pr-9

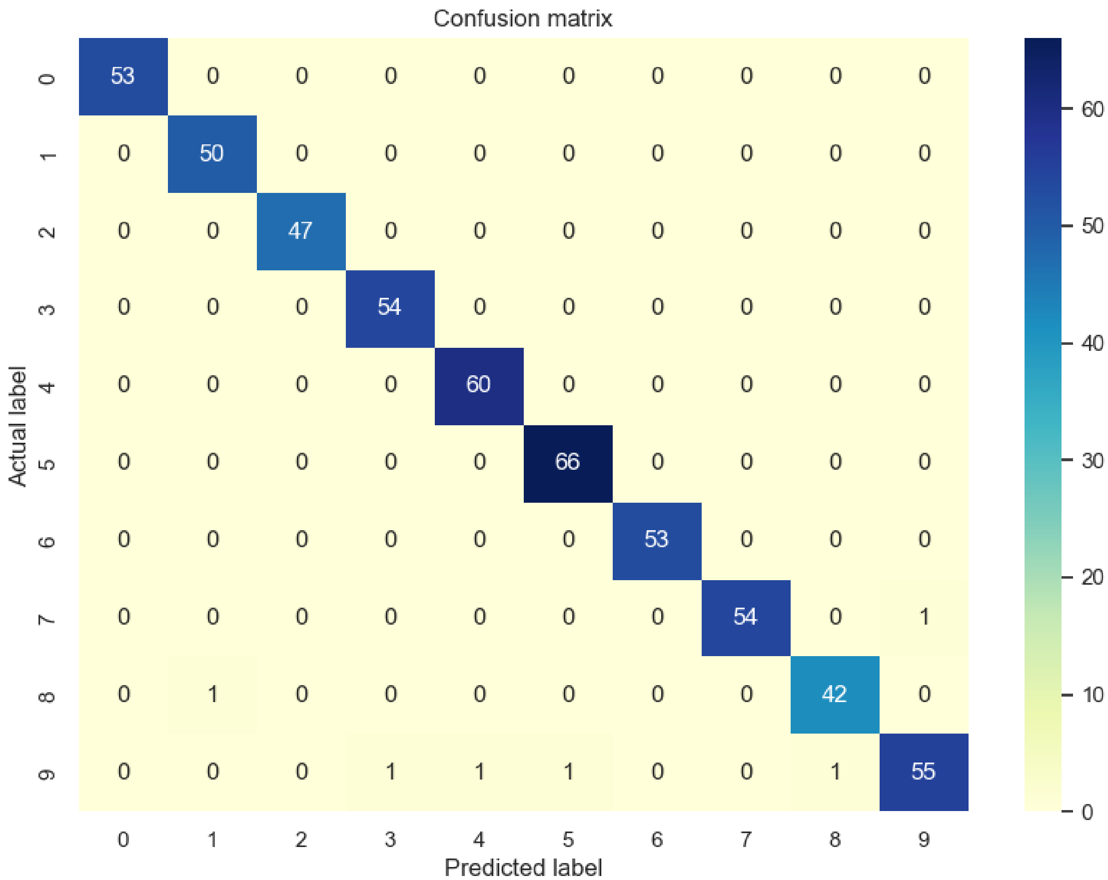
October 15, 2023

1 Classification algorithms: KNN, Logistic Regression, Decision Tree, Random Forest, SVM, Naive Bayes, Neural Network.

[17]:
```
# Import necessary libraries
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()  # for plot styling
import pandas as pd

# Load iris dataset
iris = datasets.load_digits()
X = iris.data
y = iris.target

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42) # 70% training and 30% testing

# Create KNN Classifier
knn = KNeighborsClassifier(n_neighbors=3) # 3 is an example, you might want to␣
↪use cross-validation to find the best parameter

# Train the model using the training sets
knn.fit(X_train, y_train)

# Predict the response for test dataset
y_pred = knn.predict(X_test)

# Model evaluation
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)

print(f"Confusion Matrix:\n{conf_matrix}")
print(f"Accuracy: {accuracy}")
```

```python
# Plotting confusion matrix
plt.figure(figsize=(10,7))
sns.heatmap(pd.DataFrame(conf_matrix, index=iris.target_names, columns=iris.
 ↪target_names), annot=True, cmap="YlGnBu", fmt='g')
plt.title('Confusion matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

