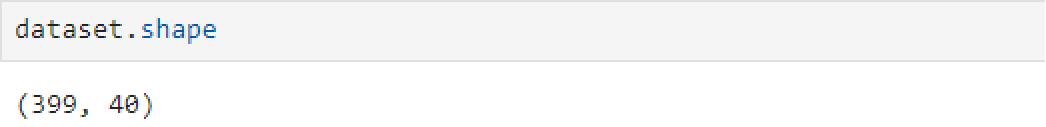
Machine Learning classification assignment

1. From the CDK dataset we need to predict the classification column, this is a classification problem statement

2. Total number of rows – 399

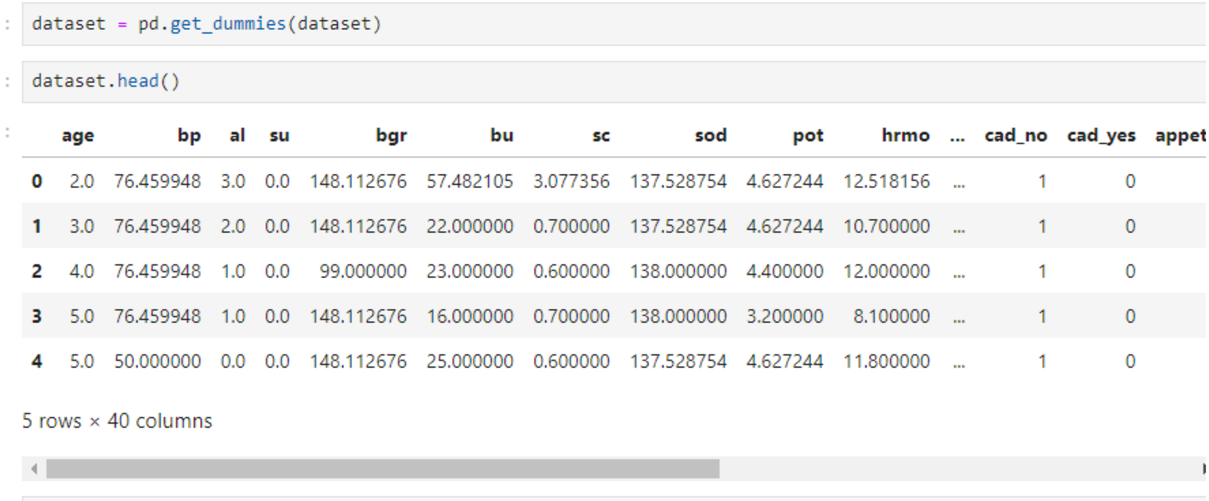
   Total number of columns – 40



```
dataset.shape
```

```
(399, 40)
```

