main

October 16, 2024

1 Loading, preparing, and visualizing data

1.1 Import MNIST dataset from Keras

[2]:
```
from keras.datasets import mnist
```

1.2 Load the datasets into Python array

[3]:
```
(data_2D_X, data_y), (do_not_use_X, do_not_use_y) = mnist.load_data()
```

1.3 Convert images to a NumPy array

[5]:
```
import numpy as np

data_X = np.array([np.array(image).ravel() for image in data_2D_X])
```

1.4 Split the dataset into training, testing, and validating datasets

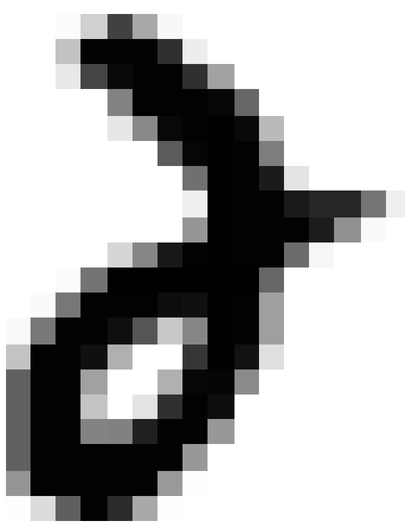
[7]:
```
from sklearn.model_selection import train_test_split

train_test_X, validate_X, train_test_y, validate_y = train_test_split(data_X,␣
↪data_y, test_size=10000, random_state=10)
train_X, test_X, train_y, test_y = train_test_split(train_test_X, train_test_y,␣
↪test_size=10000, random_state=10)

#train_X, train_y, test_X, test_y = modeling_X[:50000], modeling_y[:50000],␣
↪modeling_X[50000:], modeling_y[50000:]
print(train_X.shape)
print(train_y.shape)
print(test_X.shape)
print(test_y.shape)
print(validate_X.shape)
print(validate_y.shape)
```

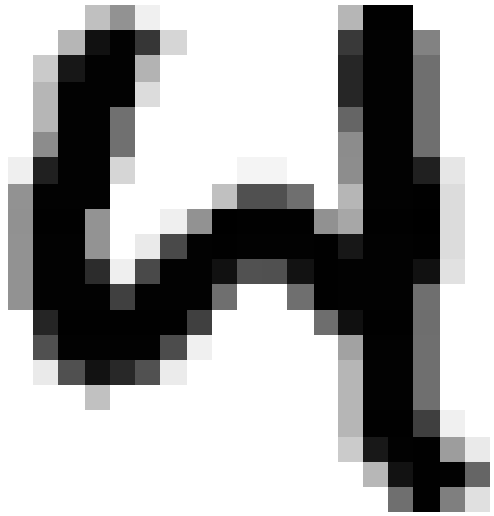
```
(40000, 784)
(40000,)
(10000, 784)
(10000,)
```

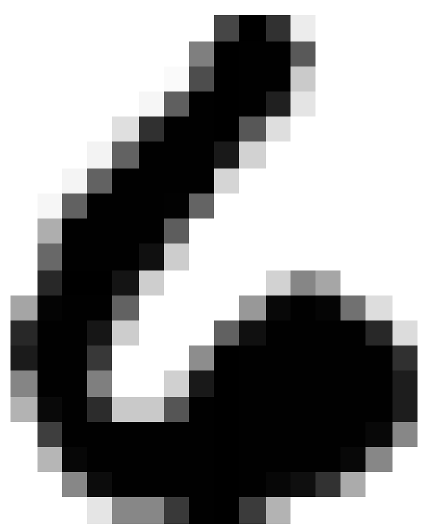
```
(10000, 784)
(10000,)
```

1.5 Visualize the images in the dataset

[8]:
```python
import matplotlib.pyplot as plt

def plot_digit(image_data):
    image = image_data.reshape(28, 28)
    plt.imshow(image, cmap="binary")
    plt.axis("off")

some_digit = train_X[34782]
plot_digit(some_digit)
plt.show()

some_digit = test_X[4783]
plot_digit(some_digit)
plt.show()

some_digit = validate_X[9283]
plot_digit(some_digit)
plt.show()
```







