CPSC-6430 Machine Learning: Implementation & Evaluation

Project 5: Binary Classification to Predict the Presence or Absence of Breast Cancer

Dineshchandar Ravichandran
C19657741
Email: dravich@g.clemson.edu

Contents

Introduction

To implement k Nearest Neighbor, Logistic Regression, Support Vector Machine, and Multilayer Perceptron algorithm using Scikit-learn library. To analyze the Wisconsin Breast Cancer Dataset data and pick the best algorithm based on its performance.

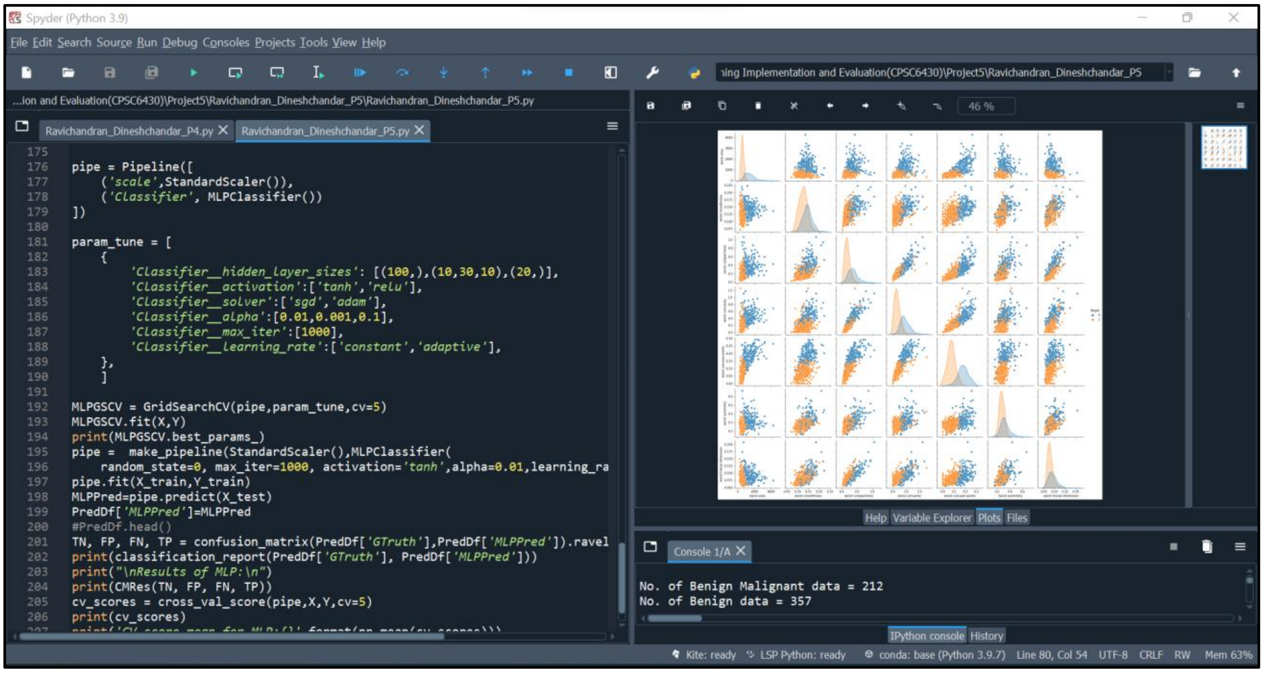
1. Problem Statement

To implement a Python program to import the breast cancer data from the Scikit-learn library. And implement a supervised machine learning model with the following algorithms:

1. k Nearest Neighbor
2. Logistic Regression
3. Support Vector Machine
4. Multilayer Perceptron

And determine the best performing algorithm using the confusion matrix and other metrics.

