**Classification Assignment**

1. Problem Statement

🡪To identify whether patience have Chronic Kidney disease or not

1. About data set

* Data set as total of 25 column in which 24 fields acts as input and 1 will be the output
* Total data set record is 399
* We perform one-hot encoding using get\_dummies

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

dataset**=**pd.read\_csv("C:\Prakash\AI\Learning\Machine\_learning\Classification\Classification\_Assignment\CKD.csv")

dataset

Out[3]:

|  | **age** | **bp** | **sg** | **al** | **su** | **rbc** | **pc** | **pcc** | **ba** | **bgr** | **...** | **pcv** | **wc** | **rc** | **htn** | **dm** | **cad** | **appet** | **pe** | **ane** | **classification** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 2.000000 | 76.459948 | c | 3.0 | 0.0 | normal | abnormal | notpresent | notpresent | 148.112676 | ... | 38.868902 | 8408.191126 | 4.705597 | no | no | no | yes | yes | no | yes |
| 1 | 3.000000 | 76.459948 | c | 2.0 | 0.0 | normal | normal | notpresent | notpresent | 148.112676 | ... | 34.000000 | 12300.000000 | 4.705597 | no | no | no | yes | poor | no | yes |
| 2 | 4.000000 | 76.459948 | a | 1.0 | 0.0 | normal | normal | notpresent | notpresent | 99.000000 | ... | 34.000000 | 8408.191126 | 4.705597 | no | no | no | yes | poor | no | yes |
| 3 | 5.000000 | 76.459948 | d | 1.0 | 0.0 | normal | normal | notpresent | notpresent | 148.112676 | ... | 38.868902 | 8408.191126 | 4.705597 | no | no | no | yes | poor | yes | yes |
| 4 | 5.000000 | 50.000000 | c | 0.0 | 0.0 | normal | normal | notpresent | notpresent | 148.112676 | ... | 36.000000 | 12400.000000 | 4.705597 | no | no | no | yes | poor | no | yes |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 394 | 51.492308 | 70.000000 | a | 0.0 | 0.0 | normal | normal | notpresent | notpresent | 219.000000 | ... | 37.000000 | 9800.000000 | 4.400000 | no | no | no | yes | poor | no | yes |
| 395 | 51.492308 | 70.000000 | c | 0.0 | 2.0 | normal | normal | notpresent | notpresent | 220.000000 | ... | 27.000000 | 8408.191126 | 4.705597 | yes | yes | no | yes | poor | yes | yes |
| 396 | 51.492308 | 70.000000 | c | 3.0 | 0.0 | normal | normal | notpresent | notpresent | 110.000000 | ... | 26.000000 | 9200.000000 | 3.400000 | yes | yes | no | poor | poor | no | yes |
| 397 | 51.492308 | 90.000000 | a | 0.0 | 0.0 | normal | normal | notpresent | notpresent | 207.000000 | ... | 38.868902 | 8408.191126 | 4.705597 | yes | yes | no | yes | poor | yes | yes |
| 398 | 51.492308 | 80.000000 | a | 0.0 | 0.0 | normal | normal | notpresent | notpresent | 100.000000 | ... | 53.000000 | 8500.000000 | 4.900000 | no | no | no | yes | poor | no | no |

399 rows × 25 columns

dataset**=**pd.get\_dummies(dataset,drop\_first**=True**)

dataset.columns

Out[9]:

Index(['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'],

dtype='object')

dataset

Out[11]:

|  | **age** | **bp** | **al** | **su** | **bgr** | **bu** | **sc** | **sod** | **pot** | **hrmo** | **...** | **pc\_normal** | **pcc\_present** | **ba\_present** | **htn\_yes** | **dm\_yes** | **cad\_yes** | **appet\_yes** | **pe\_yes** | **ane\_yes** | **classification\_yes** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 2.000000 | 76.459948 | 3.0 | 0.0 | 148.112676 | 57.482105 | 3.077356 | 137.528754 | 4.627244 | 12.518156 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 3.000000 | 76.459948 | 2.0 | 0.0 | 148.112676 | 22.000000 | 0.700000 | 137.528754 | 4.627244 | 10.700000 | ... | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 2 | 4.000000 | 76.459948 | 1.0 | 0.0 | 99.000000 | 23.000000 | 0.600000 | 138.000000 | 4.400000 | 12.000000 | ... | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 3 | 5.000000 | 76.459948 | 1.0 | 0.0 | 148.112676 | 16.000000 | 0.700000 | 138.000000 | 3.200000 | 8.100000 | ... | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 4 | 5.000000 | 50.000000 | 0.0 | 0.0 | 148.112676 | 25.000000 | 0.600000 | 137.528754 | 4.627244 | 11.800000 | ... | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 394 | 51.492308 | 70.000000 | 0.0 | 0.0 | 219.000000 | 36.000000 | 1.300000 | 139.000000 | 3.700000 | 12.500000 | ... | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 395 | 51.492308 | 70.000000 | 0.0 | 2.0 | 220.000000 | 68.000000 | 2.800000 | 137.528754 | 4.627244 | 8.700000 | ... | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 396 | 51.492308 | 70.000000 | 3.0 | 0.0 | 110.000000 | 115.000000 | 6.000000 | 134.000000 | 2.700000 | 9.100000 | ... | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 397 | 51.492308 | 90.000000 | 0.0 | 0.0 | 207.000000 | 80.000000 | 6.800000 | 142.000000 | 5.500000 | 8.500000 | ... | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 398 | 51.492308 | 80.000000 | 0.0 | 0.0 | 100.000000 | 49.000000 | 1.000000 | 140.000000 | 5.000000 | 16.300000 | ... | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

399 rows × 28 columns

dataset.columns

Out[12]:

Index(['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'],

dtype='object')

dataset["classification\_yes"].value\_counts()

Out[14]:

1 249

0 150

Name: classification\_yes, dtype: int64

independ**=**dataset[['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']]

depend**=**dataset[['classification\_yes']]

*#split into training set and test*

**from** sklearn.model\_selection **import** train\_test\_split

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(independ, depend, test\_size **=** 1**/**3, random\_state **=** 0)

*#Multinominal Navie Bayes*

**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]]

TP=50,FP=1,FN=23,TN=59

*#Bernoulli Naive\_bayes*

**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]]

TP=51,FP=0,FN=8,TN=74

*#Categorical Naive bayes*

**from** sklearn.naive\_bayes **import** CategoricalNB

classifier **=**CategoricalNB()

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)

C:\Users\admin\Anaconda3\lib\site-packages\sklearn\utils\validation.py:985: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

y = column\_or\_1d(y, warn=True)

**---------------------------------------------------------------------------**

**IndexError** Traceback (most recent call last)

**<ipython-input-20-cacafe0e0af4>** in <module>

3 classifier **=**CategoricalNB**()**

4 classifier**.**fit**(**x\_train**,** y\_train**)**

**----> 5** y\_pred **=** classifier**.**predict**(**x\_test**)**

6 **from** sklearn**.**metrics **import** confusion\_matrix

7 cm **=** confusion\_matrix**(**y\_test**,** y\_pred**)**

**~\Anaconda3\lib\site-packages\sklearn\naive\_bayes.py** in predict**(self, X)**

81 check\_is\_fitted**(**self**)**

82 X **=** self**.**\_check\_X**(**X**)**

**---> 83** jll **=** self**.**\_joint\_log\_likelihood**(**X**)**

84 **return** self**.**classes\_**[**np**.**argmax**(**jll**,** axis**=1)]**

85

**~\Anaconda3\lib\site-packages\sklearn\naive\_bayes.py** in \_joint\_log\_likelihood**(self, X)**

1459 **for** i **in** range**(**self**.**n\_features\_in\_**):**

1460 indices **=** X**[:,** i**]**

**-> 1461** jll **+=** self**.**feature\_log\_prob\_**[**i**][:,** indices**].**T

1462 total\_ll **=** jll **+** self**.**class\_log\_prior\_

1463 **return** total\_ll

**IndexError**: index 90 is out of bounds for axis 1 with size 84

*#complement naive bayes*

**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]]

TP=50,FP=1,FN=23,TN=59

*#Gaussian naive bayes*

**from** sklearn.naive\_bayes **import** GaussianNB

classifier **=**GaussianNB()

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 0.94 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

[[51 0]

[ 3 79]]

TP=50,FP=0,FN=3,TN=79

*#decision tree*

**from** sklearn.tree **import** DecisionTreeClassifier

classifier **=**DecisionTreeClassifier(criterion **=** 'entropy', random\_state **=** 0)

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 0.96 0.91 51

1 0.97 0.90 0.94 82

accuracy 0.92 133

macro avg 0.92 0.93 0.92 133

weighted avg 0.93 0.92 0.93 133

[[49 2]

[ 8 74]]

