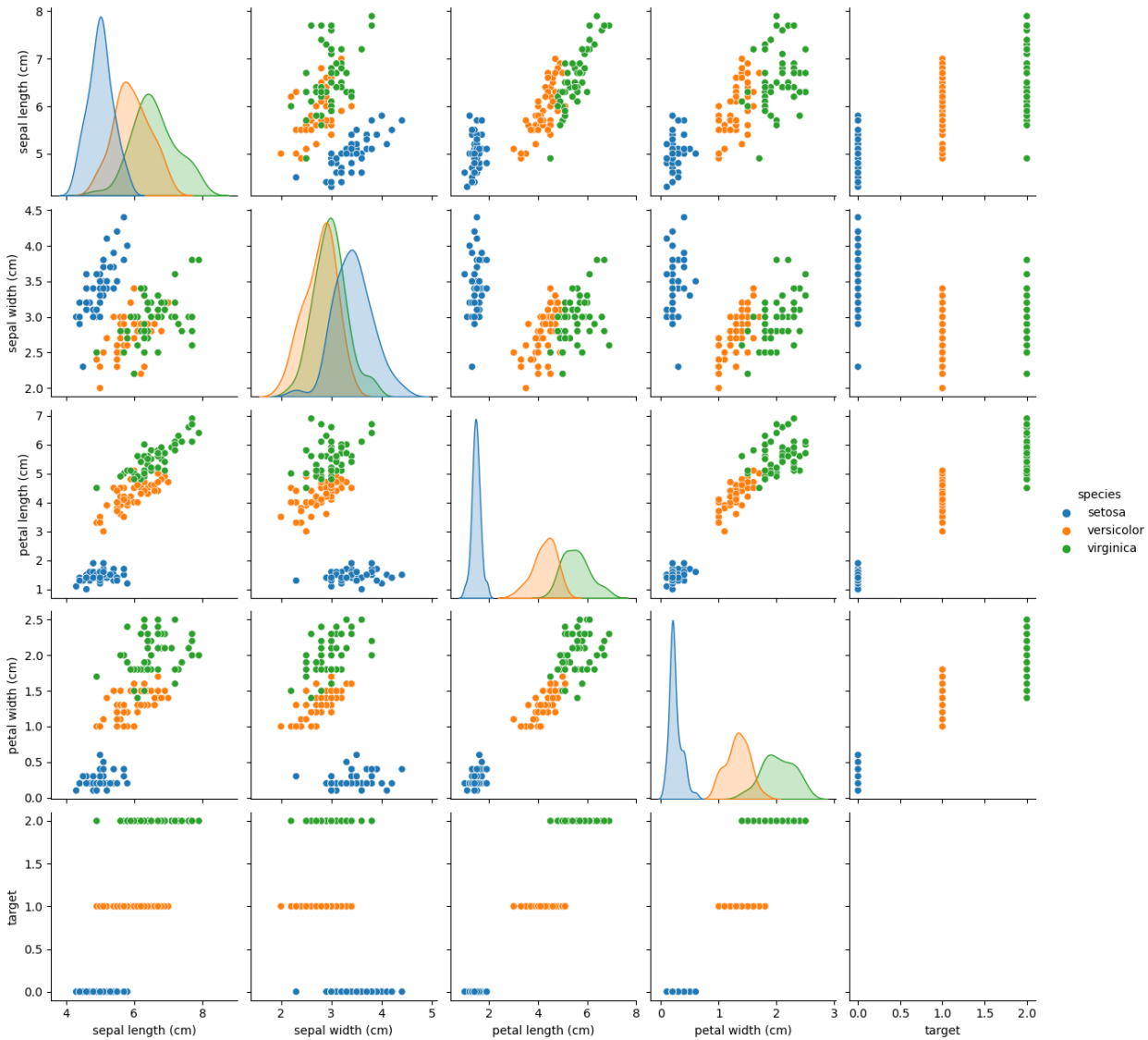
Visualization of Iris dataset



Some metrics

TP = true positive, TN = true negative, FP = false positive, FN = false negative

Accuracy. The number of samples correctly classified out of all the samples present in the (test) set.