```
Confusion Matrix:
[[53  0  0  0  0  0  0  0  0  0]
 [ 0 50  0  0  0  0  0  0  0  0]
 [ 0  0 47  0  0  0  0  0  0  0]
 [ 0  0  0 54  0  0  0  0  0  0]
 [ 0  0  0  0 60  0  0  0  0  0]
 [ 0  0  0  0  0 66  0  0  0  0]
 [ 0  0  0  0  0  0 53  0  0  0]
 [ 0  0  0  0  0  0  0 54  0  1]
 [ 0  1  0  0  0  0  0  0 42  0]
 [ 0  0  0  1  1  1  0  0  1 55]]
Accuracy: 0.9888888888888889
```



[18]:

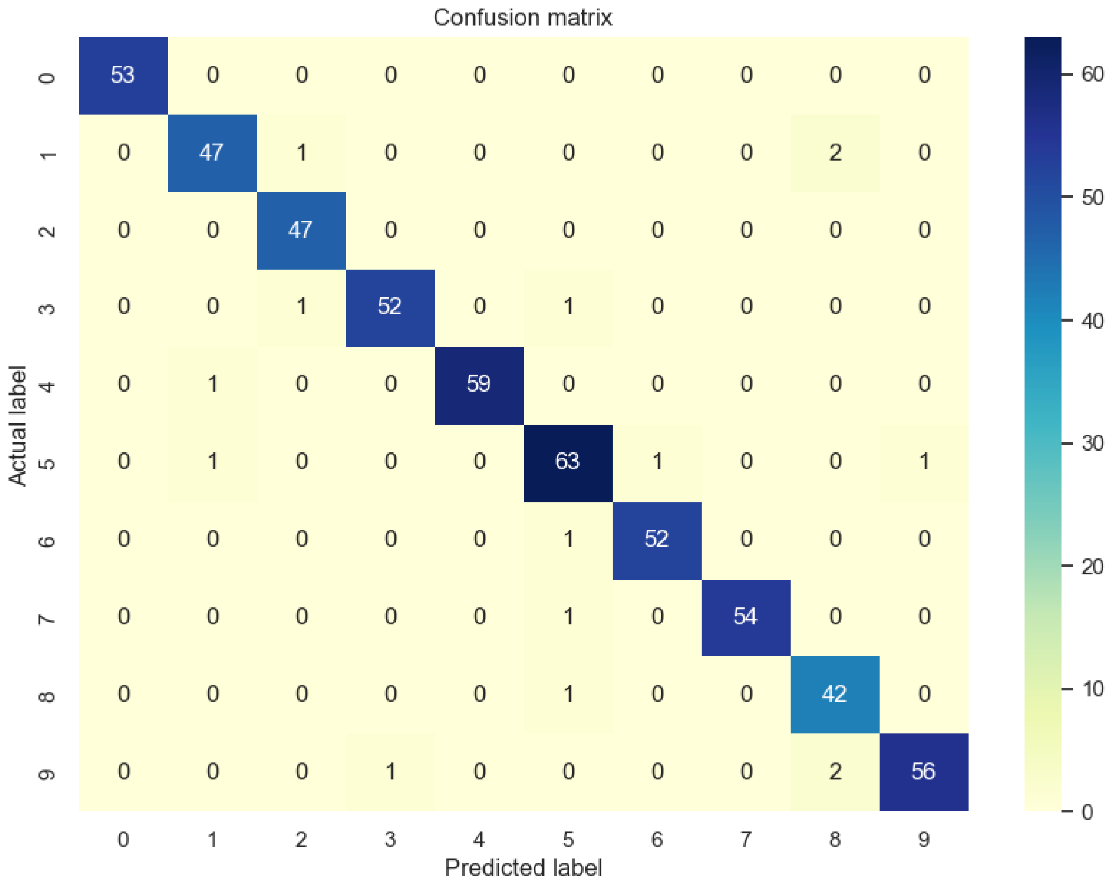
```
# Import necessary libraries
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()  # for plot styling
import pandas as pd

# Load iris dataset
iris = datasets.load_digits()
X = iris.data
y = iris.target

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42) # 70% training and 30% testing

# Create Logistic Regression classifier
```

```python
logistic_regression = LogisticRegression(max_iter=1000)  # Increased max_iter␣
↪may be necessary as logistic regression may need more iterations to converge