3. We are converting the string to nominal data using get_dummies()



```
dataset = pd.get_dummies(dataset)
```

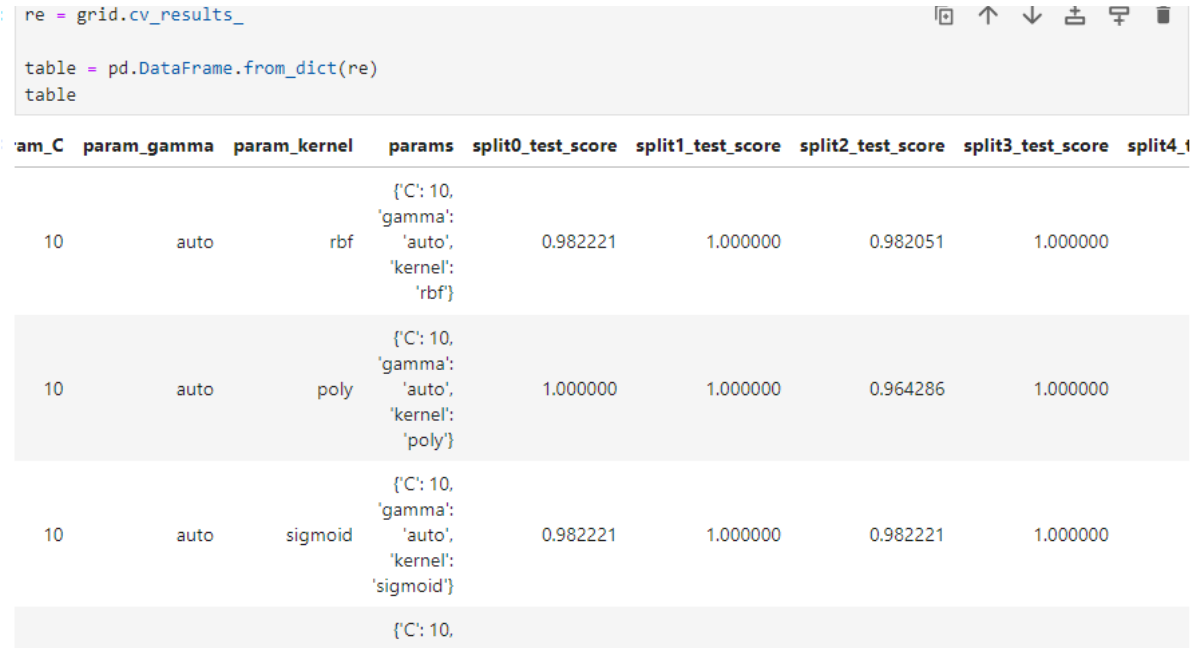
```
dataset.head()
```

	age	bp	al	su	bgr	bu	sc	sod	pot	hrmo	...	cad_no	cad_yes	appet
0	2.0	76.459948	3.0	0.0	148.112676	57.482105	3.077356	137.528754	4.627244	12.518156	...	1	0	
1	3.0	76.459948	2.0	0.0	148.112676	22.000000	0.700000	137.528754	4.627244	10.700000	...	1	0	
2	4.0	76.459948	1.0	0.0	99.000000	23.000000	0.600000	138.000000	4.400000	12.000000	...	1	0	
3	5.0	76.459948	1.0	0.0	148.112676	16.000000	0.700000	138.000000	3.200000	8.100000	...	1	0	
4	5.0	50.000000	0.0	0.0	148.112676	25.000000	0.600000	137.528754	4.627244	11.800000	...	1	0	

