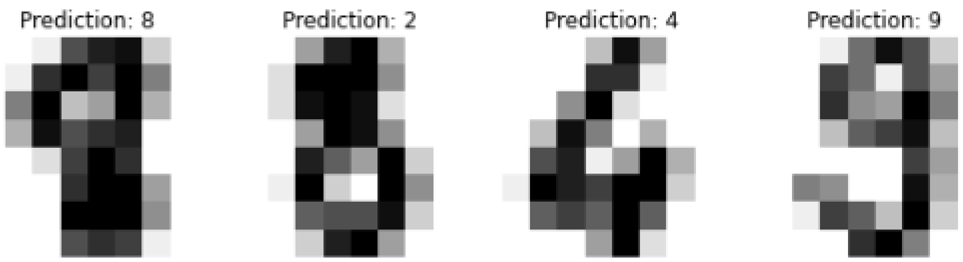
Classification prediction results of Support vector classifier

	precision	recall	f1-score	support
0	1.00	0.99	0.99	88
1	0.99	0.97	0.98	91
2	0.99	0.99	0.99	86
3	0.98	0.87	0.92	91
4	0.99	0.96	0.97	92
5	0.95	0.97	0.96	91
6	0.99	0.99	0.99	91
7	0.96	0.99	0.97	89
8	0.94	1.00	0.97	88
9	0.93	0.98	0.95	92
accuracy			0.97	899
macro avg	0.97	0.97	0.97	899
weighted avg	0.97	0.97	0.97	899



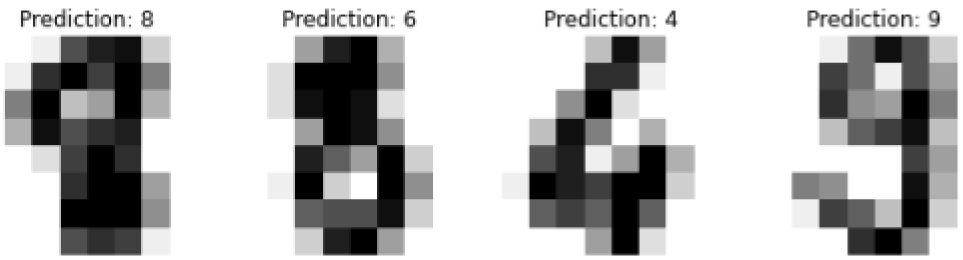
Classification prediction results of Decision tree classifier

	precision	recall	f1-score	support
0	0.92	0.90	0.91	88
1	0.81	0.60	0.69	91
2	0.86	0.71	0.78	86
3	0.72	0.74	0.73	91
4	0.69	0.80	0.74	92
5	0.60	0.77	0.68	91
6	0.85	0.87	0.86	91
7	0.85	0.69	0.76	89
8	0.62	0.65	0.63	88
9	0.66	0.72	0.69	92
accuracy			0.74	899
macro avg	0.76	0.74	0.75	899
weighted avg	0.76	0.74	0.75	899



Classification prediction results of Random Forest classifier

	precision	recall	f1-score	support
0	0.96	0.99	0.97	88
1	0.94	0.88	0.91	91
2	0.99	0.90	0.94	86
3	0.91	0.85	0.87	91
4	0.97	0.91	0.94	92
5	0.92	0.93	0.93	91
6	0.98	0.99	0.98	91
7	0.95	0.98	0.96	89
8	0.88	0.89	0.88	88
9	0.81	0.96	0.88	92
accuracy			0.93	899
macro avg	0.93	0.93	0.93	899
weighted avg	0.93	0.93	0.93	899