2. Project Screenshot:

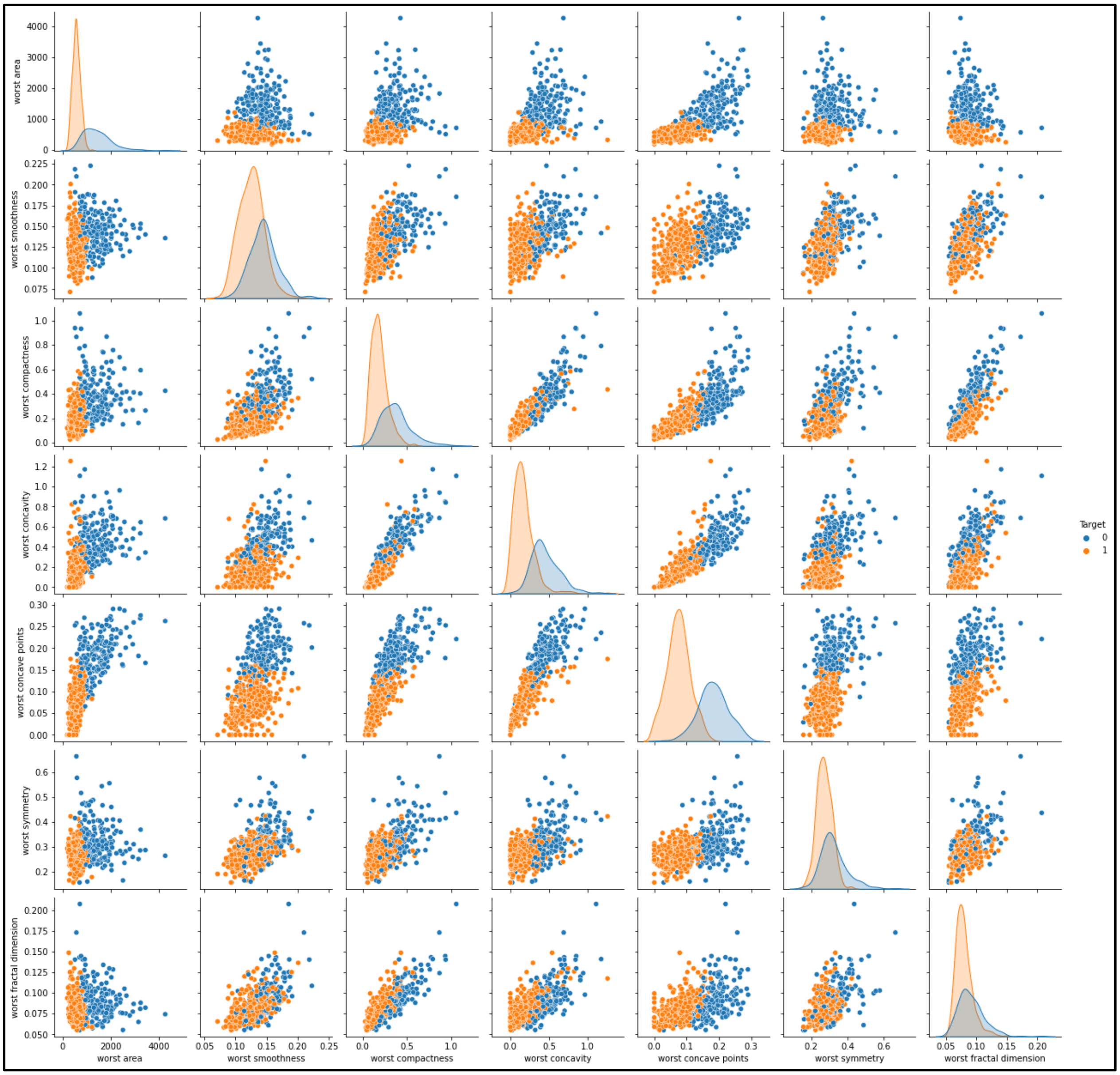

Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
...ning Implementation and Evaluation(CPSC6430)\Project5\Ravichandran_Dineshchandar_P5
...ion and Evaluation(CPSC6430)\Project5\Ravichandran_Dineshchandar_P5\Ravichandran_Dineshchandar_P5.py
Ravichandran_Dineshchandar_P4.py
Ravichandran_Dineshchandar_P5.py
46 %
Help
Variable Explorer
Plots
Files
Console 1/A
IPython console
History
Kite: ready
LSP Python: ready
conda: base (Python 3.9.7)
Line 80, Col 54
UTF-8
CRLF
RW
Mem 63%

```python

pipe = Pipeline([
    ('scale',StandardScaler()),
    ('Classifier', MLPClassifier())
])

param_tune = [
    {
        'Classifier__hidden_layer_sizes': [(100,),(10,30,10),(20,)],
        'Classifier__activation':['tanh','relu'],
        'Classifier__solver':['sgd','adam'],
        'Classifier__alpha':[0.01,0.001,0.1],
        'Classifier__max_iter':[1000],