2 Random forest classifier

2.1 Training

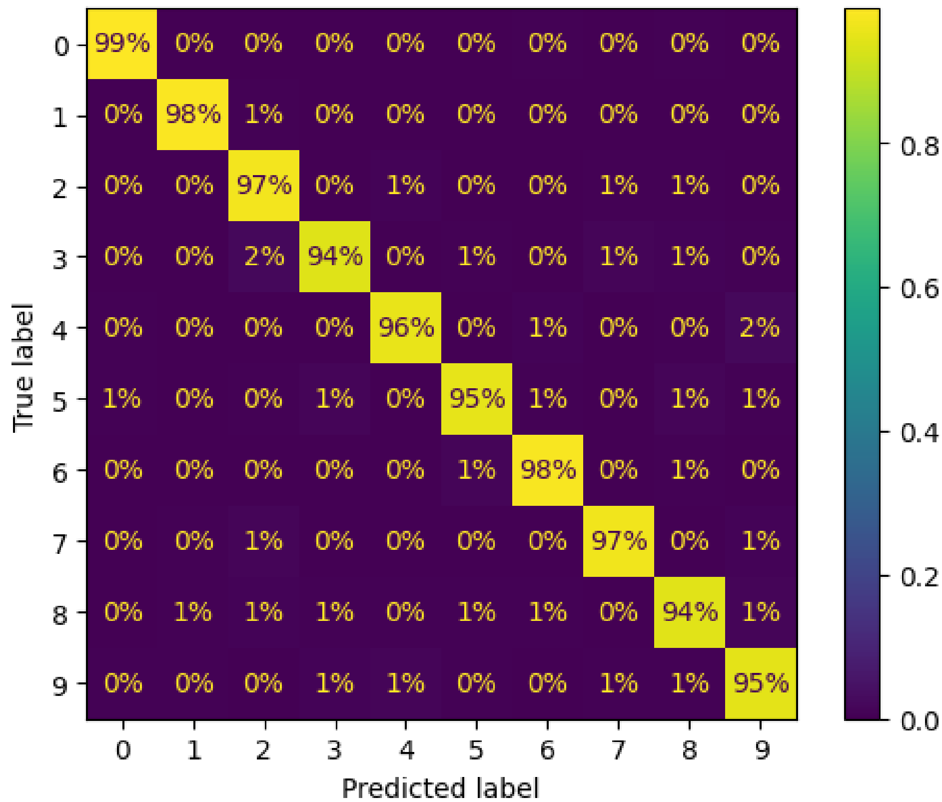
[9]:
```
from sklearn.ensemble import RandomForestClassifier

random_forest_classifier = RandomForestClassifier(n_estimators=100,
                                                   criterion='gini',
                                                   n_jobs=-1,
                                                   random_state=30)
random_forest_classifier.fit(train_X, train_y)
```

[9]:
```
RandomForestClassifier(n_jobs=-1, random_state=30)
```

2.2 Running the model on the test dataset

[10]:
```
from sklearn.metrics import ConfusionMatrixDisplay

print(random_forest_classifier.score(test_X, test_y))

predicted_y = random_forest_classifier.predict(test_X)

ConfusionMatrixDisplay.from_predictions(test_y, predicted_y, normalize="true",␣
 ↪values_format=".0%")
plt.show()
```

```
0.9621
```



3 Extra-trees classifier

3.1 Training

[11]:
```
from sklearn.ensemble import ExtraTreesClassifier

extra_trees_classifier = ExtraTreesClassifier(n_estimators=100,
                                              criterion='gini',
                                              n_jobs=-1,
                                              random_state=40)
extra_trees_classifier.fit(train_X, train_y)
```

