Problem Statement:- Goal is to to design a training model which can identify who are prone to kidney disease by providing patient details like BP, age, sugar level, Haemoglobin level.

Problem type is classification .

Dataset has 399 rows and 28 columns.

Independent dataset is pre-processed using standardscaler method. Training set is transformed using fit_transform to calculate mean and standard deviation of the values. Test set is only standardised using transform method.

1)Logistic Grid classification algorithm

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
param_grid = {'solver':['newton-cg', 'lbfgs', 'liblinear', 'saga'],
          'penalty':['l2']}
grid = GridSearchCV(LogisticRegression(),param_grid,refit = True,verbose =3,n_jobs=-
1,scoring='f1_weighted')
# fitting the model for grid search
grid.fit(X_train, y_train)
```

Outcome:

```
The f1_macro value for best parameter {'penalty': 'l2', 'solver': 'newton-cg'}: 0.9924946382275899
```

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

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

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

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

       False       0.98      1.00      0.99        51
        True       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
```

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

```
[29]: np.float64(1.0)
```

## 2)Decision Tree classification Algorithm

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
param_grid = {'criterion':['gini','entropy'],
              'max_features': ['auto','sqrt','log2'],
              'splitter':['best','random']}
grid = GridSearchCV(DecisionTreeClassifier(), param_grid, refit = True, verbose = 3,n_jobs=-
1,scoring='f1_weighted')
# fitting the model for grid search
grid.fit(X_train, y_train)
```

Outcome:-

```
The f1_macro value for best parameter {'criterion': 'entropy', 'max_features': 'sqrt', 'splitter': 'ran
dom'}: 0.9775556904684072
```

[12]:
```
print("The confusion Matrix:\n",cm)
```

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

[13]:
```
print("The report:\n",clf_report)
```

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

       False       0.94      1.00      0.97        51
        True       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
```

[14]:
```
from sklearn.metrics import roc_auc_score

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

[14]:
```
np.float64(0.9817073170731707)
```

3) <u>SVM classification Algorithm</u>

```
from sklearn.svm import SVC
grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3,n_jobs=-
1,scoring='f1_weighted')

# fitting the model for grid search
grid.fit(X_train, y_train)
```

Outcome:-

```
The f1_macro value for best parameter {'C': 10, 'degree': 2, 'gamma': 'auto', 'kernel': 'sigmoid'}: 0.9
924946382275899

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

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

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

The report:
              precision    recall  f1-score   support

       False       0.98      1.00      0.99        51
        True       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
```

4)RandomForest Classification

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
param_grid = {'criterion':['gini','entropy'],
        'max_features': ['auto','sqrt','log2'],
        'n_estimators':[10,100]}
grid = GridSearchCV(RandomForestClassifier(), param_grid, refit = True, verbose = 3,n_jobs=-
1,scoring='f1')

# fitting the model for grid search
grid.fit(X_train, y_train)
```

Outcome:-

```
The f1_macro value for best parameter {'criterion': 'entropy', 'max_features': 'log2', 'n_estimators':
100}: 1.0
The confusion Matrix:
 [[51  0]
 [ 0 82]]
```

[14]:
```
print("The report:\n",clf_report)
```

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

       False       1.00      1.00      1.00        51
        True       1.00      1.00      1.00        82

    accuracy                           1.00       133
   macro avg       1.00      1.00      1.00       133
weighted avg       1.00      1.00      1.00       133
```

Accuracy of f1 score is 1.00 which is very close to the maximum outcome and so choosing RF algorithm.