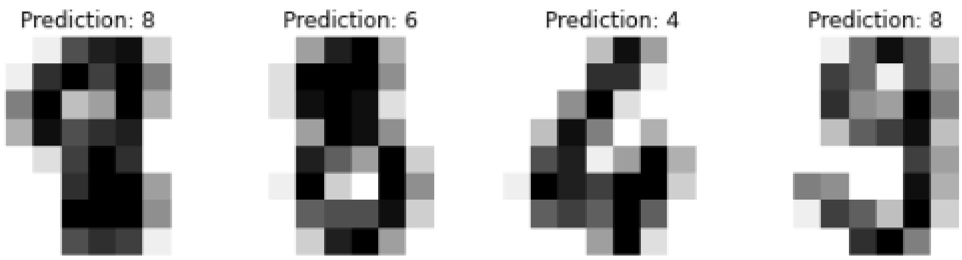


Classification prediction results of Logistic regression

```
              precision    recall  f1-score   support

           0       0.99      0.95      0.97        88
           1       0.94      0.90      0.92        91
           2       0.99      0.98      0.98        86
           3       0.94      0.84      0.88        91
           4       0.98      0.91      0.94        92
           5       0.88      0.95      0.91        91
           6       0.91      0.99      0.95        91
           7       0.98      0.96      0.97        89
           8       0.89      0.90      0.89        88
           9       0.84      0.93      0.89        92

    accuracy                           0.93       899
   macro avg       0.93      0.93      0.93       899
weighted avg       0.93      0.93      0.93       899
```



Input Code:

```python
# Importing Libarires
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
import cv2
from sklearn.datasets import load_digits
from sklearn import preprocessing
from collections import Counter
from skimage.feature import hog
import warnings
warnings.filterwarnings('ignore')
```

```python
#Loading Digits and the training sets
digits =load_digits()

_, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
for ax, image, label in zip(axes, digits.images, digits.target):
    ax.set_axis_off()
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    ax.set_title('Training: %i' % label)
```

```python
n_samples = len(digits.images)
```

```python
data = digits.images.reshape((n_samples,-1))
```

```python
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
```

```python
# Split data into 50% train and 50% test subsets
X_train, X_test, y_train, y_test = train_test_split(
    data, digits.target, test_size=0.5, shuffle=False)
```

```python
Model = MLPClassifier(activation='relu', hidden_layer_sizes=(200, 200), alpha = 0.3)
Model.fit(X_train, y_train)
```

```
MLPClassifier(alpha=0.3, hidden_layer_sizes=(200, 200))
```

```python
print("Training Score :: {}\n".format(Model.score(X_train, y_train)))
print("Testing Score :: {}\n".format(Model.score(X_test, y_test)))
```

```
Training Score :: 1.0

Testing Score :: 0.949944382647386
```

```python
predicted = Model.predict(X_test)
```

```python
from sklearn.metrics import confusion_matrix,classification_report,f1_score,plot_confusion_matrix
```

```python
confusion_matrix(y_test,predicted)
```

```python
print(classification_report(y_test,predicted))
```

```python
_, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
for ax, image, prediction in zip(axes, X_test, predicted):
    ax.set_axis_off()
    image = image.reshape(8, 8)
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    ax.set_title(f'Prediction: {prediction}')
```

```python
disp = plot_confusion_matrix(Model, X_test, y_test)
disp.figure_.suptitle("Confusion Matrix")
print(f"Confusion matrix:\n{disp.confusion_matrix}")

plt.show()
```

```python
from sklearn.svm import SVC
```

```python
clf = SVC(gamma=0.001)
```

```python
clf.fit(X_train,y_train)
pred = clf.predict(X_test)
```

```python
print(classification_report(y_test,pred))
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
clf2 = DecisionTreeClassifier()
```

```
clf2.fit(X_train,y_train)
pred2 = clf2.predict(X_test)
```

```
print(classification_report(y_test,pred2))
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
clf3 = RandomForestClassifier()
clf3.fit(X_train,y_train)
```

```
RandomForestClassifier()
```

```
pred3 = clf3.predict(X_test)
```

```
print(classification_report(y_test,pred3))
```

```
from sklearn.linear_model import LogisticRegression
```

```
clf4 = LogisticRegression()
clf4.fit(X_train,y_train)
pred4 = clf4.predict(X_test)
```

```
print(classification_report(y_test,pred4))
```