STATISTICAL DATA ANALYSIS and MACHINE LEARNING

July 14, 2020

1 INTRODUCTION AND SUMMARY

Machine learning models are based on statistics, optimization and computer science. Our project aims to compare the performance of a few machine learning models on different databases. For this analysis, we used the Python programming language on Jupyter notebooks. The project will be realized in 2 terms and we conducted the experimental part in our first period. We will present the theoretical discussion of mathematical basis of the models use in the fall period of 2021.

1.1 PROJECT PLAN

1.1.1 Aim of the Project

We aim to perform rigorous statistical analysis of 6 different machine learning models on 3 different datasets.

1.1.2 Scope of the Project

Within the scope of this project, 6 different machine learning models were selected from well-known and commonly used python libraries. A deeper theoretical exploration about the models will be given at the thesis stage. Since we have devoted our first period to experiments, we will show an explanatory code section of the project in this report.

1.1.3 Area of Usage

Within the scope of the analysis, the final version of the project can be used as a resource for those who want to learn the mathematical background of machine learning models, and how they can be applied to real-world data sets. We aim that the thesis will be a resource for those who want to learn from the beginning which models work best for which dataset.

1.1.4 Schedule

Task	Alloted Time
Research and discovery of databases	2 weeks
Clearing and preparing datasets	2 weeks
Determining the models and libraries to be used	1 week
Coding phase and evaluation of outputs	10 weeks
General arrangement of the code stage and annotation	2 weeks
Writing the report	3 weeks

1.1.5 Resources

All databases, libraries and models used are open source. All relevant references will be made to the books, articles and theses used for theory in the reference section in the thesis section. All work is stored on github. (https://github.com/Code-Cash/ahmet)

2 EXPERIMENTS

Our project is based on a through and rigorous statistical analysis of 6 different machine learning algorithms over 3 databases.

The code block you see below is the experimental stage of this thesis. We give descriptions about each cell just above the corresponding cell.

2.1 Loading the libraries

At the beginning of our project, we need to add the libraries required for our project.

[2]:
```
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
from sklearn.preprocessing import scale
from sklearn.model_selection import cross_val_score
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from PIL import Image
from sklearn.metrics import accuracy_score

import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.anova import AnovaRM

from matplotlib import pyplot as plt
from matplotlib import pyplot
import seaborn as sns
import xgboost as xgb
from sklearn.naive_bayes import GaussianNB

from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

from sklearn.feature_selection import RFE
```

2.2 Loading the models

We load 6 blank machine learning models after adding our libraries.

[3]:
```python
dt = DecisionTreeClassifier()
rf = RandomForestClassifier(n_estimators=100, random_state=0, n_jobs = -1)
logreg = LogisticRegression(solver='liblinear',multi_class='ovr')
svm = SVC(gamma='auto')
xgb = xgb.XGBClassifier()
gnb = GaussianNB()
```

2.3 Loading the first data set

We add our first data set using the pandas library. Then we display the first 5 elements of our dataset using the panda's head function.

[5]:
```python
database1 = pd.read_csv("data.csv", sep=';')
database1.head()
```

[5]:
```
   MUSK   2    3    4   5    6   7   8    9  10  …  159  160  161  162  163  \
0  MUSK  46 -108  -60 -69 -117  49  38 -161  -8  … -308   52   -7   39  126
1  MUSK  41 -188 -145  22 -117  -6  57 -171 -39  …  -59   -2   52  103  136
2  MUSK  46 -194 -145  28 -117  73  57 -168 -39  … -134 -154   57  143  142
3  MUSK  41 -188 -145  22 -117  -7  57 -170 -39  …  -60   -4   52  104  136
4  MUSK  41 -188 -145  22 -117  -7  57 -170 -39  …  -60   -4   52  104  137

   164  165  166  167  168
0  156  -50 -112   96  1.0
1  169  -61 -136   79  1.0
2  165  -67 -145   39  1.0
3  168  -60 -135   80  1.0
4  168  -60 -135   80  1.0

[5 rows x 168 columns]
```

2.4 Feature selection and splitting the train and test data sets

Let us denote the features we are going to use with `X` and response variable with `y`. With the drop function, we get the columns other than the columns 'musk' and '168'. For the response variable, we just take the column named '168'. Then we determine the size of our training and test sets. For this dataset, 90% of all data is used for training and the remaining 10% is used for testing using the function "train_test_split".

[7]:
```python
X = database1.drop(["MUSK","168"],axis=1)
y = database1['168']
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.10 )
```

2.5 A generic function for training and evaluating a model

At this stage, we create a function that will run our 6 algorithms for our first dataset and give us 3 outputs.