5 rows × 40 columns

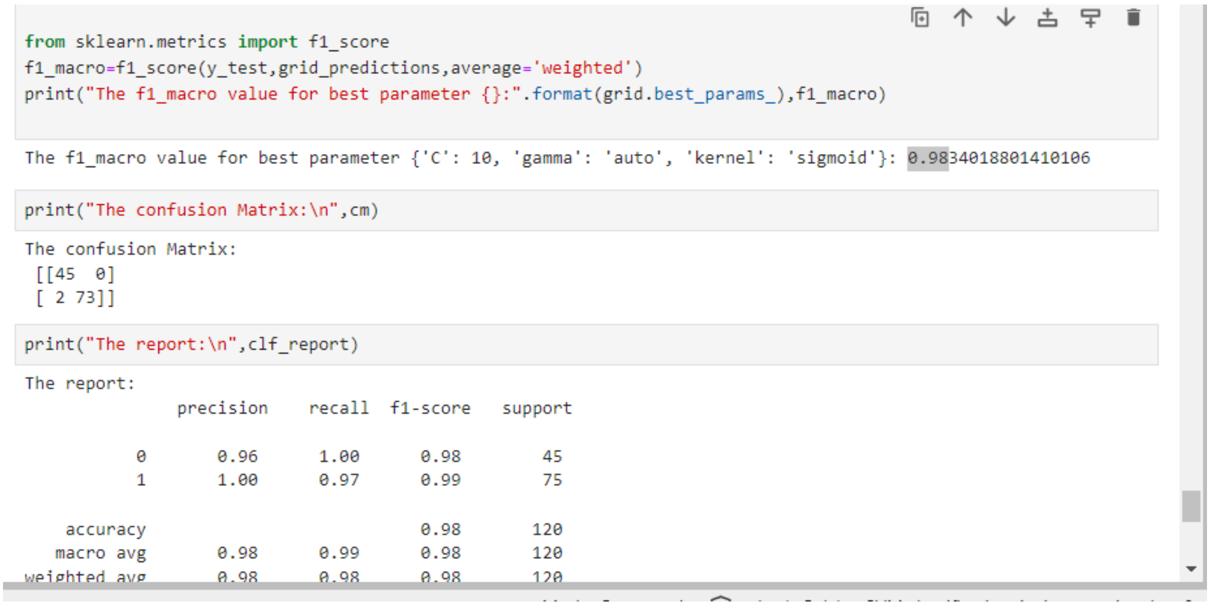
4. Models used

SVM



```
re = grid.cv_results_

table = pd.DataFrame.from_dict(re)
table
```

am_C	param_gamma	param_kernel	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_t
10	auto	rbf	{'C': 10, 'gamma': 'auto', 'kernel': 'rbf'}	0.982221	1.000000	0.982051	1.000000	
10	auto	poly	{'C': 10, 'gamma': 'auto', 'kernel': 'poly'}	1.000000	1.000000	0.964286	1.000000	
10	auto	sigmoid	{'C': 10, 'gamma': 'auto', 'kernel': 'sigmoid'}	0.982221	1.000000	0.982221	1.000000	
			{'C': 10,					



```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'C': 10, 'gamma': 'auto', 'kernel': 'sigmoid'}: 0.9834018801410106
```

```
print("The confusion Matrix:\n",cm)
```

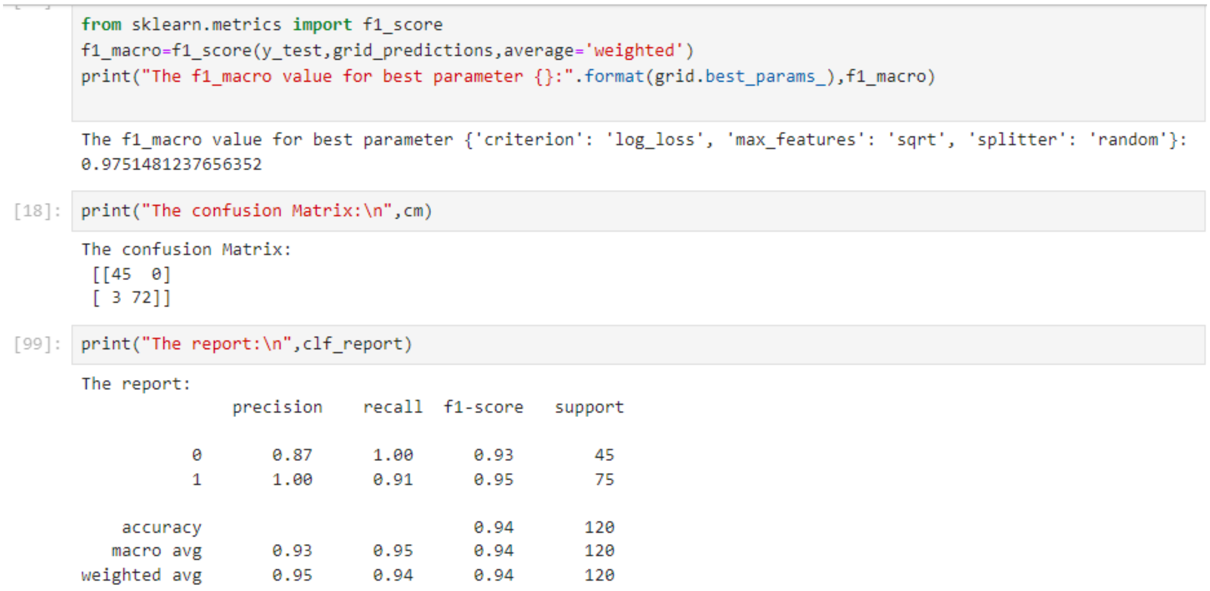
```
The confusion Matrix:
 [[45  0]
 [ 2 73]]
```

```
print("The report:\n",clf_report)
```

```
The report:
               precision    recall  f1-score   support

           0       0.96      1.00      0.98        45
           1       1.00      0.97      0.99        75

    accuracy                           0.98       120
   macro avg       0.98      0.99      0.98       120
weighted avg       0.98      0.98      0.98       120
```

