SOLUTION CLASSIFICATION

1. Stage1- Machine Learning
   Stage2- Supervised Learning
   Stage3- Classification- as we have categorical values as output

2. Dataset contains 399 rows, 25- columns

3. Converted categorical data to nominal data using one hot encoding method.

4. Best Model for the given problem is Random Forest with below parameters
   {'class_weight': 'balanced', 'criterion': 'entropy', 'max_features': 'sqrt', 'n_estimators': 50}

5. Results of Algorithms:

   Random Forest

```
In [16]: print(clf_report)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        51
           1       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
```

```
In [17]: from sklearn.metrics import roc_auc_score
         roc_score= roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
         roc_score
```

```
Out[17]: 0.9999999999999999
```

   Decision tree

```
In [15]: print(clf_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 [16]: from sklearn.metrics import roc_auc_score
         roc_score= roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
         roc_score
```

```
Out[16]: 0.9817073170731707
```

SVC

```
In [15]: print(clf_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 [16]: from sklearn.metrics import roc_auc_score
         roc_score= roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
         roc_score
```

```
Out[16]: 1.0
```

Logistic Regression:

```
In [14]: from sklearn.metrics import classification_report
         clf_report=classification_report(y_test,y_pred)
         print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        51
           1       1.00      0.98      0.99        82

    accuracy                           0.98       133
   macro avg       0.98      0.99      0.98       133
weighted avg       0.99      0.98      0.99       133
```

```
In [16]: from sklearn.metrics import roc_auc_score
         roc_score= roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
         roc_score
```

```
Out[16]: 1.0
```

KNN

```
In [15]: print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.85      1.00      0.92        51
           1       1.00      0.89      0.94        82

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

```
In [16]: from sklearn.metrics import roc_auc_score
         roc_score= roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
         roc_score
```

```
Out[16]: 1.0
```

GaussianNB

```
In [16]: print(clf_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 [17]: from sklearn.metrics import roc_auc_score
         roc_score= roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
         roc_score
```

```
Out[17]: 1.0
```

Bernoulli NB

```
In [15]: print(clf_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 [16]: from sklearn.metrics import roc_auc_score
         roc_score= roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
         roc_score
```

```
Out[16]: 1.0
```

Multinomial NB

```
In [15]: print(clf_report)
```

```
                precision    recall  f1-score   support

           0       0.89      0.98      0.93        51
           1       0.99      0.93      0.96        82

    accuracy                           0.95       133
   macro avg       0.94      0.95      0.95       133
weighted avg       0.95      0.95      0.95       133
```

```
In [16]: from sklearn.metrics import roc_auc_score
         roc_score= roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
         roc_score
```

```
Out[16]: 0.9851745576279292
```

Complement NB

```
In [14]: print(clf_report)
```

```
                precision    recall  f1-score   support

           0       0.71      0.98      0.83        51
           1       0.98      0.76      0.86        82

    accuracy                           0.84       133
   macro avg       0.85      0.87      0.84       133
weighted avg       0.88      0.84      0.84       133
```

```
In [15]: from sklearn.metrics import roc_auc_score
         roc_score= roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
         roc_score
```

```
Out[15]: 0.9356767097082734
```

Categorical NB

```
In [14]: print(clf_report)
```

```
                precision    recall  f1-score   support

           0       0.99      1.00      1.00       150
           1       1.00      1.00      1.00       249

    accuracy                           1.00       399
   macro avg       1.00      1.00      1.00       399
weighted avg       1.00      1.00      1.00       399
```

```
In [15]: from sklearn.metrics import roc_auc_score
         roc_score= roc_auc_score(dep,grid.predict_proba(indep)[:,1])
         roc_score
```

```
Out[15]: 0.9999196787148594
```

6. Best Model for the given problem is Random Forest with below parameters {'class_weight': 'balanced', 'criterion': 'entropy', 'max_features': 'sqrt', 'n_estimators': 50}. Random Forest has provided more accurate predictions, also has flexibility of standardisation and can be used for different range of inputs.