If we go into detail: first, we train our model with the features `X`, which is the training group we mentioned in Section 4, with the features in `y` as dependent variable. Then we make a prediction with the predict function according to the test data set. We record this prediction as `y_pred`. The test data set `y_test` is the test set of the results will give us the accuracy of the model. Then we print the classification report and confusion matrix with the help of the scikit-learn library. The explanation of all these functions will be found in the theory section of our thesis.

[4]:
```
def model(name, X_train, y_train, X_test, y_test):
    name.fit(X_train, y_train)
    y_pred = name.predict(X_test)
    score= accuracy_score(y_test, y_pred) * 100
    print(str(name) + "Accuracy:",score)
    report_name=classification_report(y_test, y_pred)
    print(report_name)
    print(confusion_matrix(y_test,y_pred))
```

2.6 Recursive Feature Elimination

The purpose of the function below is to find the most suitable features for us in our database. The `rfe` (Recursive Feature Elimination) function produces an output based on the model we want and the number of desired features. It shows them in order, according to their effect.

[5]:
```
def rfe(model,X,y):
    rfe = RFE(model, 16)
    fit = rfe.fit(X, y)
    print("Num Features: %s" % (fit.n_features_))
    print("Selected Features: %s" % (fit.support_))
    print("Feature Ranking: %s" % (fit.ranking_))
```

2.7 Cross validation

Cross validation calculates the average accuracy and standard deviation for each part by dividing our data into as many parts as we want. This gives us an important idea of how well the model works.

[6]:
```
def crossval(model,X,y,n=10):
    scores=cross_val_score(model, X, y, cv=n, scoring ="accuracy")
    print(str(model) + "Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.
↪std() * 2))
```

2.8 Experiments on the first data set

2.8.1 Decision Tree

Let us evaluate the Decision Tree model on our first data set.

[15]:
```
model(dt,X_train,y_train,X_test,y_test)
crossval(dt,X,y,n)
rfe(dt,X,y)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')Accuracy:
96.21212121212122
              precision    recall  f1-score   support

         0.0       0.97      0.98      0.98       549
         1.0       0.90      0.87      0.89       111

    accuracy                           0.96       660
   macro avg       0.94      0.93      0.93       660
weighted avg       0.96      0.96      0.96       660

[[538  11]
 [ 14  97]]
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')Accuracy: 0.78 (+/-
0.40)
Num Features: 16
Selected Features: [False False False False False False False False  True False
False False
 False False False False False False False False False False False False
 False False False False False False False  True False False False  True
 False  True False False False False  True False False False False False
 False  True False False False False False False False False False False
 False False False False False  True False False False False False False
 False False False False  True False False False False False  True False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False  True False  True False  True False False False False False  True
 False False False False False False False  True False False False False
 False False False False False False  True False False False False False
 False False False False False False  True False False False]
Feature Ranking: [146 145  45  62  46  32  66  61   1  53  71  70  75  60  78
40   9  69
```

```
 83  15  93  12  90  88  24 103   8 108  43 110  85   1 122  41  34   1
 54   1  33  74  80  18   1 136 141  86 134  36  56   1 139   3  68   2
  6  37  95  98  92  58  38  82  21  26 140   1 117  50  35 118 128 100
144 130  17 143   1 113 119 123 121 149   1 142   4 138  89  31   5 137
 77 106  14  59  84  16  91  76  11  73  25  65  27  63 109  10  81  28
 64 107  51  67 120 105 101  30  22  96 114  52  29   1  97   1  99   1
127 147 129  47  49   1   7 125 124  44 102  94 111   1  19  48  39 135
132 133  87 126 115 131   1  23  72 116 104  42 112  13  79  55  57  20
  1 148 150 151]
```

2.8.2 Random Forest

Let us evaluate the Random Forest algorithm on our first data set.