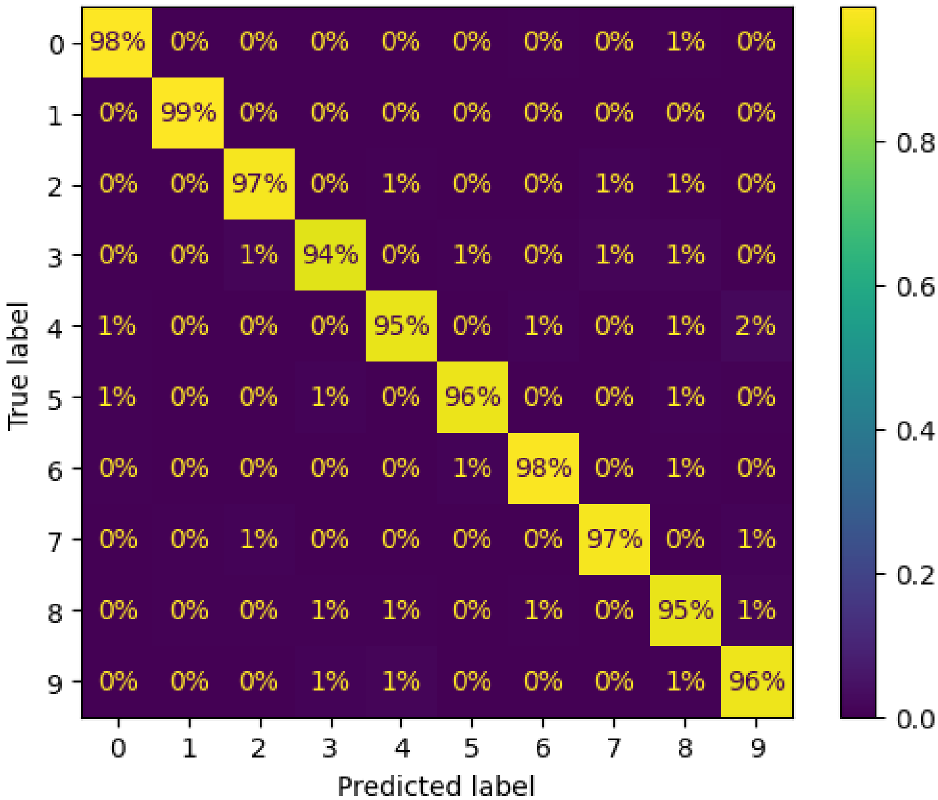
[11]: ExtraTreesClassifier(n_jobs=-1, random_state=40)

3.2 Running the model on the test dataset

[12]:
```
from sklearn.metrics import ConfusionMatrixDisplay

print(extra_trees_classifier.score(test_X, test_y))
```

```
predicted_y = extra_trees_classifier.predict(test_X)

ConfusionMatrixDisplay.from_predictions(test_y, predicted_y, normalize="true",␣
↪values_format=".0%")
plt.show()
```

0.9664



4 Logistical regression model

4.1 Training

[13]:
```
from sklearn.linear_model import LogisticRegression

logistical_regressor = LogisticRegression(penalty='l1',
                                          C=0.01,
                                          random_state=50,
                                          max_iter=10000,
                                          solver='saga',
                                          tol=0.01,
                                          n_jobs=-1)
```

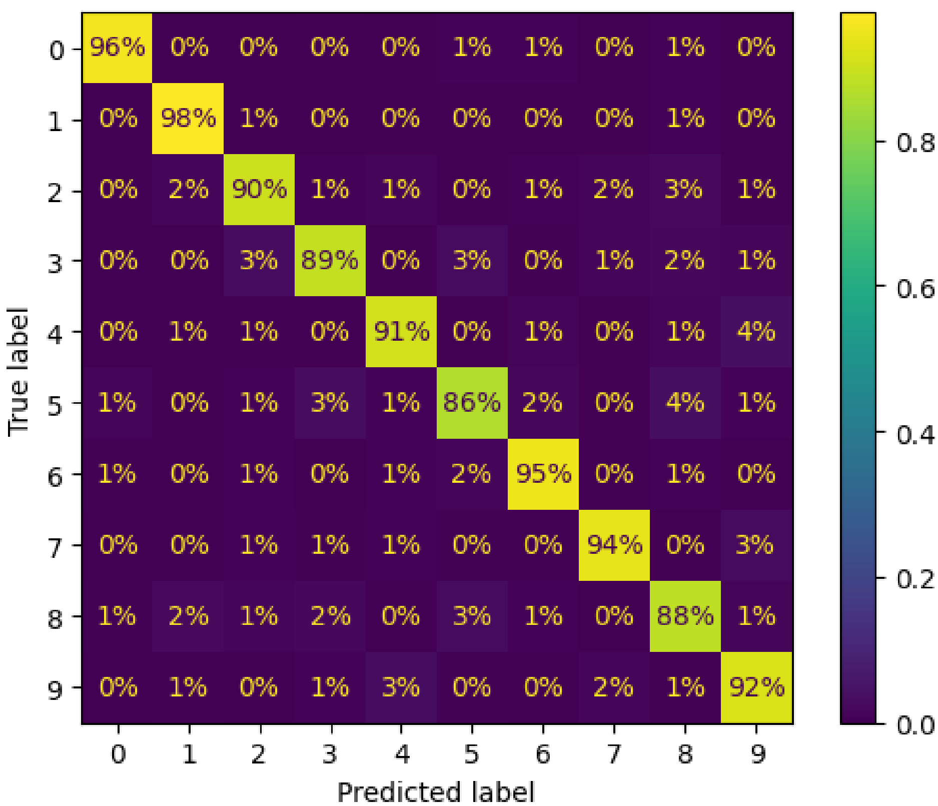
```
logistical_regressor.fit(train_X, train_y)
```