The best score for SVM is 0.98

Decision tree



```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'criterion': 'log_loss', 'max_features': 'sqrt', 'splitter': 'random'}:
0.9751481237656352
```

```
[18]: print("The confusion Matrix:\n",cm)
```

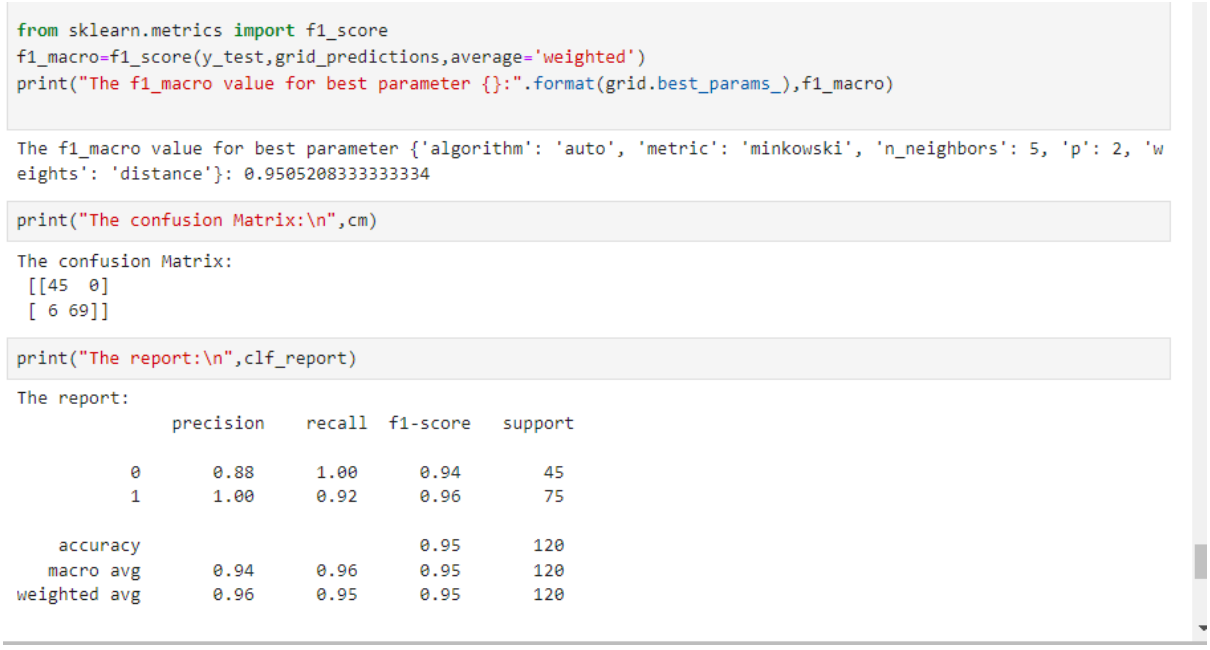
```
The confusion Matrix:
 [[45  0]
 [ 3 72]]
```

```
[99]: print("The report:\n",clf_report)
```

```
The report:
               precision    recall  f1-score   support

           0       0.87      1.00      0.93        45
           1       1.00      0.91      0.95        75

    accuracy                           0.94       120
   macro avg       0.93      0.95      0.94       120
weighted avg       0.95      0.94      0.94       120
```

F1 score is 0.975

KNN classification



```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'algorithm': 'auto', 'metric': 'minkowski', 'n_neighbors': 5, 'p': 2, 'w
eights': 'distance'}: 0.9505208333333334
```

```
print("The confusion Matrix:\n",cm)
```

