Chronic Kieny Disease Predict

Problem Statement:

The goal is to develop a machine learning model to predict Chronic Kidney Disease (CKD) based on patient medical data. The system will analyze features like age, blood pressure, glucose levels, and other clinical parameters. Early detection of CKD can help in timely intervention and better management of the disease. The solution should ensure accuracy, scalability, and ease of use for healthcare providers. Data preprocessing, feature selection, and model evaluation are critical to achieving this objective.

Basic Dataset Info:

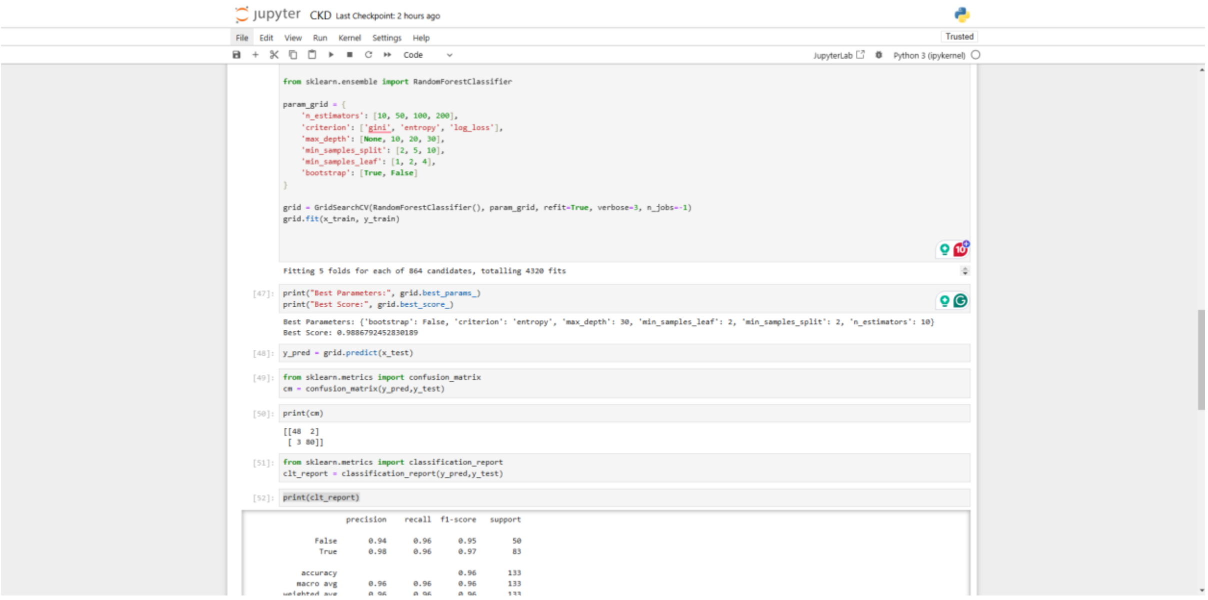
Total number of Row : 399

Total number of Column: 25

Pre-Processing:

The code uses **StandardScaler** from sklearn.preprocessing to standardize the feature data in x_train and x_test. Standardization involves scaling the data such that each feature has a mean of 0 and a standard deviation of 1. The fit_transform() method computes the necessary scaling parameters (mean and standard deviation) from the x_train data and applies the transformation. The transform() method then applies this scaling to x_test using the parameters learned from x_train, ensuring consistent scaling across both datasets. This preprocessing step helps machine learning models perform better by normalizing feature magnitudes and improving convergence.

All The Reach Value:



jupyter CKD Last Checkpoint: 2 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```python
from sklearn.ensemble import RandomForestClassifier

param_grid = {
    'n_estimators': [10, 50, 100, 200],
    'criterion': ['gini', 'entropy', 'log_loss'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

grid = GridSearchCV(RandomForestClassifier(), param_grid, refit=True, verbose=3, n_jobs=-1)
grid.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 864 candidates, totalling 4320 fits
```

```python
[47]: print("Best Parameters:", grid.best_params_)
      print("Best Score:", grid.best_score_)
```