[13]:
```
LogisticRegression(C=0.01, max_iter=10000, n_jobs=-1, penalty='l1',
                   random_state=50, solver='saga', tol=0.01)
```

4.2 Running the model on the test dataset

[14]:
```
from sklearn.metrics import ConfusionMatrixDisplay

print(logistical_regressor.score(test_X, test_y))

predicted_y = logistical_regressor.predict(test_X)

ConfusionMatrixDisplay.from_predictions(test_y, predicted_y, normalize="true",␣
 ↪values_format=".0%")
plt.show()
```

```
0.9199
```



5 Decision tree

5.1 Training

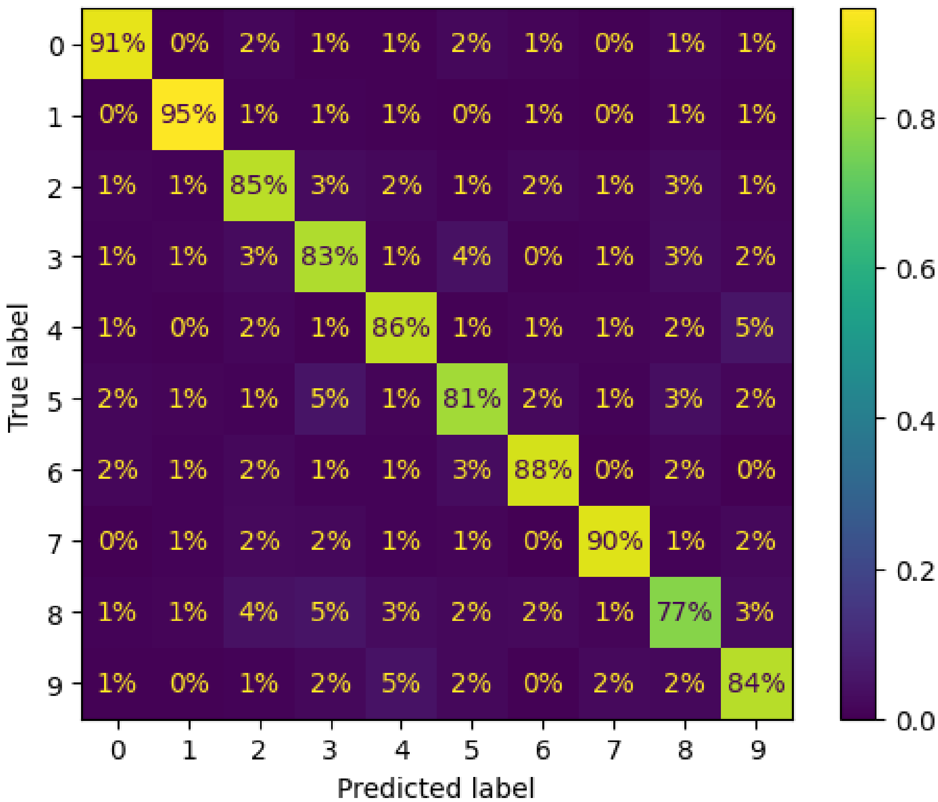
[15]:
```
from sklearn.tree import DecisionTreeClassifier

decision_tree_classifier = DecisionTreeClassifier(criterion='gini',
                                                  min_samples_split=5,
                                                  random_state=60)
decision_tree_classifier.fit(train_X, train_y)
```

[15]:
```
DecisionTreeClassifier(min_samples_split=5, random_state=60)
```

5.2 Running the model on the test dataset

[16]:
```
from sklearn.metrics import ConfusionMatrixDisplay

print(decision_tree_classifier.score(test_X, test_y))

predicted_y = decision_tree_classifier.predict(test_X)

ConfusionMatrixDisplay.from_predictions(test_y, predicted_y, normalize="true",␣
 ↪values_format=".0%")
plt.show()
```

```
0.8626
```