[17]:
```
model(rf,X_train,y_train,X_test,y_test)
crossval(rf,X,y)
rfe(rf,X,y)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=-1, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)Accuracy: 97.12121212121212
              precision    recall  f1-score   support

         0.0       0.97      1.00      0.98       549
         1.0       0.99      0.84      0.91       111

    accuracy                           0.97       660
   macro avg       0.98      0.92      0.95       660
weighted avg       0.97      0.97      0.97       660

[[548   1]
 [ 18  93]]
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=-1, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)Accuracy: 0.80 (+/- 0.41)
Num Features: 16
Selected Features: [False False False False False False False False  True False
False False
```

```
 False False False False False False False False False False False False
 False False False False False False False False False False False  True
 False False False False False False False False False False False False
 False  True False  True False False False False False False False False
 False False False False False  True False False False False False False
 False False False False False False False False False False False False
 False False False False False False False  True False False  True False
 False False False False False False False False False False False False
 False  True False False False False False False False False False False
 False  True False  True False  True False False False False False  True
 False False False False False False False  True False False False False
 False False False False False False  True False False False False False
 False False False False False  True  True False False False]
Feature Ranking: [  7 143  50  72 150  77  87  83   1  20 110 136  45  19 128
145  28 123
  38  44   2   8  58 121  17  95 131 134  94  96 106  25  43  53  24   1
 140  40  74 130  76  29   5  69   4 125 144  68  86   1  49   1  56  98
  32  59 147  90  42 105  10  97   9  92 116   1 151  47 138 113 108  31
 122 126  62 149  91  82  23  39  89  22  12  75  11  16 139 119  26 114
 127   1  61  81   1  27 120  60  71 141 115  13  55  63  35  93  73  46
  15   1  66  30 102  99  37  36  65  54 135 112  84   1  52   1  41   1
 103  57  78  64  51   1  18  80  67 129 137 101 104   1  21 132  70  33
 142  88 148 118 117  79   1 124  85 100 107 111 146 109 133   3  48   1
   1  34   6  14]
```

2.8.3 Logistic Regression

Let us evaluate the Logistic Regression algorithm on our first data set.

[18]:
```
model(logreg,X_train,y_train,X_test,y_test)
crossval(logreg,X,y)
rfe(logreg,X,y)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='ovr', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)Accuracy: 95.0
              precision    recall  f1-score   support

         0.0       0.95      0.99      0.97       549
         1.0       0.92      0.77      0.84       111

    accuracy                           0.95       660
   macro avg       0.94      0.88      0.90       660
weighted avg       0.95      0.95      0.95       660

[[542   7]
```

```
 [ 26  85]]
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='ovr', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)Accuracy: 0.81 (+/- 0.28)
Num Features: 16
Selected Features: [False False False False False False False False False False
False False
  True False False False False False False False False False False False
 False False  True False False False False False False False False False
 False False False False  True False  True False False False False False
 False False False False False False False False  True  True False False
 False False False False False False  True False False False False  True
 False False False False False False False False False False False False
 False False  True False False False False False False False False False
 False False  True False False False False False False False False False
 False False False False False False False False False False  True False
 False False False False False False False False False False False False
 False False False False False  True False False False False False False
 False False False False  True False False False False False False False
 False False False  True False  True  True False False False]
Feature Ranking: [ 16  48  14  73 145  60 146  95  13  10  71  35   1  62  63
65   4 122
 142 140 138  12 128 137  67  88   1 101  55 125  49  43  17 130  25  39
 100  19  38  31   1  41   1 143  85 147  46  78  94  21  80   6  76 149
 148  91   1   1  57 106 117 123 114 108  27  18   1 102 141  32  15   1
  70  84  87  82 124  64  93 139  75 121  81 116   9  45   1  92  97  59
 134  51  54  69 151   2 133   5   1 103  40  52  74  86  61 127  83 104
 111 112 110 129  77  29 105 107  72  90   1  11  66  56  58  98   3  50
   8 126   7  26  28 119  79  36  96  20  47   1  33  34 109 150  68  23
 144 120  37 118   1  30  24  53 136  99  42  44 135 113 131   1 132   1
   1  22  89 115]
```

2.8.4 Support Vector Machines (SVM)

Let us evaluate the Support Vector Machines algorithm on our first data set.

[19]:
```
model(svm,X_train,y_train,X_test,y_test)
crossval(svm,X,y)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)Accuracy: 88.63636363636364
              precision    recall  f1-score   support

         0.0       0.88      1.00      0.94       549
```

```
         1.0       1.00      0.32      0.49       111

    accuracy                           0.89       660
   macro avg       0.94      0.66      0.71       660
weighted avg       0.90      0.89      0.86       660

[[549   0]
 [ 75  36]]
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)Accuracy: 0.85 (+/- 0.01)
```

2.8.5 XGBoost

Let us evaluate the XGBoost algorithm on our first data set.