```
Best Parameters: {'bootstrap': False, 'criterion': 'entropy', 'max_depth': 30, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 10}
Best Score: 0.9886792452830189
```

```python
[48]: y_pred = grid.predict(x_test)

[49]: from sklearn.metrics import confusion_matrix
      cm = confusion_matrix(y_pred,y_test)

[50]: print(cm)
```

```
[[48  2]
 [ 3 80]]
```

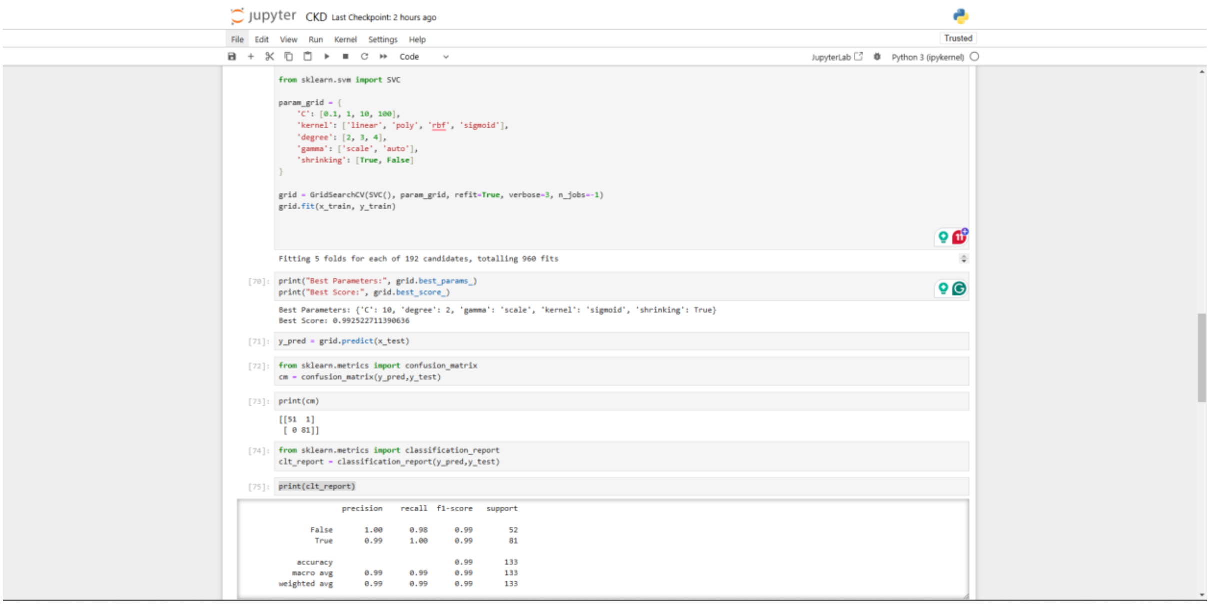
```python
[51]: from sklearn.metrics import classification_report
      clt_report = classification_report(y_pred,y_test)

[52]: print(clt_report)
```

```
              precision    recall  f1-score   support

       False       0.94      0.96      0.95        50
        True       0.98      0.96      0.97        83

    accuracy                           0.96       133
   macro avg       0.96      0.96      0.96       133
weighted avg       0.96      0.96      0.96       133
```



jupyter CKD Last Checkpoint: 2 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```python
from sklearn.svm import SVC

param_grid = {
    'C': [0.1, 1, 10, 100],
    'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
    'degree': [2, 3, 4],
    'gamma': ['scale', 'auto'],
    'shrinking': [True, False]
}

grid = GridSearchCV(SVC(), param_grid, refit=True, verbose=3, n_jobs=-1)
grid.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 192 candidates, totalling 960 fits
```

```python
[70]: print("Best Parameters:", grid.best_params_)
      print("Best Score:", grid.best_score_)
```

```
Best Parameters: {'C': 10, 'degree': 2, 'gamma': 'scale', 'kernel': 'sigmoid', 'shrinking': True}
Best Score: 0.992522711390636
```

```python
[71]: y_pred = grid.predict(x_test)

[72]: from sklearn.metrics import confusion_matrix
      cm = confusion_matrix(y_pred,y_test)

[73]: print(cm)
```

```
[[51  1]
 [ 0 81]]
```