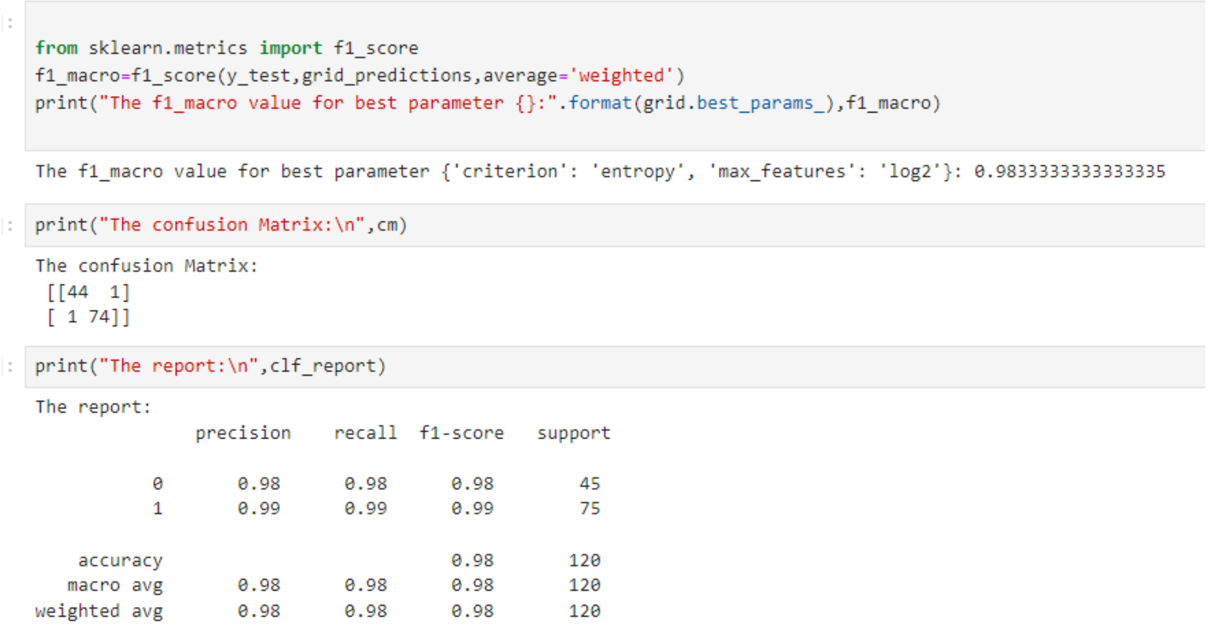
```
The confusion Matrix:
 [[45  0]
 [ 6 69]]
```

```
print("The report:\n",clf_report)
```

```
The report:
               precision    recall  f1-score   support

           0       0.88      1.00      0.94        45
           1       1.00      0.92      0.96        75

    accuracy                           0.95       120
   macro avg       0.94      0.96      0.95       120
weighted avg       0.96      0.95      0.95       120
```

The F1 score is 0.950

Random Forest classifier



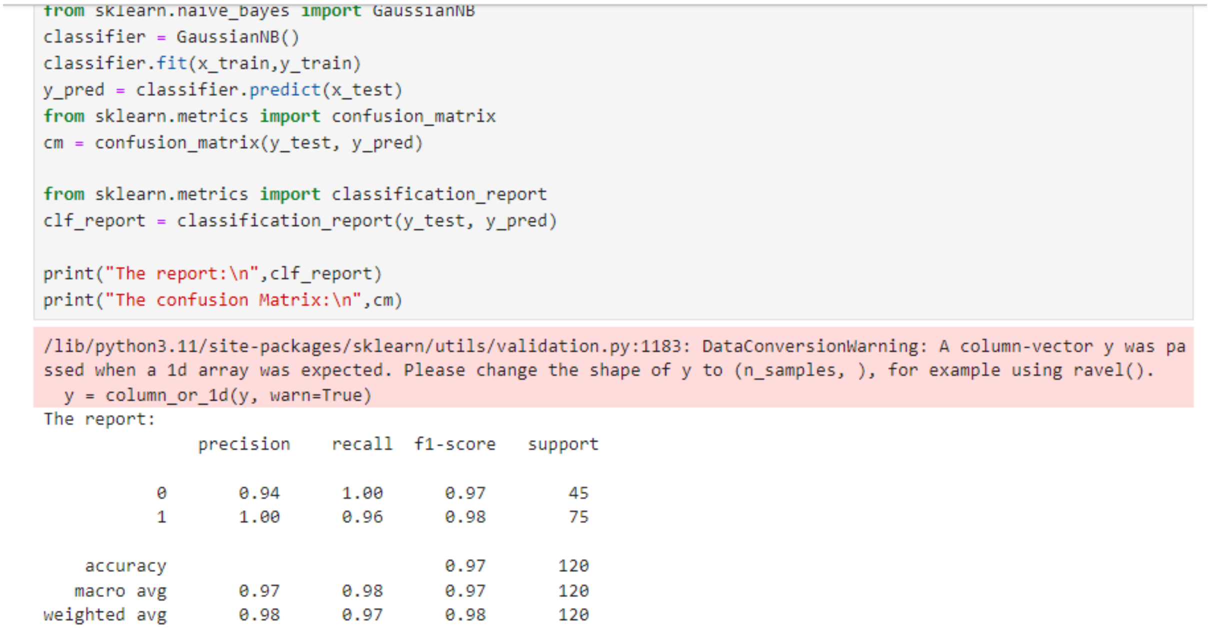
```
:
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)

The f1_macro value for best parameter {'criterion': 'entropy', 'max_features': 'log2'}: 0.9833333333333335
```

```
:  print("The confusion Matrix:\n",cm)

The confusion Matrix:
 [[44  1]
 [ 1 74]]
```

```
:  print("The report:\n",clf_report)

The report:
               precision    recall  f1-score   support

           0       0.98      0.98      0.98        45
           1       0.99      0.99      0.99        75

    accuracy                           0.98       120
   macro avg       0.98      0.98      0.98       120
weighted avg       0.98      0.98      0.98       120
```