        'Classifier__learning_rate':['constant','adaptive'],
    },
    ]

MLPGSCV = GridSearchCV(pipe,param_tune,cv=5)
MLPGSCV.fit(X,Y)
print(MLPGSCV.best_params_)
pipe =  make_pipeline(StandardScaler(),MLPClassifier(
    random_state=0, max_iter=1000, activation='tanh',alpha=0.01,learning_ra
pipe.fit(X_train,Y_train)
MLPPred=pipe.predict(X_test)
PredDf['MLPPred']=MLPPred
#PredDf.head()
TN, FP, FN, TP = confusion_matrix(PredDf['GTruth'],PredDf['MLPPred']).ravel
print(classification_report(PredDf['GTruth'], PredDf['MLPPred']))
print("\nResults of MLP:\n")
print(CMRes(TN, FP, FN, TP))
cv_scores = cross_val_score(pipe,X,Y,cv=5)
print(cv_scores)
```

```
No. of Benign Malignant data = 212
No. of Benign data = 357
```

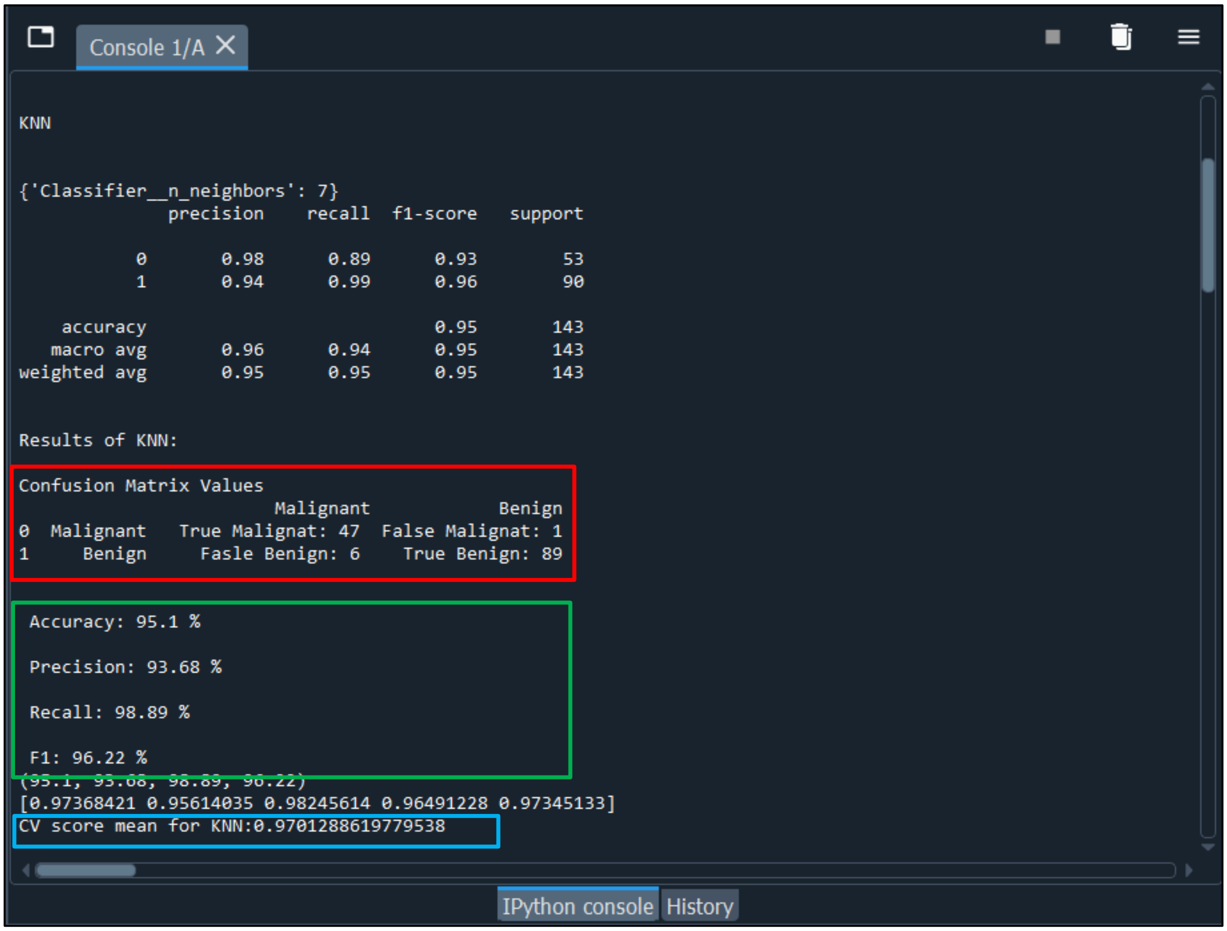
- The above screenshots represent the code in the SPYDER IDE, along with the no. of Benign and Malignant entries in sklearn's breast cancer data.
    - Malignant data =212
    - Benign data =357