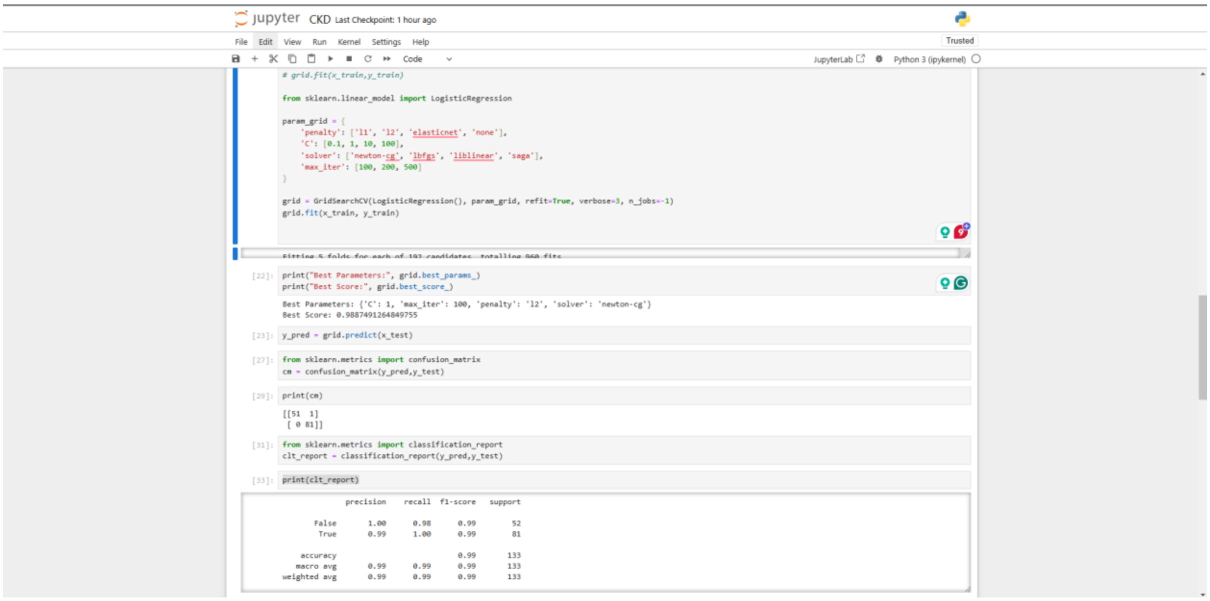
```python
[74]: from sklearn.metrics import classification_report
      clt_report = classification_report(y_pred,y_test)

[75]: print(clt_report)
```

```
              precision    recall  f1-score   support

       False       1.00      0.98      0.99        52
        True       0.99      1.00      0.99        81

    accuracy                           0.99       133
   macro avg       0.99      0.99      0.99       133
weighted avg       0.99      0.99      0.99       133
```



Final Model:

Algorithm: SVC

Classification Report:

Accuracy: 0.992522711390636

	precision	recall	f1-score	support
False	1.00	0.98	0.99	52
True	0.99	1.00	0.99	81
accuracy			0.99	133
macro avg	0.99	0.99	0.99	133
weighted avg	0.99	0.99	0.99	133

Precision: Measures the accuracy of positive predictions; 100% for False and 99% for True.

Recall: Measures the proportion of actual positives correctly identified; 98% for False and 100% for True.

F1-score: Harmonic mean of precision and recall; 0.99 for both False and True, reflecting balanced performance.

Support: Number of actual occurrences of each class; 52 for False and 81 for True.

Accuracy: Overall correctness of the model, achieving 99% accuracy across all 133 samples.

Justify The Final Model:

The dataset consists of numerical input features, making it suitable for machine learning algorithms.

The output is categorical, indicating that a classification algorithm is appropriate for this problem.

The selected model, SVC, achieves an overall accuracy of 99%, demonstrating strong performance in classifying the given data.

The high precision, recall, and F1-scores for both classes further validate the model's reliability.

Based on the accuracy and metrics, SVC effectively addresses the classification task.