# Train the model using the training sets
logistic_regression.fit(X_train, y_train)

# Predict the response for test dataset
y_pred = logistic_regression.predict(X_test)

# Model Evaluation
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)

print(f"Confusion Matrix:\n{conf_matrix}")
print(f"Accuracy: {accuracy}")

# Plotting confusion matrix
plt.figure(figsize=(10,7))
sns.heatmap(pd.DataFrame(conf_matrix, index=iris.target_names, columns=iris.
↪target_names), annot=True, cmap="YlGnBu", fmt='g')
plt.title('Confusion matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

```
/opt/homebrew/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
Confusion Matrix:
[[53  0  0  0  0  0  0  0  0  0]
 [ 0 47  1  0  0  0  0  0  2  0]
 [ 0  0 47  0  0  0  0  0  0  0]
 [ 0  0  1 52  0  1  0  0  0  0]
 [ 0  1  0  0 59  0  0  0  0  0]
 [ 0  1  0  0  0 63  1  0  0  1]
 [ 0  0  0  0  0  1 52  0  0  0]
 [ 0  0  0  0  0  1  0 54  0  0]
 [ 0  0  0  0  0  1  0  0 42  0]
 [ 0  0  0  1  0  0  0  0  2 56]]
```

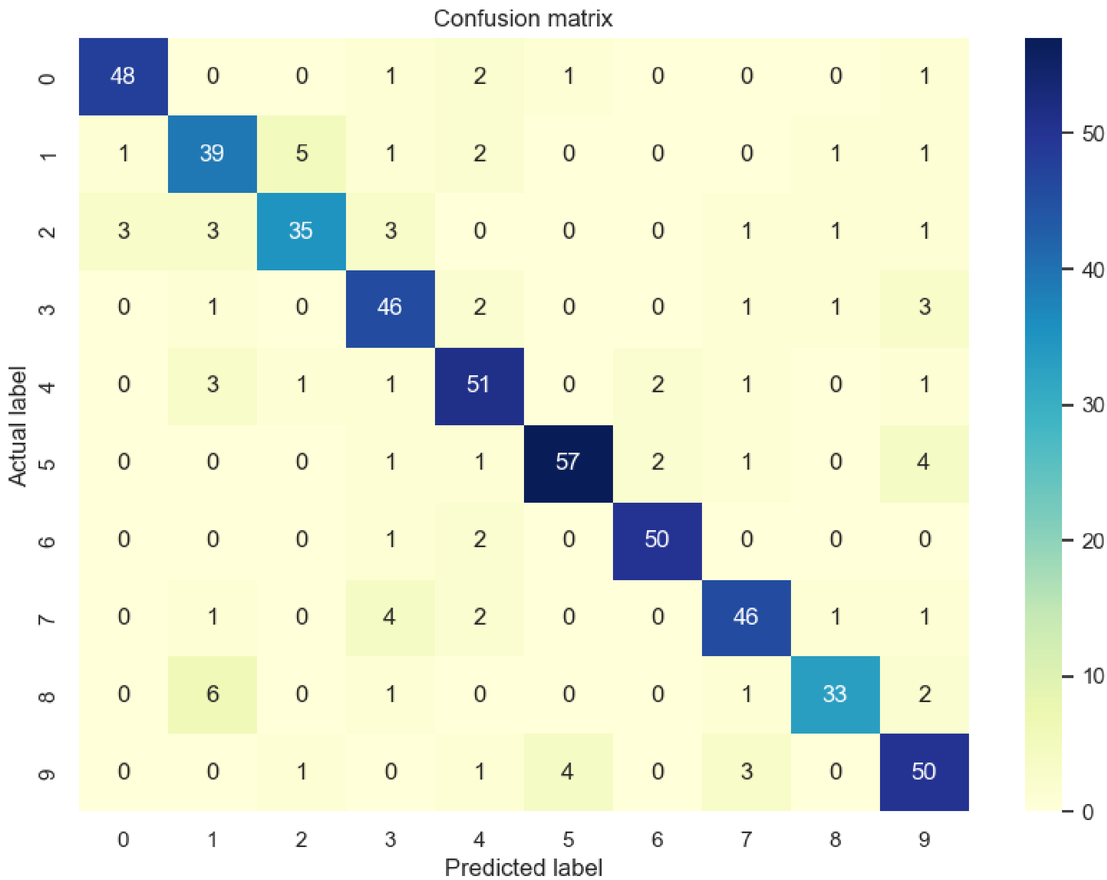
```
Accuracy: 0.9722222222222222
```



[20]:
```python
# Import necessary libraries
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
from sklearn import tree
import seaborn as sns
import pandas as pd

# Load iris dataset
iris = load_digits()
X = iris.data
y = iris.target