$$Accuracy = \frac{TP + TN}{(TP + TN + FP + FN)}$$

Precision (for the positive class). The number of samples actually belonging to the positive class out of all the samples that were predicted to be of the positive class by the model.

$$Precision = \frac{TP}{(TP + FP)}$$

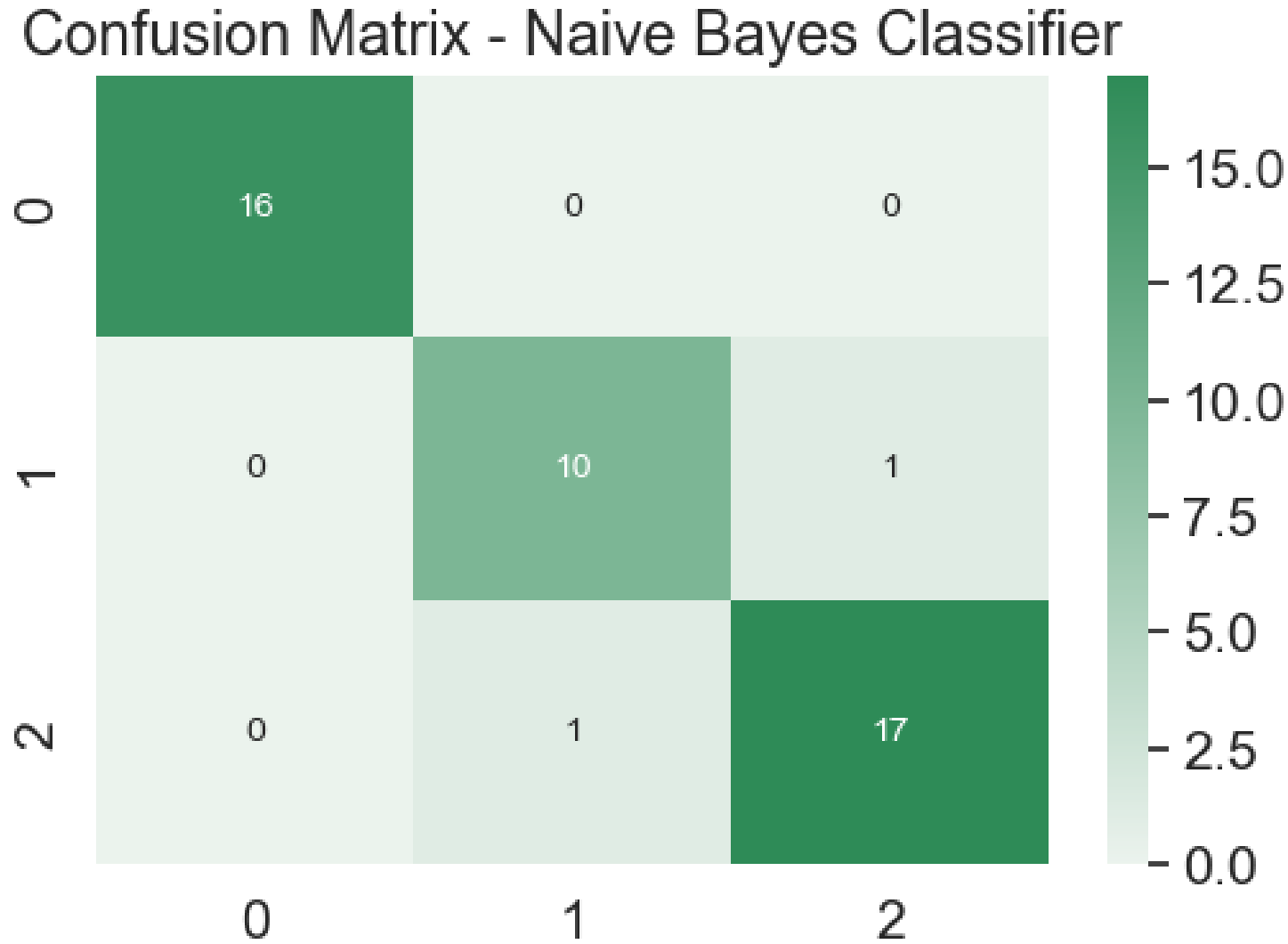
Recall (for the positive class). The number of samples predicted correctly to be belonging to the positive class out of all the samples that actually belong to the positive class.