The best F1 score is 0.983

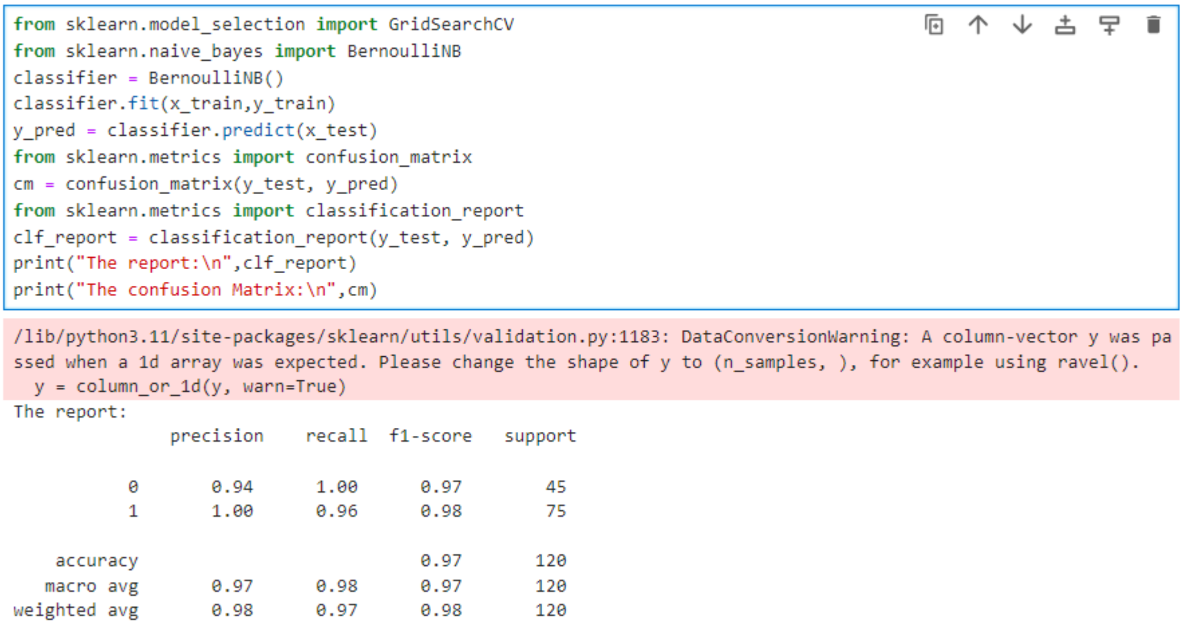
Naïve Bayes – gaussianNB



```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train,y_train)
y_pred = classifier.predict(x_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

from sklearn.metrics import classification_report
clf_report = classification_report(y_test, y_pred)

print("The report:\n",clf_report)
print("The confusion Matrix:\n",cm)
```

```
/lib/python3.11/site-packages/sklearn/utils/validation.py:1183: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
The report:
               precision    recall  f1-score   support

           0       0.94      1.00      0.97        45
           1       1.00      0.96      0.98        75

    accuracy                           0.97       120
   macro avg       0.97      0.98      0.97       120
weighted avg       0.98      0.97      0.98       120
```