- The following is the data visualization of the worst area, worst smoothness, worst compactness, worst concavity, worst concave points, worst symmetry, and worst fractal dimension. Which is being highlighted in Orange-Malignant and Blue-Benign data.



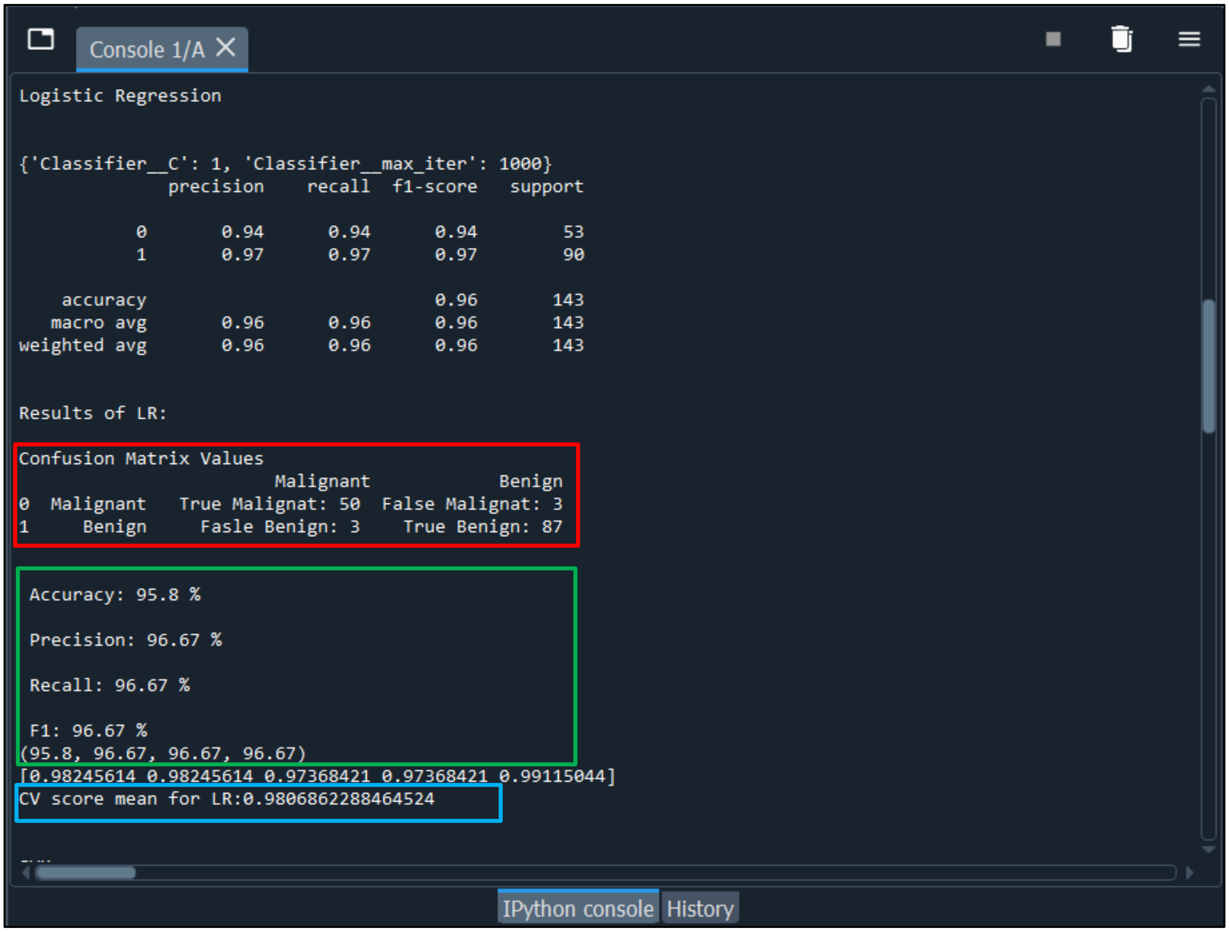
- Console Screen Shot for all the four algorithms' results along with best parameters output from GridsearchCV, confusion matrix (highlighted in red in below image), performance metrics(highlighted in green in below image), and CV mean values (highlighted in blue in below image) are illustrated below:

KNN:



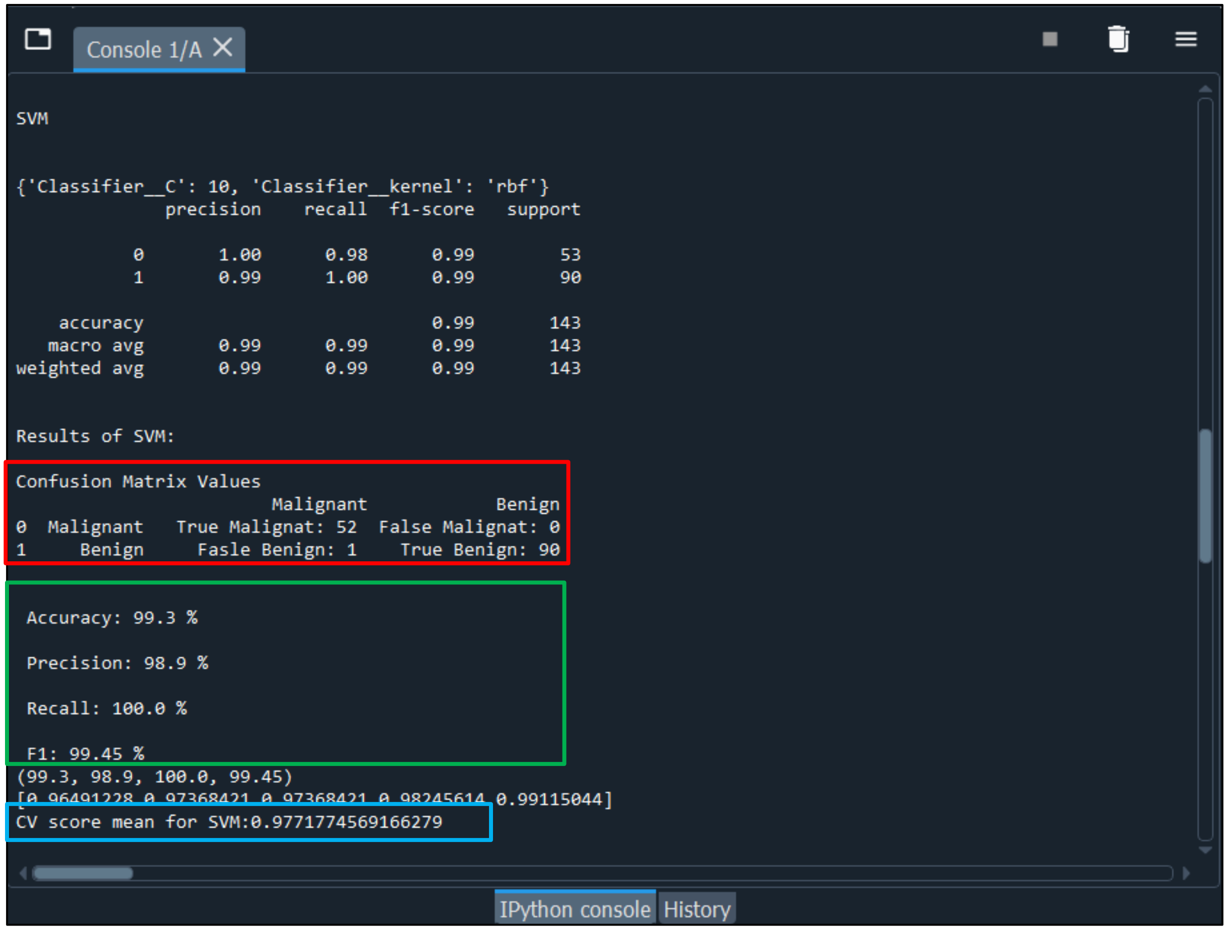
```
KNN


{'Classifier__n_neighbors': 7}
              precision    recall  f1-score   support

           0       0.98      0.89      0.93        53
           1       0.94      0.99      0.96        90

    accuracy                           0.95       143
   macro avg       0.96      0.94      0.95       143
weighted avg       0.95      0.95      0.95       143


Results of KNN:

Confusion Matrix Values
                        Malignant            Benign
0  Malignant   True Malignat: 47  False Malignat: 1
1     Benign     Fasle Benign: 6     True Benign: 89

 Accuracy: 95.1 %

 Precision: 93.68 %

 Recall: 98.89 %

 F1: 96.22 %
(95.1, 93.68, 98.89, 96.22)
[0.97368421 0.95614035 0.98245614 0.96491228 0.97345133]
CV score mean for KNN:0.9701288619779538
```