# Split dataset into training set and test set
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42) # 70% training and 30% testing

# Create Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=42)

# Train the model using the training sets
clf.fit(X_train, y_train)

# Predict the response for test dataset
y_pred = clf.predict(X_test)

# Model Evaluation
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)

print(f"Confusion Matrix:\n{conf_matrix}")
print(f"Accuracy: {accuracy}")

# Plotting confusion matrix
plt.figure(figsize=(10,7))
sns.heatmap(pd.DataFrame(conf_matrix, index=iris.target_names, columns=iris.
↪target_names), annot=True, cmap="YlGnBu", fmt='g')
plt.title('Confusion matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

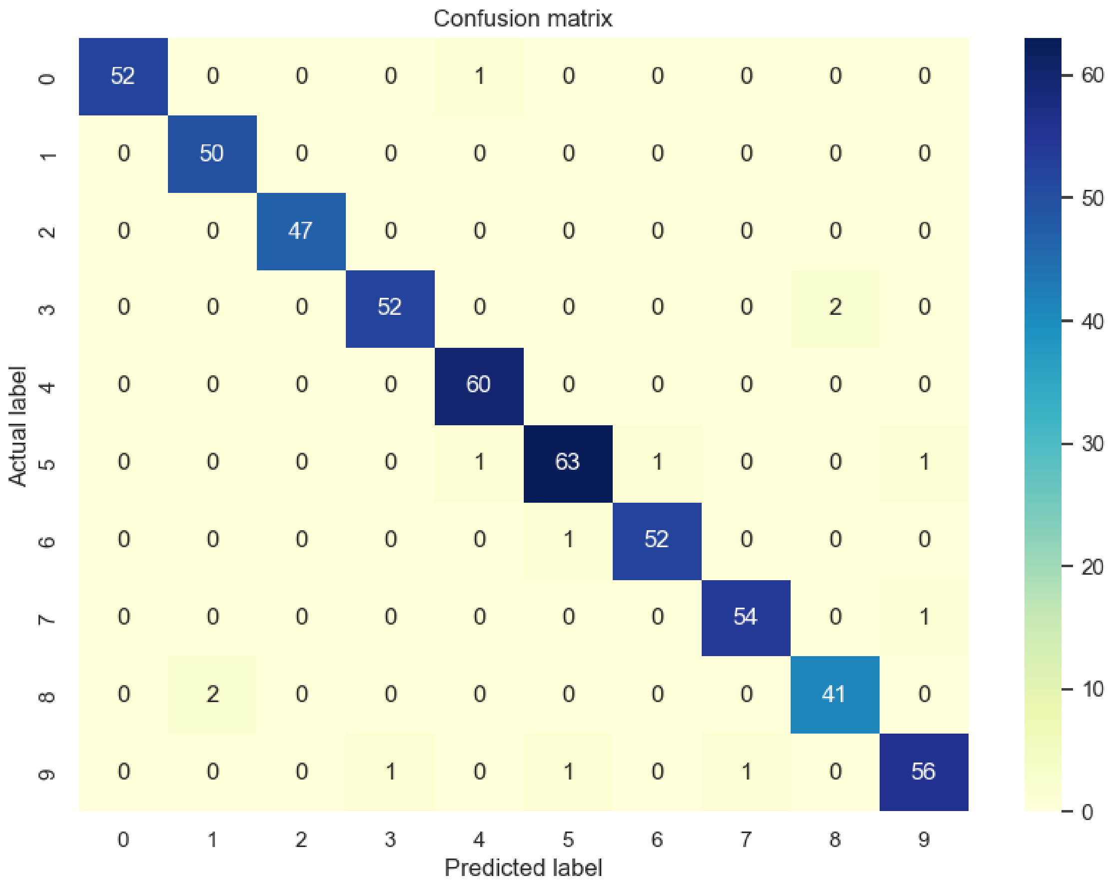
```
Confusion Matrix:
[[48  0  0  1  2  1  0  0  0  1]
 [ 1 39  5  1  2  0  0  0  1  1]
 [ 3  3 35  3  0  0  0  1  1  1]
 [ 0  1  0 46  2  0  0  1  1  3]
 [ 0  3  1  1 51  0  2  1  0  1]
 [ 0  0  0  1  1 57  2  1  0  4]
 [ 0  0  0  1  2  0 50  0  0  0]
 [ 0  1  0  4  2  0  0 46  1  1]
 [ 0  6  0  1  0  0  0  1 33  2]
 [ 0  0  1  0  1  4  0  3  0 50]]
Accuracy: 0.8425925925925926
```



[21]:
```
# Import necessary libraries
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Load iris dataset
iris = load_digits()
X = iris.data
y = iris.target