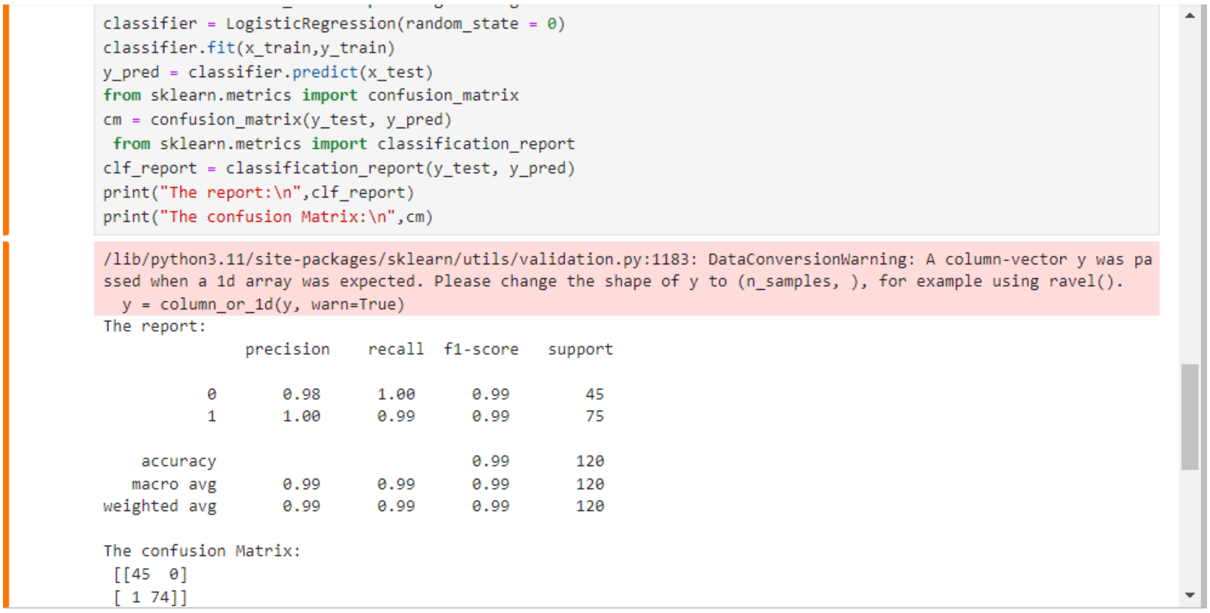
Bernoulli NB



```
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import BernoulliNB
classifier = BernoulliNB()
classifier.fit(x_train,y_train)
y_pred = classifier.predict(x_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import classification_report
clf_report = classification_report(y_test, y_pred)
print("The report:\n",clf_report)
print("The confusion Matrix:\n",cm)
```

```
/lib/python3.11/site-packages/sklearn/utils/validation.py:1183: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
The report:
               precision    recall  f1-score   support

           0       0.94      1.00      0.97        45
           1       1.00      0.96      0.98        75

    accuracy                           0.97       120
   macro avg       0.97      0.98      0.97       120
weighted avg       0.98      0.97      0.98       120
```

Logistic Regression



```
classifier = LogisticRegression(random_state = 0)
classifier.fit(x_train,y_train)
y_pred = classifier.predict(x_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
 from sklearn.metrics import classification_report
clf_report = classification_report(y_test, y_pred)
print("The report:\n",clf_report)
print("The confusion Matrix:\n",cm)
```

```
/lib/python3.11/site-packages/sklearn/utils/validation.py:1183: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
The report:
               precision    recall  f1-score   support

           0       0.98      1.00      0.99        45
           1       1.00      0.99      0.99        75

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120

The confusion Matrix:
 [[45  0]
 [ 1 74]]
```

The F1 score is 0.99

Best Model

Out of these algorithms, based on the F1 score logistic regression is having 0.99 score. So logistic regression is considered as the best model