Machine Learning – Classification - Assignment

Problem Statement:

A requirement from the Hospital, Management asked us to create a predictive model which will predict the Chronic Kidney Disease (CKD).

Solution:

3 stages :

1. Domain Selection – Machine learning (data is structured (i.e.)in Excel)
2. Learning – Supervised Learning (Input and Output is clear and present in data itself)
3. Regression/Classification – Classification (Output is classified as Yes or No)

No.Of Rows in Data sheet– 399

No.Of columns – 25 (24 Input and 1 Output)

12 Columns have nominal data. So need to use One-Hot Encoding to convert the string to number.

After converting No.Of columns – 28 (27 Input and 1 Output)

Input Fields	Output Field
'age', 'bp', 'al', 'su', 'bgr', 'bu', 'sc', 'sod', 'pot', 'hrmo', 'pcv', 'wc', 'rc', 'sg_b', 'sg_c', 'sg_d', 'sg_e', 'rbc_normal', 'pc_normal', 'pcc_present', 'ba_present', 'htn_yes', 'dm_yes', 'cad_yes', 'appet_yes', 'pe_yes', 'ane_yes'	classification_yes

Algorithms:

1. Decision Tree – Accuracy 0.97

	precision	recall	f1-score	support
0	0.96	0.96	0.96	51
1	0.98	0.98	0.98	82
accuracy			0.97	133
macro avg	0.97	0.97	0.97	133
weighted avg	0.97	0.97	0.97	133

```
print(cm)
```

```
[[49  2]
 [ 2 80]]
```

```
print("Best parameters:", grid.best_params_)
print("Best accuracy:", grid.best_score_)
```

```
Best parameters: {'criterion': 'log_loss', 'max_depth': 20, 'max_features': 'log2'}
Best accuracy: 0.973654786862334
```

2. Support Vector Machine - Accuracy 0.98

```
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.98      0.96      0.97        51
           1       0.98      0.99      0.98        82

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

```
print(cm)
```

```
[[49  2]
 [ 1 81]]
```

```
print("Best parameters:", grid.best_params_)
print("Best accuracy:", grid.best_score_)
```

```
Best parameters: {'C': 10, 'gamma': 'scale', 'kernel': 'linear'}
Best accuracy: 0.9586303284416491
```

3. Random Forest – Accuracy 0.99

```
print(clf_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
```

```
print(cm)
```

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

```
print("Best parameters:", grid.best_params_)
print("Best accuracy:", grid.best_score_)
```

```
Best parameters: {'criterion': 'entropy', 'max_features': 'log2', 'n_estimators': 100}
Best accuracy: 0.9849755415793151
```

4. KNN – Accuracy 0.82

```
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.70      0.92      0.80        51
           1       0.94      0.76      0.84        82

    accuracy                           0.82       133
   macro avg       0.82      0.84      0.82       133
weighted avg       0.85      0.82      0.82       133
```

```
print(cm)
```

```
[[47  4]
 [20 62]]
```

```
print("Best parameters:", grid.best_params_)
print("Best accuracy:", grid.best_score_)
```

```
Best parameters: {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'}
Best accuracy: 0.7518518518518519
```

5. XGBoost – Accuracy 0.98

```
print(clf_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
```

```
print(cm)
```

```
[[50  1]
 [ 1 81]]
```

```
print("Best parameters:", grid.best_params_)
print("Best accuracy:", grid.best_score_)
```

```
Best parameters: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 100}
Best accuracy: 0.9548567435359889
```

6. Navie's Bayes – Accuracy 0.94 (Best of 3)

```
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
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(clf_report)
print(cm)
```

```
              precision    recall  f1-score   support

           0       0.68      0.98      0.81        51
           1       0.98      0.72      0.83        82

    accuracy                           0.82       133
   macro avg       0.83      0.85      0.82       133
weighted avg       0.87      0.82      0.82       133

[[50  1]
 [23 59]]
```

```
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(clf_report)
print(cm)
```

```
              precision    recall  f1-score   support

           0       0.86      1.00      0.93        51
           1       1.00      0.90      0.95        82

    accuracy                           0.94       133
   macro avg       0.93      0.95      0.94       133
weighted avg       0.95      0.94      0.94       133

[[51  0]
 [ 8 74]]
```

```
from sklearn.naive_bayes import ComplementNB
classifier =ComplementNB()
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(clf_report)
print(cm)
```

```
              precision    recall  f1-score   support

           0       0.68      0.98      0.81        51
           1       0.98      0.72      0.83        82

    accuracy                           0.82       133
   macro avg       0.83      0.85      0.82       133
weighted avg       0.87      0.82      0.82       133

[[50  1]
 [23 59]]
```

S.No	Algorithms	Accuracy
1	Decision Tree	0.97
2	Support Vector Machine	0.98
3	Random Forest	0.99
4	K Nearest Neighbour	0.82
5	XGBoost	0.98
6	Navies Bayes	0.94

By comparing the Accuracy value of all the algorithms , Random Forest Model has good performance with 99% accuracy.

Conclusion:

The final best model for the problem statement – Predicting Chronic Kidney Disease (CKD) is **Random Forest with Accuracy 99%.**