Logistic Regression:



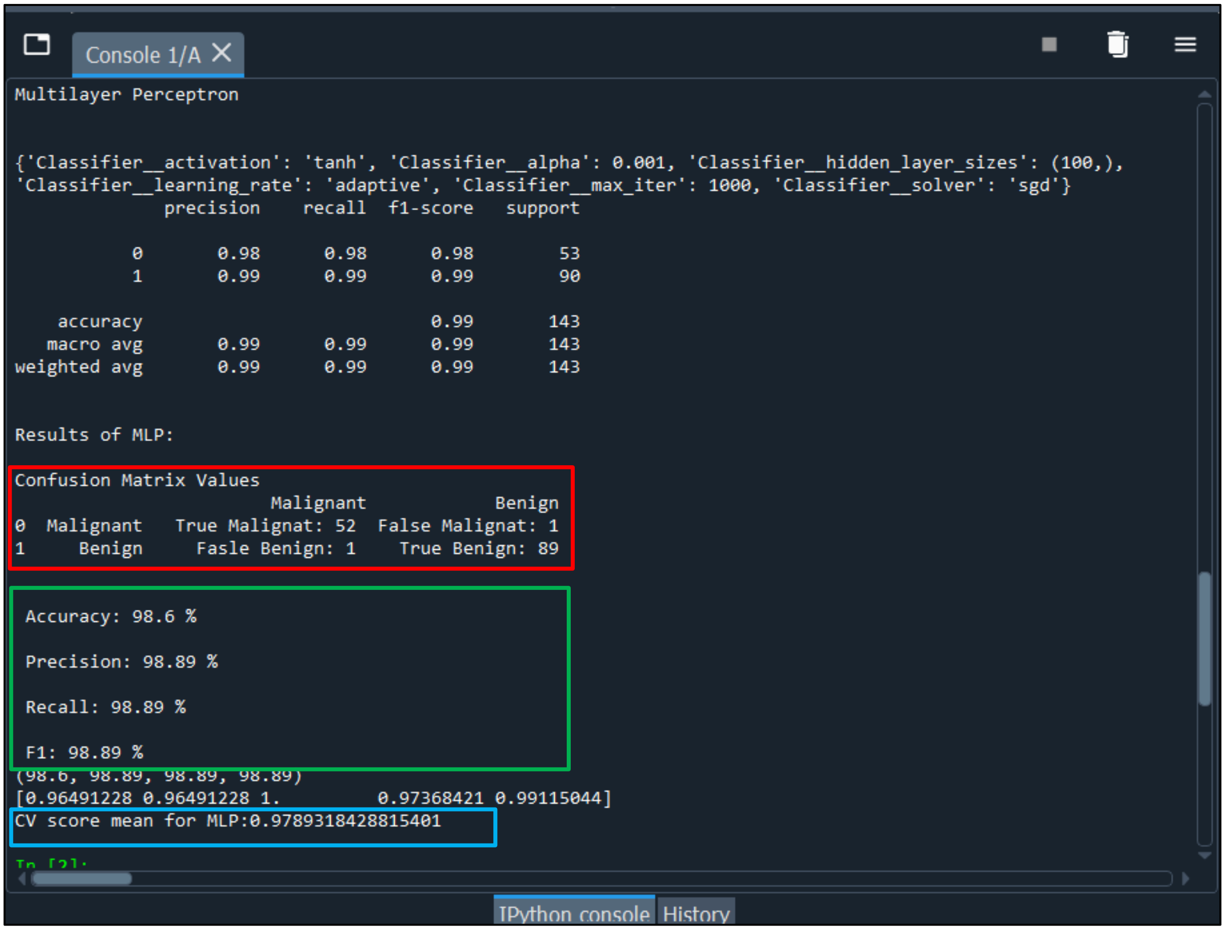
```
Logistic Regression


{'Classifier__C': 1, 'Classifier__max_iter': 1000}
              precision    recall  f1-score   support

           0       0.94      0.94      0.94        53
           1       0.97      0.97      0.97        90

    accuracy                           0.96       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.96      0.96      0.96       143


Results of LR:

Confusion Matrix Values
                        Malignant              Benign
0  Malignant   True Malignat: 50  False Malignat: 3
1     Benign     Fasle Benign: 3    True Benign: 87

 Accuracy: 95.8 %

 Precision: 96.67 %

 Recall: 96.67 %

 F1: 96.67 %
(95.8, 96.67, 96.67, 96.67)
[0.98245614 0.98245614 0.97368421 0.97368421 0.99115044]
CV score mean for LR:0.9806862288464524
```

SVM:



```
SVM


{'Classifier__C': 10, 'Classifier__kernel': 'rbf'}
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        53
           1       0.99      1.00      0.99        90

    accuracy                           0.99       143
   macro avg       0.99      0.99      0.99       143
weighted avg       0.99      0.99      0.99       143


Results of SVM:

Confusion Matrix Values
                        Malignant             Benign