6 Stochastic gradient descent (SGD) classifier

6.1 Preprocessing

[17]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
train_X_scaled = scaler.fit_transform(train_X.astype("float64"))
```

6.2 Training

[18]:
```
from sklearn.linear_model import SGDClassifier

sgd_classifier = SGDClassifier(random_state=70,
                               n_jobs=-1)
sgd_classifier.fit(train_X, train_y)
```

[18]:
```
SGDClassifier(n_jobs=-1, random_state=70)
```

6.3 Cross-evaluation

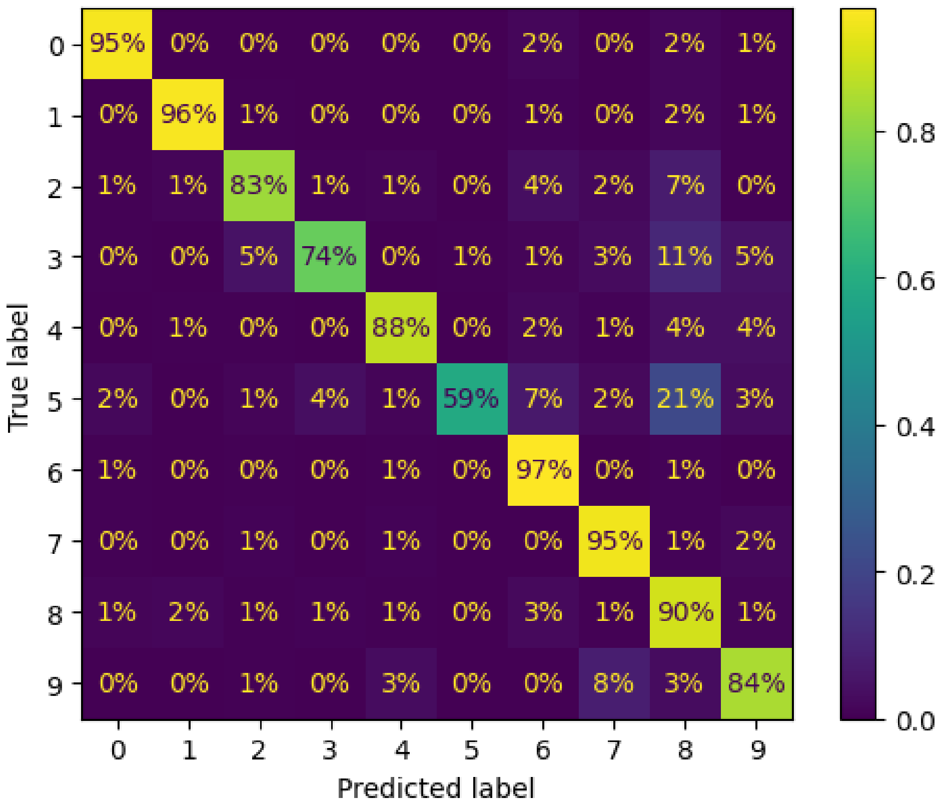
[19]:
```
import pandas as pd
from sklearn.model_selection import cross_val_score

sgd_classifier_accuracy = cross_val_score(sgd_classifier, train_X, train_y,␣
    ↪cv=5, scoring="accuracy")

pd.Series(sgd_classifier_accuracy).describe()
```

[19]:
```
count    5.000000
mean     0.869625
std      0.010678
min      0.856125
25%      0.861250
50%      0.873375
75%      0.875125
max      0.882250
dtype: float64
```

6.4 Running the model on the test dataset

[20]:
```
from sklearn.metrics import ConfusionMatrixDisplay

print(sgd_classifier.score(test_X, test_y))

predicted_y = sgd_classifier.predict(test_X)

ConfusionMatrixDisplay.from_predictions(test_y, predicted_y, normalize="true",␣
    ↪values_format=".0%")
plt.show()
```

```
0.8635
```



7 Ensemble (Using all 5 models)

7.1 Training

[21]:
```python
from sklearn.ensemble import VotingClassifier

hard_vote_classifier_5 = VotingClassifier(
    estimators=[
        ('random_forest_classifier', random_forest_classifier),
        ('extra_trees_classifier', extra_trees_classifier),
        ('logistical_regressor', logistical_regressor),
        ('decision_tree_classifier', decision_tree_classifier),
        ('sgd_classifier', sgd_classifier)],
    voting='hard',
    n_jobs=-1,
)

