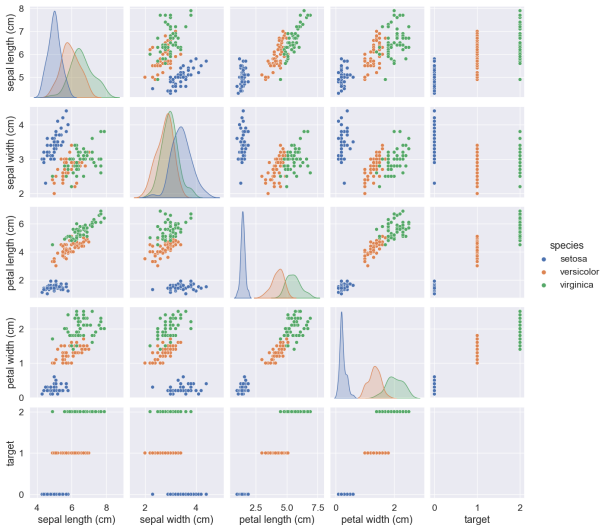


Visualization of Iris dataset



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.

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

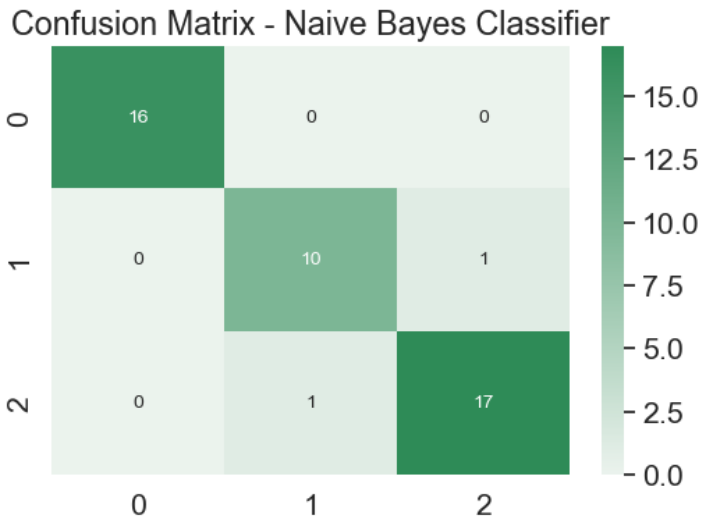
Recall (for the positive class). The number of samples predicted correctly to be belonging to the positive class out of all the samples that actually belong to the positive class.

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

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



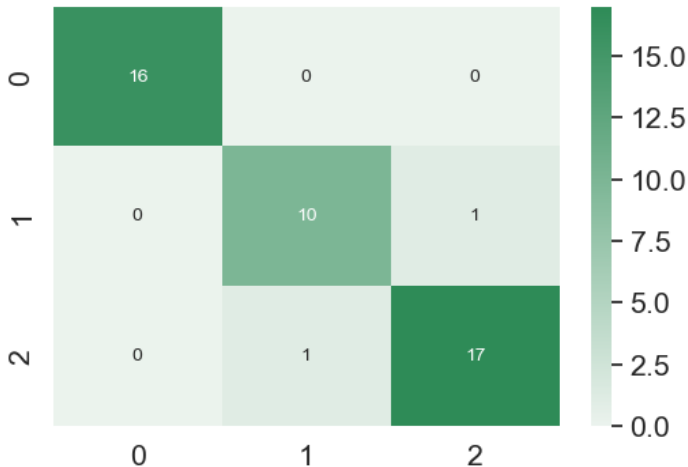
Naïve Bayes: code chunk and metrics

```
1 from sklearn.naive_bayes import GaussianNB
2
3 # create a Gaussian NB classifier
4 nb_model = GaussianNB()
5
6 # fit the model to the iris dataset
7 nb_model.fit(X_train,y_train)
8
9 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

Confusion Matrix - Random Forest Classifier



Random Forest: code chunk and metrics

```
1 from sklearn.ensemble import RandomForestClassifier
2
3 # create a Gaussian RF classifier
4 rf_model = RandomForestClassifier(n_estimators=1000)
5
6 # fit the model to the iris dataset
7 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

Confusion Matrix - Logistic Regression Classifier



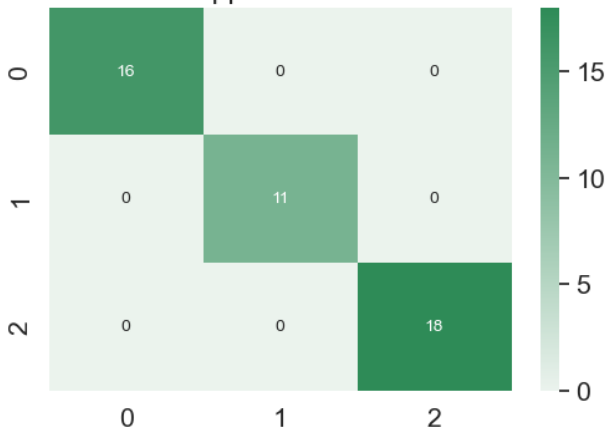
Logistic Regression: code chunk and metrics

```
1 from sklearn.linear_model import LogisticRegression
2
3 # create a Logistic regression 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

Confusion Matrix - Support Vector Machines Classifier



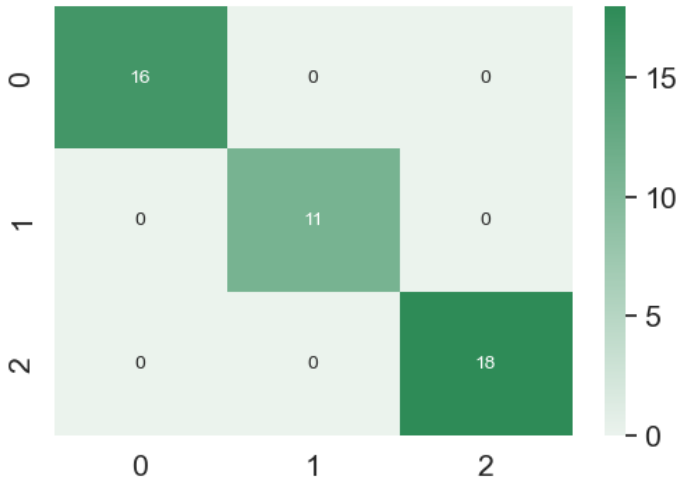
Support Vector Machines: code chunk and metrics

```
1 from sklearn.svm import SVC
2
3 # create a SVM model
4 svm_model = SVC(kernel = 'linear', random_state = 0)
5
6 # fit the model to the iris dataset
7 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

Confusion Matrix - Neural Networks Classifier



Neural Networks: code chunk and metrics

```
1 from sklearn.neural_network import MLPClassifier
2
3 mlp_model = MLPClassifier(hidden_layer_sizes=(10, 5), max_iter=1000)
4
5 # fit the model to the iris dataset
6 mlp_model.fit(X_train, y_train)
7
8 # make predictions on test set
9 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

Confusion Matrix - K-nearest Neighbors Classifier



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

```
1 import sys
2 !{sys.executable} -m pip install xgboost
3 from xgboost import XGBClassifier
4
5 xgb_model = XGBClassifier(n_estimators=100, learning_rate= 0.3)
6 # fit the model to the iris dataset
7 xgb_model.fit(X_train, y_train)
8
9 # make predictions on test set
10 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>