$$Recall = \frac{TP}{(TP + FP)}$$

F1-Score (for the positive class). The harmonic mean of the precision and recall scores obtained for the positive class.

$$F1 - score = \frac{2 * Precision * Recall}{(Precision + Recall)}$$

Naïve Bayes: confusion matrix

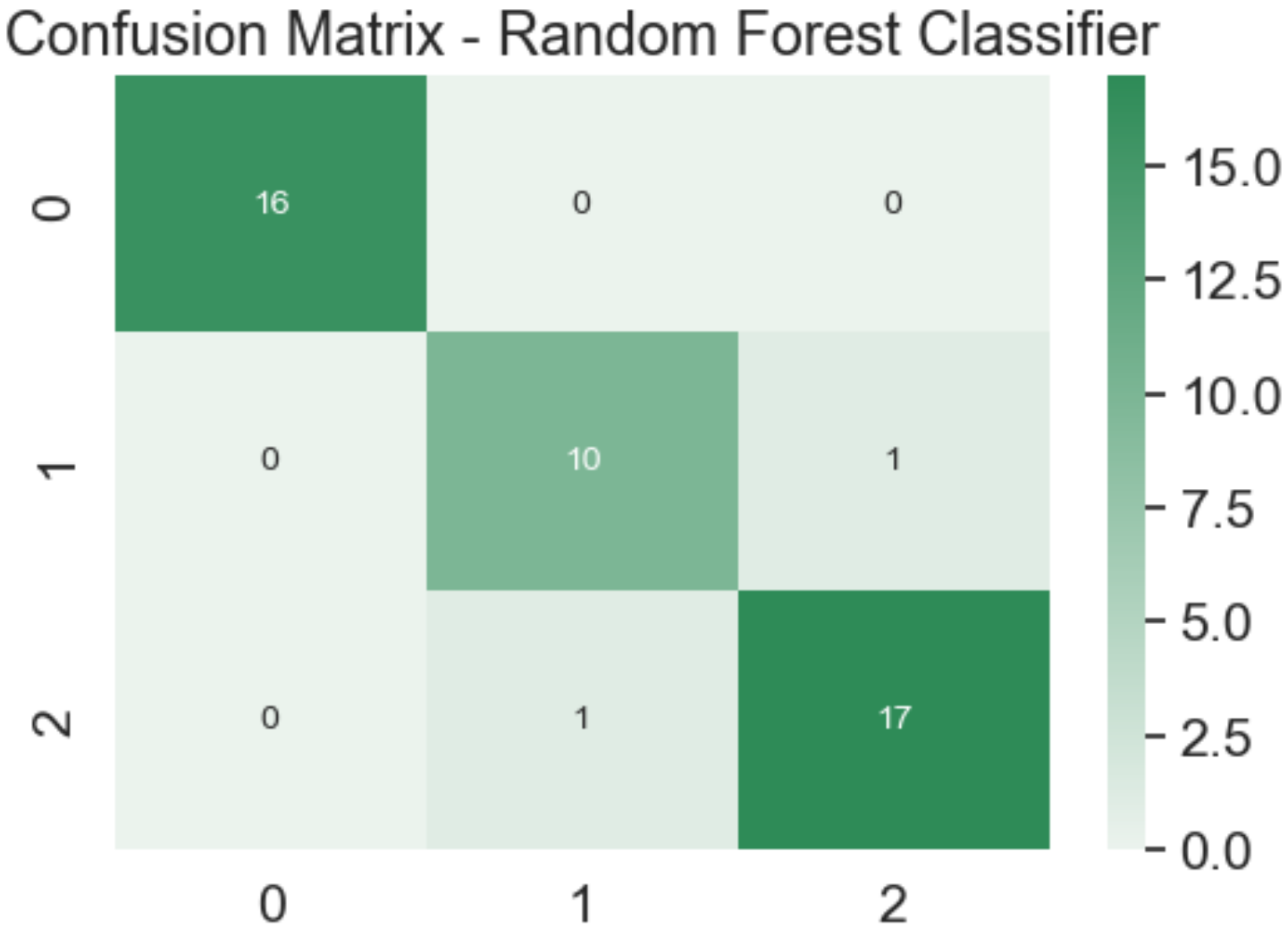


Naïve Bayes: code chunk and metrics

```
from sklearn.naive_bayes import GaussianNB

# create a Gaussian RF classifier
nb_model = GaussianNB()

# fit the model to the iris dataset
nb_model.fit(X_train,y_train)

sns.light_palette("seagreen", as_cmap=True)
```

	Testing
accuracy	95.56
precision	95.56
recall	95.56
f1-score	95.56

Random Forest: confusion matrix



Random Forest: code chunk and metrics

```
from sklearn.ensemble import RandomForestClassifier

# create a Gaussian RF classifier
rf_model = RandomForestClassifier(n_estimators=1000)

# fit the model to the iris dataset
rf_model.fit(X_train,y_train)
```