hard_vote_classifier_5.fit(train_X, train_y)
```

[21]:

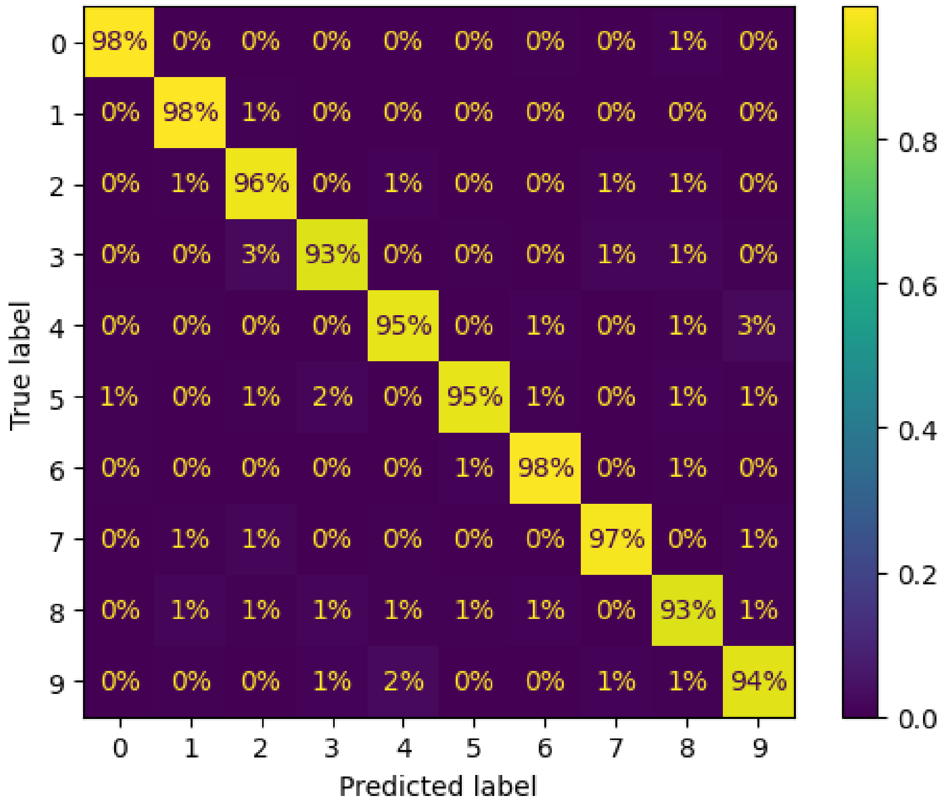
```
VotingClassifier(estimators=[('random_forest_classifier',
                              RandomForestClassifier(n_jobs=-1,
                                                     random_state=30)),
                             ('extra_trees_classifier',
                              ExtraTreesClassifier(n_jobs=-1, random_state=40)),
                             ('logistical_regressor',
                              LogisticRegression(C=0.01, max_iter=10000,
                                                 n_jobs=-1, penalty='l1',
                                                 random_state=50, solver='saga',
                                                 tol=0.01)),
                             ('decision_tree_classifier',
                              DecisionTreeClassifier(min_samples_split=5,
                                                     random_state=60)),
                             ('sgd_classifier',
                              SGDClassifier(n_jobs=-1, random_state=70))],
                 n_jobs=-1)
```

7.2 Running the model on the test dataset

[22]:

```python
from sklearn.metrics import ConfusionMatrixDisplay

print(hard_vote_classifier_5.score(test_X, test_y))

predicted_y = hard_vote_classifier_5.predict(test_X)

ConfusionMatrixDisplay.from_predictions(test_y, predicted_y, normalize="true",␣
 ↪values_format=".0%")
plt.show()
```

```
0.9574
```



7.3 Compare the ensemble model with individual models

[23]:
```
print("Individual estimator's accuracy:")
for estimator in hard_vote_classifier_5.estimators_:
    print("\tAccuracy of {}: {}".format(estimator, estimator.score(test_X,␣
 ↪test_y)))
print("\nEnsemble-5 model's accuracy: {}".format(hard_vote_classifier_5.
 ↪score(test_X, test_y)))
```

```
Individual estimator's accuracy:
        Accuracy of RandomForestClassifier(n_jobs=-1, random_state=30): 0.9621
        Accuracy of ExtraTreesClassifier(n_jobs=-1, random_state=40): 0.9664
        Accuracy of LogisticRegression(C=0.01, max_iter=10000, n_jobs=-1,
penalty='l1',
                   random_state=50, solver='saga', tol=0.01): 0.9199
        Accuracy of DecisionTreeClassifier(min_samples_split=5,
random_state=60): 0.8626
        Accuracy of SGDClassifier(n_jobs=-1, random_state=70): 0.8635