[20]:
```
model(xgb,X_train,y_train,X_test,y_test)
crossval(xgb,X,y)
rfe(xgb,X,y)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=0, num_parallel_tree=1,
              objective='binary:logistic', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1,
verbosity=None)Accuracy: 99.0909090909091
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99       549
         1.0       0.99      0.95      0.97       111

    accuracy                           0.99       660
   macro avg       0.99      0.98      0.98       660
weighted avg       0.99      0.99      0.99       660

[[548   1]
 [  5 106]]
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=0, num_parallel_tree=1,
```

```
              objective='binary:logistic', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1,
verbosity=None)Accuracy: 0.86 (+/- 0.28)
Num Features: 16
Selected Features: [ True False False False False False False False  True False
False False
 False False False False False False False False False  True False False
 False False False False False False False  True False False  True  True
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False  True False False  True False False False False False False
 False False False False False False False False False False False False
 False False False False False False False  True False False False False
 False False False False False False False False False False False False
 False  True False False False False False False False False False False
 False False False  True False  True False False False False False  True
 False False False False False False False False False False False False
 False False False False False False  True False False False False False
 False False False False False False  True False  True False]
Feature Ranking: [  1  47  51  66  50  49  92 125   1   3 116 113 102  99  87
95   6  70
  80 121  10   1  75 109  53 150  31  81  77  63  18   1  67  40   1   1
 111   4 119  38 148  15  29  74 127 136 128  82  94  16  62  21  84  73
   8  46 103  48  36 142  34  11   1 132 135   1  27  39  90  93 147 122
 120 104  97 105   9 139 133  20  85 112  26  98 144  43 151  35  23 118
  56   1  19 107 114  54 134  88  78 124  24  37  57  89  65  79 106 110
  55   1  32 108 117  33  52  41  44  71  91  72  64  14  58   1  83   1
 140 126 129  69   5   1  17 123 115  42  28  61 130   2  59 145  96  45
  25  13 146  60 143 149   1 141  12  76  30   7 138 101 100  86 131  22
   1  68   1 137]
```

2.8.6 Gaussian Naive Bayes

Let us evaluate the Gaussian Naive Bayes algorithm on our first data set.

[13]:
```
model(gnb,X_train,y_train,X_test,y_test)
crossval(gnb,X,y)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)Accuracy: 83.03030303030303
              precision    recall  f1-score   support

         0.0       0.95      0.84      0.89       560
         1.0       0.46      0.76      0.58       100

    accuracy                           0.83       660
   macro avg       0.71      0.80      0.73       660
weighted avg       0.88      0.83      0.85       660
```

```
[[472  88]
 [ 24  76]]
GaussianNB(priors=None, var_smoothing=1e-09)Accuracy: 0.80 (+/- 0.31)
```

2.9 Experiments on Our Second Data Set

Our second dataset is slightly different from the first one. The training and test sets consist of two different csv blocks, so we define these two datasets separately.

[8]:
```
database2 = pd.read_csv("shuttle-train.csv", sep=';')
database2.head()
```

[8]:

	f1	f2	f3	f4	f5	f6	f7	f8	f9	d
0	50	21	77	0	28	0	27	48	22	2
1	55	0	92	0	0	26	36	92	56	4
2	53	0	82	0	52	-5	29	30	2	1
3	37	0	76	0	28	18	40	48	8	1
4	37	0	79	0	34	-26	43	46	2	1

[9]:
```
database3 = pd.read_csv("shuttle-test.csv", sep=';')
database3.head()
```

[9]:

	f1	f2	f3	f4	f5	f6	f7	f8	f9	d
0	55	0	81	0	-6	11	25	88	64	4
1	56	0	96	0	52	-4	40	44	4	4
2	50	-1	89	-7	50	0	39	40	2	1
3	53	9	79	0	42	-2	25	37	12	4
4	55	2	82	0	54	-6	26	28	2	1

2.9.1 Train and Test Split

At this stage, we already have separate test and train subsets, and as a result, we do not need to use the `train_test_split` function we uses for the first dataset. Because currently our training and test sets are already given. All we have to do is to assign the independent and dependent variables.

[10]:
```
X1_train = database2.drop(["d"], axis=1)
X1_test = database3.drop(["d"], axis=1)
y1_train = database2['d']
y1_test = database3['d']
```

2.9.2 Combining Train and Test for Cross-Validation

This stage is actually combining the operations we did in stage 9, the two datasets above and then assigning the properties as df_row_reindex_X and the results as df_row_reindex_y. As a result, it allows us to work with a single dataset, not two. We do this with the function named concat in the pandas library.

[11]:
```
df_row_reindex = pd.concat([database2, database3], ignore_index=True)
df_row_reindex_X = df_row_reindex.drop(["d"], axis=1)
df_row_reindex_y = df_row_reindex['d']
```

2.9.3 Decision Trees

Let us evaluate the Decision Tree algorithm on our second data set.