0  Malignant   True Malignat: 52  False Malignat: 0
1     Benign     Fasle Benign: 1     True Benign: 90

 Accuracy: 99.3 %

 Precision: 98.9 %

 Recall: 100.0 %

 F1: 99.45 %
(99.3, 98.9, 100.0, 99.45)
[0.96491228 0.97368421 0.97368421 0.98245614 0.99115044]
CV score mean for SVM:0.9771774569166279
```

Multilayer Perceptron:



```
Multilayer Perceptron

{'Classifier__activation': 'tanh', 'Classifier__alpha': 0.001, 'Classifier__hidden_layer_sizes': (100,),
'Classifier__learning_rate': 'adaptive', 'Classifier__max_iter': 1000, 'Classifier__solver': 'sgd'}
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        53
           1       0.99      0.99      0.99        90

    accuracy                           0.99       143
   macro avg       0.99      0.99      0.99       143
weighted avg       0.99      0.99      0.99       143

Results of MLP:

Confusion Matrix Values
                        Malignant            Benign
0  Malignant   True Malignat: 52  False Malignat: 1
1     Benign      Fasle Benign: 1     True Benign: 89

 Accuracy: 98.6 %

 Precision: 98.89 %

 Recall: 98.89 %

 F1: 98.89 %
(98.6, 98.89, 98.89, 98.89)
[0.96491228 0.96491228 1.         0.97368421 0.99115044]
CV score mean for MLP:0.9789318428815401

In [2]:
```