	Testing
accuracy	95.56
precision	95.56
recall	95.56
f1-score	95.56

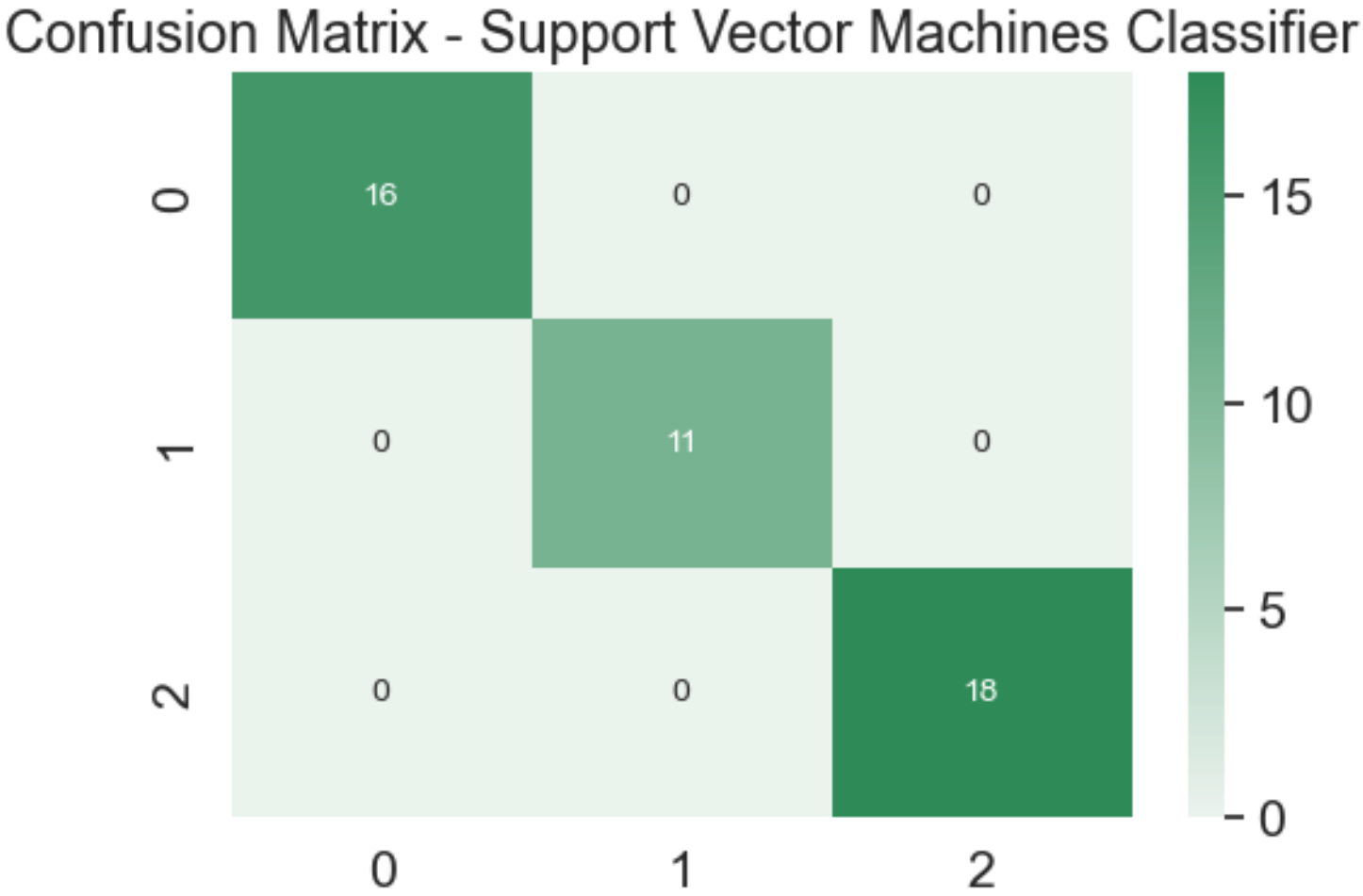
Logistic Regression: confusion matrix

Logistic Regression: code chunk and metrics

```
1 from sklearn.linear_model import  LogisticRegression
2
3 # create a Logistic regresion model
4 lr_model = LogisticRegression(solver= 'lbfgs', max_iter=400, multi_class = '
      multinomial')
5
6 # fit the model to the iris dataset
7 lr_model.fit(X_train, y_train)
```