Ensemble-5 model's accuracy: 0.9574
```

The accuracy on the test dataset of the ensemble-5 model is in fact smaller than that of some individual models. We suspect that this is because the last 3 estimators, whose accuracy is significantly lower than that of the first 2, were hurting the ensemble-5 model's accuracy.

8 Ensemble (Using only 3 best models)

8.1 Training

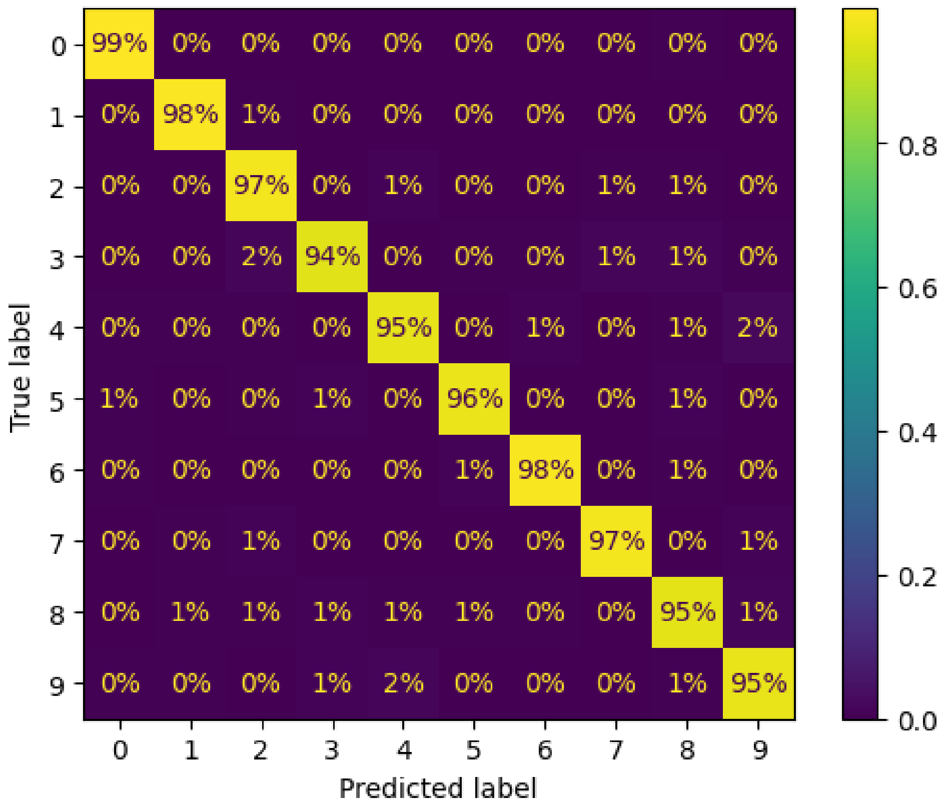
[24]:
```
from sklearn.ensemble import VotingClassifier

hard_vote_classifier_3 = VotingClassifier(
    estimators=[
        ('random_forest_classifier', random_forest_classifier),
        ('extra_trees_classifier', extra_trees_classifier),
        ('logistical_regressor', logistical_regressor)],
    voting='hard',
    n_jobs=-1,
)

hard_vote_classifier_3.fit(train_X, train_y)
```

[24]:
```
VotingClassifier(estimators=[('random_forest_classifier',
                              RandomForestClassifier(n_jobs=-1,
                                                     random_state=30)),
                             ('extra_trees_classifier',
                              ExtraTreesClassifier(n_jobs=-1, random_state=40)),
                             ('logistical_regressor',
                              LogisticRegression(C=0.01, max_iter=10000,
                                                 n_jobs=-1, penalty='l1',
                                                 random_state=50, solver='saga',
                                                 tol=0.01))],
                 n_jobs=-1)
```

8.2 Running the model on the test dataset

[25]:
```
from sklearn.metrics import ConfusionMatrixDisplay

print(hard_vote_classifier_3.score(test_X, test_y))

predicted_y = hard_vote_classifier_3.predict(test_X)

ConfusionMatrixDisplay.from_predictions(test_y, predicted_y, normalize="true",␣
 ↪values_format=".0%")
plt.show()
```