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42) # 70% training and 30% testing

# Create a Random Forest Classifier
```

```
rf = RandomForestClassifier(n_estimators=100, random_state=42)  # n_estimators␣
↪= number of trees in the forest

# Train the model using the training sets
rf.fit(X_train, y_train)

# Predict the response for test dataset
y_pred = rf.predict(X_test)

# Model Evaluation
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)

print(f"Confusion Matrix:\n{conf_matrix}")
print(f"Accuracy: {accuracy}")

# Plotting confusion matrix
plt.figure(figsize=(10,7))
sns.heatmap(pd.DataFrame(conf_matrix, index=iris.target_names, columns=iris.
↪target_names), annot=True, cmap="YlGnBu", fmt='g')
plt.title('Confusion matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

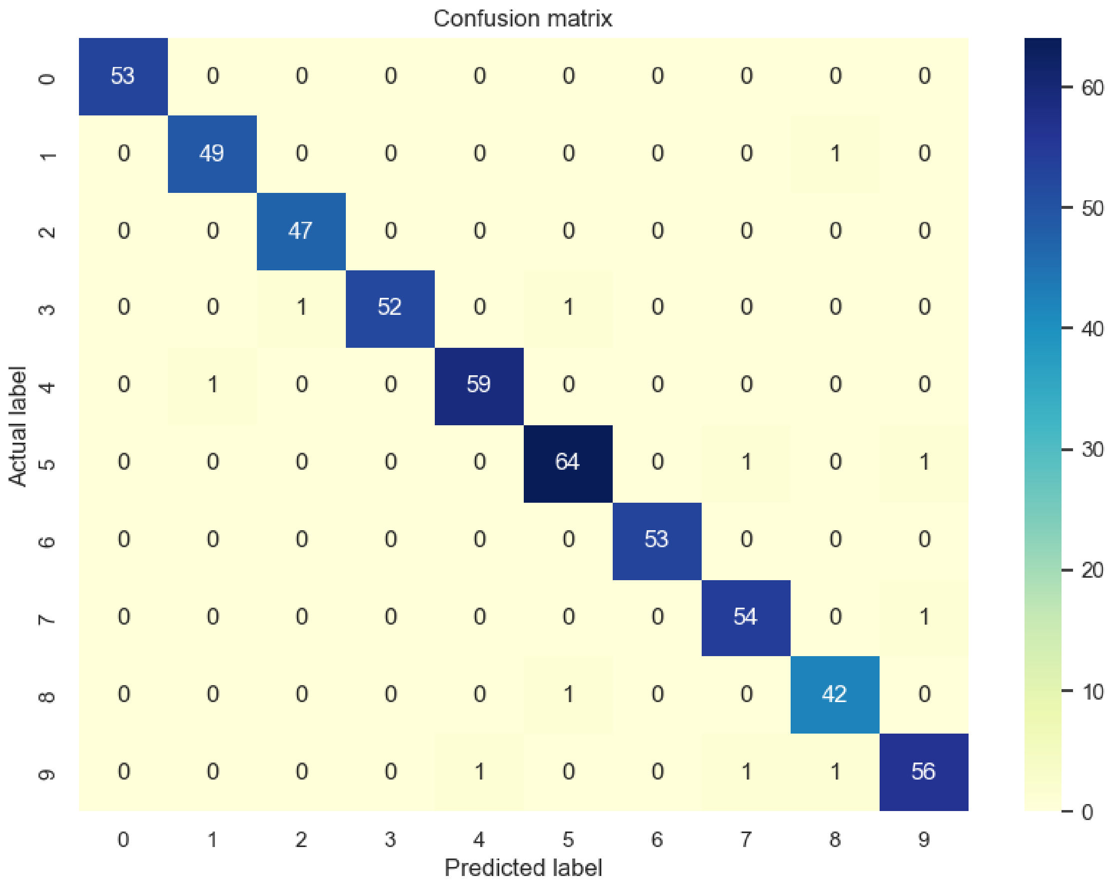
```
Confusion Matrix:
[[52  0  0  0  1  0  0  0  0  0]
 [ 0 50  0  0  0  0  0  0  0  0]
 [ 0  0 47  0  0  0  0  0  0  0]
 [ 0  0  0 52  0  0  0  0  2  0]
 [ 0  0  0  0 60  0  0  0  0  0]
 [ 0  0  0  0  1 63  1  0  0  1]
 [ 0  0  0  0  0  1 52  0  0  0]
 [ 0  0  0  0  0  0  0 54  0  1]
 [ 0  2  0  0  0  0  0  0 41  0]
 [ 0  0  0  1  0  1  0  1  0 56]]
Accuracy: 0.975925925925926
```