3. Project Input and Output

3.1. Input:

- SK-Learn Wisconsin Breast Cancer Dataset:

```
from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
```

3.2 Output:

1. **KNN:**
   Accuracy: 95.1 %
   Precision: 93.68 %
   Recall: 98.89 %
   F1: 96.22 %
   Confusion Matrix:

	Predicted: No	**Predicted: Yes**
Actual: No	True Malignant: 47	False Malignant: 1
Actual: Yes	False Benign: 6	True Benign: 89

   Cross-validation mean score: 0.9701
   - The above results were achieved using the following tuning parameters:
     - **Neighbors: 7**
     - CV:5

2. **Logistic Regression**:
   Accuracy: 95.8 %
   Precision: 96.67 %
   Recall: 96.67 %
   F1: 96.67 %
   Confusion Matrix:

	Predicted: No	**Predicted: Yes**
Actual: No	True Malignant: 50	False Malignant: 3
Actual: Yes	False Benign: 3	True Benign: 87

   Cross-validation mean score: 0.9771
   - The above results were achieved using the following tuning parameters:
     - **Max Iterations: 1000**
     - C(regularization value): 1
     - CV:5

3. **SVM:**
   Accuracy: 99.3 %
   Precision: 98.9 %
   Recall: 100.0 %
   F1: 99.45 %
   Confusion Matrix:

	Predicted: No	**Predicted: Yes**
Actual: No	True Malignant: 52	False Malignant: 0
Actual: Yes	False Benign: 1	True Benign: 90

Cross-validation mean score: 0.9771

- The above results were achieved using the following tuning parameters:
  - **Max Iterations: 1000 (default value)**
  - Classifier__kernel: “rbf”
  - C (regularization value): 1
  - CV:5

4. **Multilayer Perceptron:**
   Accuracy: 98.6 %
   Precision: 98.89 %
   Recall: 98.89 %
   F1: 98.89 %
   Confusion Matrix:

	Predicted: No	**Predicted: Yes**
Actual: No	True Malignant: 52	False Malignant: 1
Actual: Yes	False Benign: 1	True Benign: 89

Cross-validation mean score: 0.9789

- The above results were achieved using the following tuning parameters:
  - hidden_layer_sizes: 100
  - activation: 'tanh'
  - solver: sgd
  - alpha: 0.001
  - **max_iter:1000**
  - learning_rate: adaptive

4. Conclusion

- As we can observe in the above output, we can see that the **SVM algorithm is able to perform** the best out of the k Nearest Neighbor, Logistic Regression, Support Vector Machine, and Multilayer Perceptron algorithms. And algorithms like KNN and Logistic regression can safely be excluded from the study, because of poor accuracy and CV score.
- SVM has an **Accuracy** of **99.3%**, **Recall** of **100.00%, F1** of **99.45%,** and **CV-mean** score of **0.9771 and MLP has an Accuracy** of **98.6%, Recall** of **98.89 %, F1** of **98.89 %,** and **CV-mean** score of **0.9789.**
- Even though MLP has a marginally better CV-mean score compared to SVM, but the SVM has better accuracy, F1, and Recall rate for the given data set.
- Hence, I believe the **SVM algorithm** with the above tuning parameter will be the best algorithm for the above data set "Wisconsin Breast Cancer Dataset" in predicting the **"Benign and Malignant"** data. As inherently SVM algorithm tries to find the best decision boundary. However, if we were to implement the same on a bigger data set, given the better CV-mean score MLP would perform better, but to evaluate the same we would require a bigger data set.
- But since **SVM** for the given data set is able to generate **100% recall**, for this particular data SVM is the better fit. Since it is crucial in medical(cancer) application where we want to avoid mispredicting people with cancer as **"Benign."** Which generates zero false malignant.