Machine Learning Classification Assignment

Chronic Kidney Disease Prediction

1. Problem Statement

The objective of this project is to develop a Machine Learning model that can predict whether a patient is suffering from Chronic Kidney Disease (CKD) based on various medical parameters provided in the dataset. The model will help hospitals in early diagnosis and decision-making.

2. Dataset information

- Dataset Name: Chronic Kidney Disease Dataset
- Total Number of Rows: 399
- Total Number of Columns: 29
- Target Variable: classification_yes

Features include:

- Numerical features: age, bp, sg, al, su, bgr, bu, sc, sod, pot, hrmo, pcv, wc, rc
- Categorical features: rbc, pc, pcc, ba, htn, dm, cad, appet, pe, ane

3. Data Pre-processing

The following pre-processing steps were performed:

1. Converted categorical variables into numerical form using One-Hot Encoding.
2. Applied StandardScaler for algorithms such as:
    - Decision Tree
    - Random Forest
    - Support Vector Machine

4. Models developed

The following Machine Learning models were implemented:

1. Decision Tree
2. Random Forest
3. Support Vector Machine (SVM)
4. Logistic Regression
5. K-Nearest Neighbour (KNN)
6. Naive Bayes (Gaussian)

5. Evaluation Metrics Used

The models were evaluated using the following performance metrics:

- Confusion Matrix
- Accuracy
- Precision
- Recall
- F1 Score
- ROC AUC Score

6. Results of Each Model

The results of each models are given as follows:

6.1 Decision Tree Results

The Decision Tree model achieved an accuracy of **97%** and ROC score of **0.975** indicating good performance.

```
In [14]: print("The confusion Matrix:\n",cm)

The confusion Matrix:
 [[51  0]
 [ 4 78]]

In [15]: print("The report:\n",clf_report)

The report:
                precision    recall  f1-score   support

           0       0.93      1.00      0.96        51
           1       1.00      0.95      0.97        82

    accuracy                           0.97       133
   macro avg       0.96      0.98      0.97       133
weighted avg       0.97      0.97      0.97       133

In [16]: from sklearn.metrics import roc_auc_score

         roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])

Out[16]: 0.975609756097561
```

6.2 Random Forest Results

Random Forest achieved 98% accuracy and high ROC score showing strong performance.

```
In [16]: print("The confusion Matrix:\n",cm)

The confusion Matrix:
 [[50  1]
 [ 1 81]]

In [17]: print("The report:\n",clf_report)

The report:
                precision    recall  f1-score   support

           0       0.98      0.98      0.98        51
           1       0.99      0.99      0.99        82

    accuracy                           0.98       133
   macro avg       0.98      0.98      0.98       133
weighted avg       0.98      0.98      0.98       133

In [18]: from sklearn.metrics import roc_auc_score

         roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])

         C:\Anaconda3\lib\site-packages\sklearn\base.py:444: UserWa
         hout feature names
           f"X has feature names, but {self.__class__.__name__} was

Out[18]: 0.9866092778574844
```

6.3 Support Vector Machine (SVM)

SVM achieved the highest performance with 99% accuracy and ROC AUC score of 1.0.

```
In [16]: print("The confusion Matrix:\n",cm)

The confusion Matrix:
 [[51  0]
 [ 1 81]]
```

```
In [17]: print("The report:\n",clf_report)

The report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        51
           1       1.00      0.99      0.99        82

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

```
In [18]: from sklearn.metrics import roc_auc_score

         roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])

Out[18]: 1.0
```

6.4 Logistic Regression

Logistic Regression also achieved 99% accuracy with excellent precision and recall values.

```
In [14]: print("The confusion Matrix:\n",cm)

The confusion Matrix:
 [[50  1]
 [ 0 82]]
```

```
In [15]: from sklearn.metrics import classification_report
         clf_report = classification_report(y_test, grid_predictions)
```

```
In [16]: print("The report:\n",clf_report)

The report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        51
           1       0.99      1.00      0.99        82

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

```
In [17]: from sklearn.metrics import roc_auc_score

         roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])

Out[17]: 0.9992826398852224
```

6.5 K-Nearest Neighbour (KNN)

KNN showed lower performance with **79% accuracy**, indicating it is less suitable for this dataset.

```
In [30]: print("The confusion Matrix:\n",cm)

The confusion Matrix:
 [[46  5]
 [23 59]]

In [31]: from sklearn.metrics import classification_report
clf_report = classification_report(y_test, grid_predictions)

In [32]: print("The report:\n",clf_report)

The report:
              precision    recall  f1-score   support

           0       0.67      0.90      0.77        51
           1       0.92      0.72      0.81        82

    accuracy                           0.79       133
   macro avg       0.79      0.81      0.79       133
weighted avg       0.82      0.79      0.79       133

In [33]: from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])

Out[33]: 0.8433763749402198
```

6.6 Naive Bayes

Naive Bayes performed well with 98% accuracy and strong recall values.

```
In [10]: print("The confusion Matrix:\n",cm)

The confusion Matrix:
 [[51  0]
 [ 3 79]]

In [11]: print("The report:\n",clf_report)

The report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        51
           1       1.00      0.96      0.98        82

    accuracy                           0.98       133
   macro avg       0.97      0.98      0.98       133
weighted avg       0.98      0.98      0.98       133

In [12]: from sklearn.metrics import roc_auc_score

roc_auc_score(y_test,classifier.predict_proba(X_test)[:,1])

Out[12]: 1.0
```

7. Comparison Table

Model	Accuracy	Precision	Recall	F1-score	ROC
Decision Tree	0.97	0.97	0.97	0.97	0.975
Random Forest	0.98	0.98	0.98	0.98	0.986
SVM	0.99	0.99	0.99	0.99	1.0
Logistic Regression	0.99	0.99	0.99	0.99	0.999
KNN	0.79	0.82	0.79	0.79	0.84
Naïve Bayes	0.98	0.98	0.98	0.98	1.0

8. Final Model Selection

Among all the models, Logistic Regression and SVM provided the best performance.**Logistic Regression** was selected as the final model because:

- It provides very high accuracy (99%), which is almost equal to SVM
- It is simple and interpretable
- Suitable for medical decision-making systems

Even though SVM showed slightly better numerical performance, Logistic Regression is more suitable for real-world hospital usage due to its interpretability and reliability.

9. Conclusion

The project successfully developed a Machine Learning model to predict Chronic Kidney Disease. The final model achieved high accuracy and can be used to assist doctors in early diagnosis and treatment planning.