	Testing
accuracy	97.78
precision	97.78
recall	97.78
f1-score	97.78

Support Vector Machines: confusion matrix

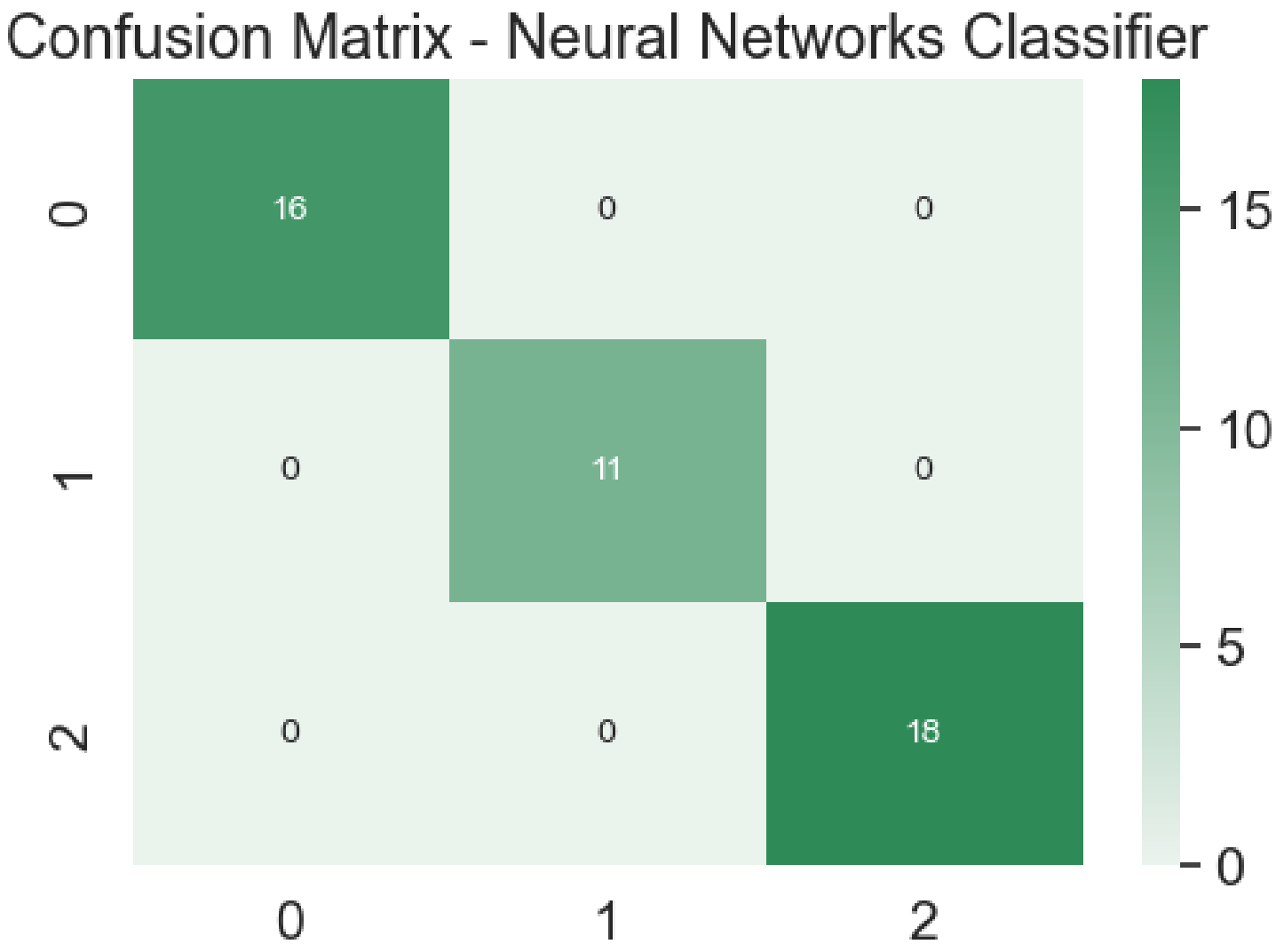


Support Vector Machines: code chunk and metrics

```
from sklearn.svm import SVC

# create a SVM model
svm_model = SVC(kernel = 'linear', random_state = 0)

# fit the model to the iris dataset
svm_model.fit(X_train, y_train)
```