TP=49,FP=2,FN=8,TN=74

*#Random forest*

**from** sklearn.ensemble **import** RandomForestClassifier

classifier **=** RandomForestClassifier(n\_estimators **=** 10, criterion **=** 'entropy', random\_state **=** 0)

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 0.98 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

[[50 1]

[ 1 81]]

TP=50,FP=1,FN=1,TN=81

*#Logistic regression*

**from** sklearn.linear\_model **import** LogisticRegression

classifier **=** LogisticRegression( random\_state **=** 0)

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 0.91 0.94 0.92 51

1 0.96 0.94 0.95 82

accuracy 0.94 133

macro avg 0.93 0.94 0.94 133

weighted avg 0.94 0.94 0.94 133

[[48 3]

[ 5 77]]

TP=48,FP=3,FN=6,TN=77

*#KNN classification*

**from** sklearn.neighbors **import** KNeighborsClassifier

classifier**=**KNeighborsClassifier(n\_neighbors**=**7,metric**=**'minkowski',p**=**2)

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 0.57 0.78 0.66 51

1 0.83 0.63 0.72 82

accuracy 0.69 133

macro avg 0.70 0.71 0.69 133

weighted avg 0.73 0.69 0.70 133

[[40 11]

[30 52]]

TP=40,FP=11,FN=30,TN=52

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| module | Confusion Matrix | TP | FP | FN | TN |
| Multinominal NB | [[50 1]  [23 59]] | 50 | 1 | 23 | 59 |
| Categorical NB |  |  |  |  |  |
| Gausian NB | [[51 0]  [ 3 79]] | 51 | 0 | 23 | 59 |
| complement NB | [[50 1]  [23 59]] | 50 | 1 | 23 | 59 |
| Bernoulli NB | [[51 0]  [ 8 74]] | 51 | 0 | 8 | 74 |
| Random Forest | [[50 1]  [ 1 81]] | 50 | 1 | 1 | 81 |
| Decision Tree | [[49 2]  [ 8 74]] | 49 | 2 | 8 | 74 |
| Logistic | [[48 3]  [ 5 77]] | 48 | 3 | 5 | 77 |
| KNN | [[40 11]  [30 52]] | 40 | 11 | 30 | 52 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Parameter | Precision | | Recall | | F1-Score | |  |  |  |
| module | Confusion Matrix | Accuracy | Not Purchased | Purchased | Not Purchased | Purchased | Not Purchased | Purchased | Specificity TN/(TN+FP) | Type 1 error FP/TN+FP | Type 2 error FP/TP+FN |
| Multinominal NB | [[50 1]  [23 59]] | 82 | 0.68 | 0.98 | 0.98 | 0.72 | 0.81 | 0.83 | 0.9833333 | 0.01666667 | 0.01369863 |
| Categorical NB |  |  |  |  |  |  |  |  |  |  |  |
| Gausian NB | [[51 0]  [ 3 79]] | 0.98 | 0.94 | 1 | 1 | 0.96 | 0.97 | 0.98 | 1 | 0 | 0 |
| complement NB | [[50 1]  [23 59]] | 0.82 | 0.68 | 0.98 | 0.98 | 0.72 | 0.81 | 0.83 | 0.9833333 | 0.01666667 | 0.01369863 |
| Bernoulli NB | [[51 0]  [ 8 74]] | 0.94 | 0.86 | 1 | 1 | 0.9 | 0.93 | 0.95 | 1 | 0 | 0 |
| Random Forest | [[50 1]  [ 1 81]] | 0.98 | 0.98 | 0.99 | 0.98 | 0.99 | 0.98 | 0.99 | 0.9878049 | 0.01219512 | 0.01960784 |
| Decision Tree | [[49 2]  [ 8 74]] | 0.92 | 0.86 | 0.97 | 0.96 | 0.9 | 0.91 | 0.94 | 0.9736842 | 0.02631579 | 0.03508772 |
| Logistic | [[48 3]  [ 5 77]] | 0.94 | 0.91 | 0.96 | 0.94 | 0.94 | 0.92 | 0.95 | 0.9625 | 0.0375 | 0.05660377 |
| KNN | [[40 11]  [30 52]] | 0.69 | 0.57 | 0.83 | 0.78 | 0.63 | 0.66 | 0.72 | 0.8253968 | 0.17460317 | 0.15714286 |

**Conclusion**

Out of all models two model gives high accuracy rate which is 98%

Gausian Naïve bayes and Random forest