[22]:
```
# Import necessary libraries
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Load iris dataset
iris = load_digits()
X = iris.data
y = iris.target

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42) # 70% training and 30% testing

# Create a Support Vector Machine Classifier
```

```python
svc = SVC(kernel='linear')  # You can try other kernels like 'rbf', 'poly', etc.

# Train the model using the training sets
svc.fit(X_train, y_train)

# Predict the response for test dataset
y_pred = svc.predict(X_test)

# Model Evaluation
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)

print(f"Confusion Matrix:\n{conf_matrix}")
print(f"Accuracy: {accuracy}")

# Plotting confusion matrix
plt.figure(figsize=(10,7))
sns.heatmap(pd.DataFrame(conf_matrix, index=iris.target_names, columns=iris.
 ↪target_names), annot=True, cmap="YlGnBu", fmt='g')
plt.title('Confusion matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

```
Confusion Matrix:
[[53  0  0  0  0  0  0  0  0  0]
 [ 0 49  0  0  0  0  0  0  1  0]
 [ 0  0 47  0  0  0  0  0  0  0]
 [ 0  0  1 52  0  1  0  0  0  0]
 [ 0  1  0  0 59  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  1  0  1]
 [ 0  0  0  0  0  0 53  0  0  0]
 [ 0  0  0  0  0  0  0 54  0  1]
 [ 0  0  0  0  0  1  0  0 42  0]
 [ 0  0  0  0  1  0  0  1  1 56]]
Accuracy: 0.9796296296296296
```

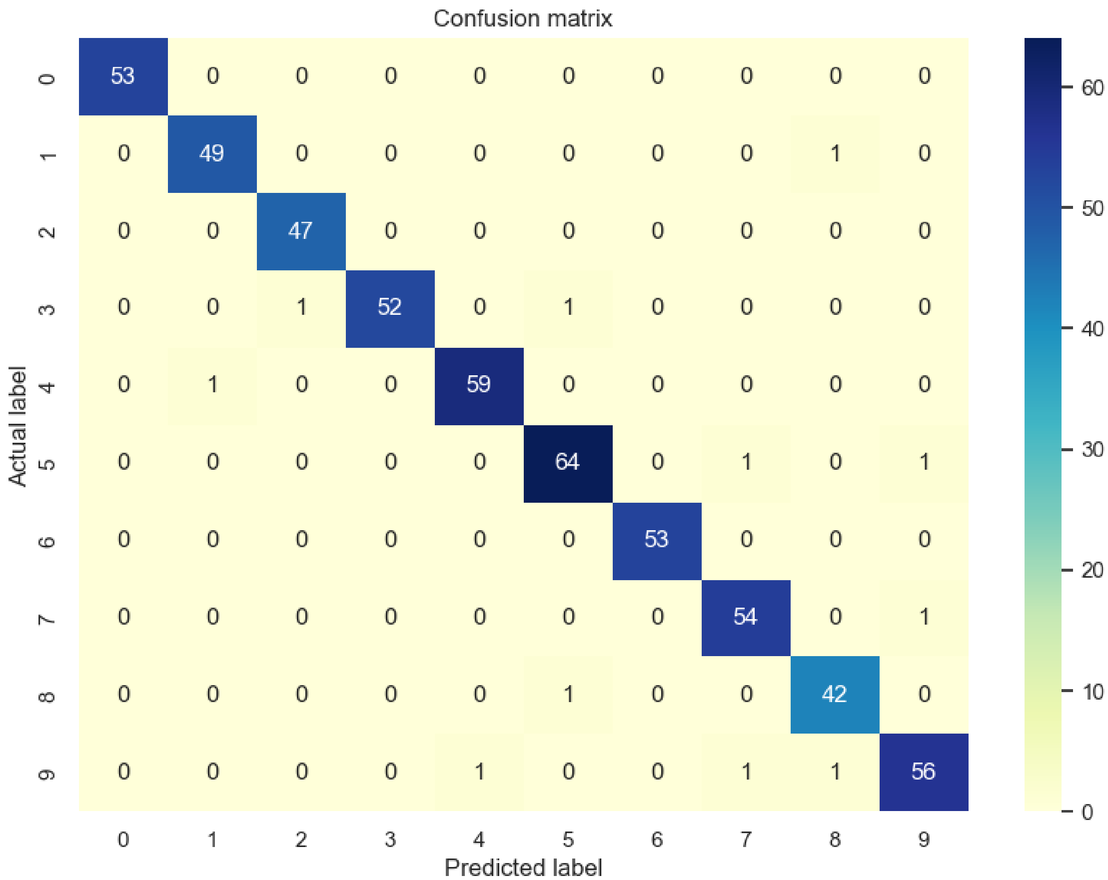


[23]:

```
# Import necessary libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Load iris dataset
iris = load_digits()
X = iris.data
y = iris.target

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42) # 70% training and 30% testing

# Create a Support Vector Machine Classifier
```

```python
svc = SVC(kernel='linear')  # You can try other kernels like 'rbf', 'poly', etc.

# Train the model using the training sets
svc.fit(X_train, y_train)

# Predict the response for test dataset
y_pred = svc.predict(X_test)

# Model Evaluation
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)

print(f"Confusion Matrix:\n{conf_matrix}")
print(f"Accuracy: {accuracy}")

# Plotting confusion matrix
plt.figure(figsize=(10,7))
sns.heatmap(pd.DataFrame(conf_matrix, index=iris.target_names, columns=iris.
 ↪target_names), annot=True, cmap="YlGnBu", fmt='g')
plt.title('Confusion matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

```
Confusion Matrix:
[[53  0  0  0  0  0  0  0  0  0]
 [ 0 49  0  0  0  0  0  0  1  0]
 [ 0  0 47  0  0  0  0  0  0  0]
 [ 0  0  1 52  0  1  0  0  0  0]
 [ 0  1  0  0 59  0  0  0  0  0]
 [ 0  0  0  0  0 64  0  1  0  1]
 [ 0  0  0  0  0  0 53  0  0  0]
 [ 0  0  0  0  0  0  0 54  0  1]
 [ 0  0  0  0  0  1  0  0 42  0]
 [ 0  0  0  0  1  0  0  1  1 56]]
Accuracy: 0.9796296296296296
```