	Testing
accuracy	100.0
precision	100.0
recall	100.0
f1-score	100.0

Neural Network: confusion matrix

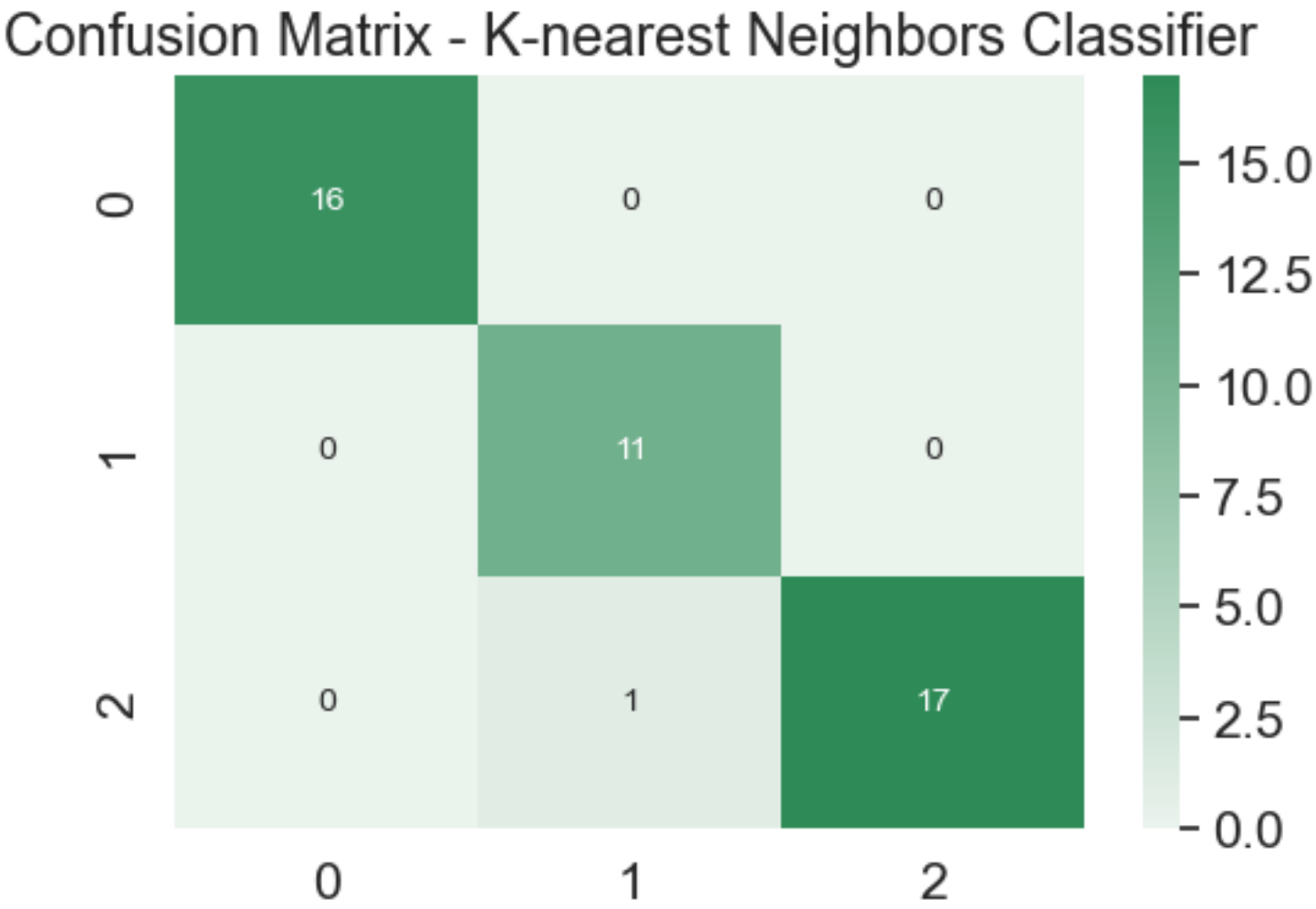


Neural Networks: code chunk and metrics

```
from sklearn.neural_network import MLPClassifier

mlp_model = MLPClassifier(hidden_layer_sizes=(10, 5), max_iter=1000)

# fit the model to the iris dataset
mlp_model.fit(X_train, y_train)

# make predictions on test set
y_pred_mlp = mlp_model.predict(X_test)
```

	Testing
accuracy	100.0
precision	100.0
recall	100.0
f1-score	100.0

K-Nearest Neighbors: confusion matrix



K-Nearest Neighbors: code chunk and metrics

```
1 from sklearn.neighbors import KNeighborsClassifier
2
3 # create a KNN model
4 knn_model = KNeighborsClassifier(n_neighbors = 5, weights = 'distance')
5
6 # fit the model to the iris dataset
7 knn_model.fit(X_train, y_train)
```

	Testing
accuracy	97.78
precision	97.78
recall	97.78
f1-score	97.78

XGBoost: confusion matrix

XGBoost: code chunk and metrics

```
import sys
!{sys.executable} -m pip install xgboost
from xgboost import XGBClassifier

xgb_model = XGBClassifier(n_estimators=100, learning_rate= 0.3)
# fit the model to the iris dataset
xgb_model.fit(X_train, y_train)

# make predictions on test set
y_pred_xgb = xgb_model.predict(X_test)
```

	Testing
accuracy	95.56
precision	95.56
recall	95.56
f1-score	95.56

References

Python:

https://www.python.org/

https://www.v7labs.com/blog/confusion-matrix-guide