```
0.9641
```



8.3 Compare the ensemble model with individual models

[26]:
```
print("Individual estimator's accuracy:")
for estimator in hard_vote_classifier_3.estimators_:
    print("\tAccuracy of {}: {}".format(estimator, estimator.score(test_X,␣
↪test_y)))
print("\nEnsemble-5 model's accuracy: {}".format(hard_vote_classifier_3.
↪score(test_X, test_y)))
```

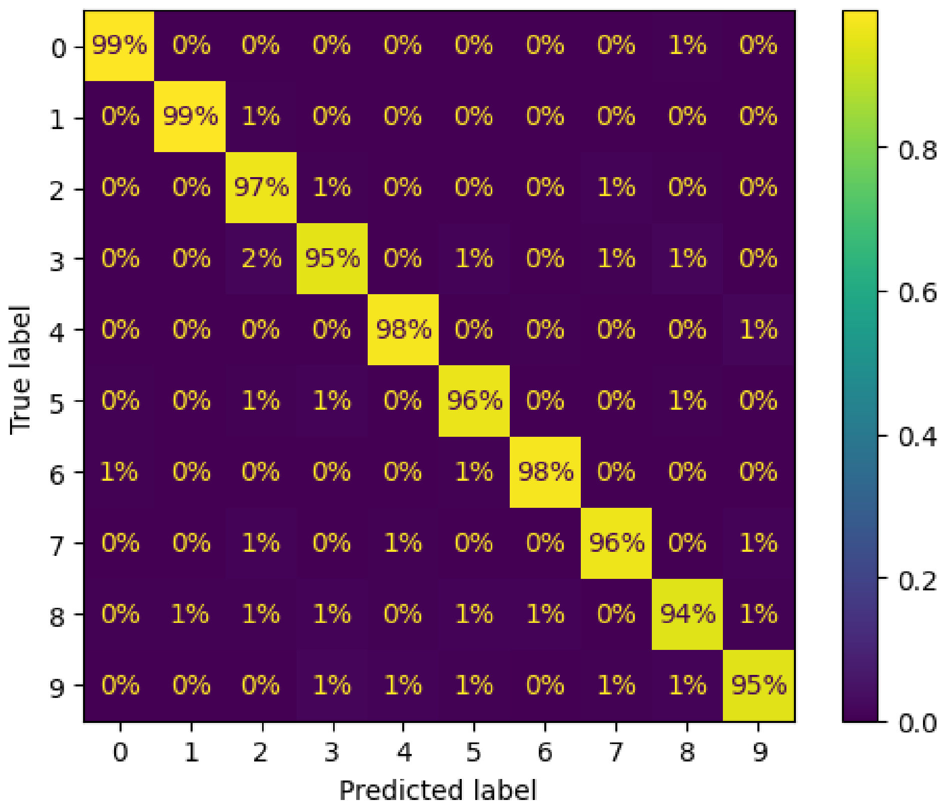
```
Individual estimator's accuracy:
        Accuracy of RandomForestClassifier(n_jobs=-1, random_state=30): 0.9621
        Accuracy of ExtraTreesClassifier(n_jobs=-1, random_state=40): 0.9664
        Accuracy of LogisticRegression(C=0.01, max_iter=10000, n_jobs=-1,
penalty='l1',
                   random_state=50, solver='saga', tol=0.01): 0.9199

Ensemble-5 model's accuracy: 0.9641
```

The accuracy on the test dataset of the ensemble-3 model improves, but this figure is still slightly smaller than that of the Extra-Trees Classifer. This is because even though we have eliminated the worst two models (whose accuracy was less than 0.90), the 0.92-accuracy of the logistical_estimator still hurts the ensemble's performance. Thus, this model can potentially be made better by improving the logistical_regressor.

9 Applying the ensemble model on the validation dataset

[27]:
```
from sklearn.metrics import ConfusionMatrixDisplay

print(hard_vote_classifier_3.score(validate_X, validate_y))

hard_vote_classifier_prediction = hard_vote_classifier_3.predict(validate_X)
ConfusionMatrixDisplay.from_predictions(validate_y,␣
↪hard_vote_classifier_prediction, normalize="true", values_format=".0%")
plt.show()
```

```
0.9671
```



This ensemble model achieves 96.71% accuracy on the validation dataset, with at least 94% recall rate for each class!

10 Saving the model

[28]:
```
import joblib
joblib.dump(hard_vote_classifier_3, "hard_vote_classifier_3_model.pkl")
```

[28]:
```
['hard_vote_classifier_3_model.pkl']
```

11 Acknowledgment

My code is inspired from Aurélien Géron's book "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd Edition"