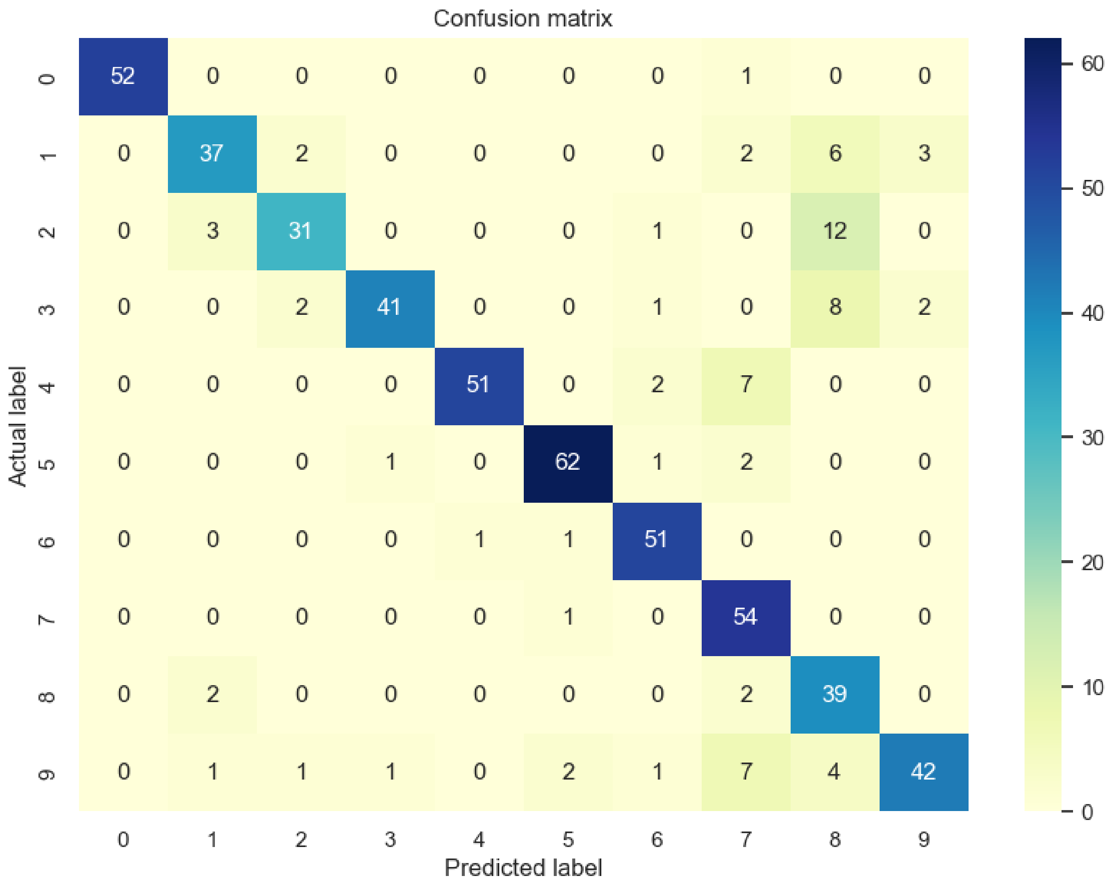


[24]:

```
# Import necessary libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB  # for Gaussian Naive Bayes
from sklearn.metrics import confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Load iris dataset
iris = load_digits()
X = iris.data
y = iris.target

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42) # 70% training and 30% testing

# Create a Gaussian Naive Bayes Classifier
```

```
gnb = GaussianNB()

# Train the model using the training sets
gnb.fit(X_train, y_train)

# Predict the response for test dataset
y_pred = gnb.predict(X_test)

# Model Evaluation
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)

print(f"Confusion Matrix:\n{conf_matrix}")
print(f"Accuracy: {accuracy}")

# Plotting confusion matrix
plt.figure(figsize=(10,7))
sns.heatmap(pd.DataFrame(conf_matrix, index=iris.target_names, columns=iris.
 ↪target_names), annot=True, cmap="YlGnBu", fmt='g')
plt.title('Confusion matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

```
Confusion Matrix:
[[52  0  0  0  0  0  0  1  0  0]
 [ 0 37  2  0  0  0  0  2  6  3]
 [ 0  3 31  0  0  0  1  0 12  0]
 [ 0  0  2 41  0  0  1  0  8  2]
 [ 0  0  0  0 51  0  2  7  0  0]
 [ 0  0  0  1  0 62  1  2  0  0]
 [ 0  0  0  0  1  1 51  0  0  0]
 [ 0  0  0  0  0  1  0 54  0  0]
 [ 0  2  0  0  0  0  0  2 39  0]
 [ 0  1  1  1  0  2  1  7  4 42]]
Accuracy: 0.8518518518518519
```



Confusion matrix
Actual label
Predicted label
0
1
2
3
4
5
6
7
8
9
52 0 0 0 0 0 0 1 0 0
0 37 2 0 0 0 0 2 6 3
0 3 31 0 0 0 1 0 12 0
0 0 2 41 0 0 1 0 8 2
0 0 0 0 51 0 2 7 0 0
0 0 0 1 0 62 1 2 0 0
0 0 0 0 1 1 51 0 0 0
0 0 0 0 0 1 0 54 0 0
0 2 0 0 0 0 0 2 39 0
0 1 1 1 0 2 1 7 4 42
60
50
40
30
20
10
0