[13]:
```
model(dt,X1_train,y1_train,X1_test,y1_test)
crossval(dt,df_row_reindex_X,df_row_reindex_y)
rfe(dt,df_row_reindex_X,df_row_reindex_y)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')Accuracy:
99.99310344827586
              precision    recall  f1-score   support

           1       1.00      1.00      1.00     11478
           2       1.00      0.92      0.96        13
           3       1.00      1.00      1.00        39
           4       1.00      1.00      1.00      2155
           5       1.00      1.00      1.00       809
           6       1.00      1.00      1.00         4
           7       1.00      1.00      1.00         2

    accuracy                           1.00     14500
   macro avg       1.00      0.99      0.99     14500
weighted avg       1.00      1.00      1.00     14500

[[11478     0     0     0     0     0     0]
 [    0    12     0     1     0     0     0]
 [    0     0    39     0     0     0     0]
 [    0     0     0  2155     0     0     0]
 [    0     0     0     0   809     0     0]
 [    0     0     0     0     0     4     0]
 [    0     0     0     0     0     0     2]]
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')Accuracy: 1.00 (+/-
0.00)
Num Features: 9
```

```
Selected Features: [ True  True  True  True  True  True  True  True  True]
Feature Ranking: [1 1 1 1 1 1 1 1 1]
```

2.9.4 Random Forest

Let us evaluate the Random Forest algorithm on our second data set.

[14]:
```
model(rf,X1_train,y1_train,X1_test,y1_test)
crossval(rf,df_row_reindex_X,df_row_reindex_y)
rfe(rf,df_row_reindex_X,df_row_reindex_y)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=-1, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)Accuracy: 99.97931034482758
              precision    recall  f1-score   support

           1       1.00      1.00      1.00     11478
           2       1.00      0.92      0.96        13
           3       0.97      1.00      0.99        39
           4       1.00      1.00      1.00      2155
           5       1.00      1.00      1.00       809
           6       1.00      0.75      0.86         4
           7       1.00      0.50      0.67         2

    accuracy                           1.00     14500
   macro avg       1.00      0.88      0.92     14500
weighted avg       1.00      1.00      1.00     14500

[[11478     0     0     0     0     0     0]
 [    0    12     0     1     0     0     0]
 [    0     0    39     0     0     0     0]
 [    0     0     0  2155     0     0     0]
 [    0     0     0     0   809     0     0]
 [    0     0     0     1     0     3     0]
 [    0     0     1     0     0     0     1]]
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=-1, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)Accuracy: 1.00 (+/- 0.00)
```

```
Num Features: 9
Selected Features: [ True  True  True  True  True  True  True  True  True]
Feature Ranking: [1 1 1 1 1 1 1 1 1]
```

2.9.5 Logistic Regression

Let us evaluate the Logistic Regression algorithm on our second data set.

[15]:
```
model(logreg, X1_train,y1_train,X1_test,y1_test)
crossval(logreg,df_row_reindex_X,df_row_reindex_y)
rfe(logreg,df_row_reindex_X,df_row_reindex_y)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='ovr', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)Accuracy: 93.10344827586206
              precision    recall  f1-score   support

           1       0.93      0.99      0.96     11478
           2       0.00      0.00      0.00        13
           3       0.00      0.00      0.00        39
           4       0.91      0.61      0.73      2155
           5       1.00      1.00      1.00       809
           6       0.00      0.00      0.00         4
           7       0.00      0.00      0.00         2

    accuracy                           0.93     14500
   macro avg       0.41      0.37      0.38     14500
weighted avg       0.93      0.93      0.92     14500

[[11372     0     0   104     0     0     2]
 [    8     0     0     5     0     0     0]
 [   17     0     0    22     0     0     0]
 [  835     0     0  1320     0     0     0]
 [    1     0     0     0   808     0     0]
 [    0     0     0     4     0     0     0]
 [    2     0     0     0     0     0     0]]

/home/kaygun/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1268: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/kaygun/.local/lib/python3.8/site-packages/sklearn/svm/_base.py:946:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='ovr', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)Accuracy: 0.93 (+/- 0.01)
Num Features: 9
Selected Features: [ True  True  True  True  True  True  True  True  True]
Feature Ranking: [1 1 1 1 1 1 1 1 1]
```

2.9.6 Support Vector Machines

Let us evaluate the Support Vector Machines algorithm on our second data set. (My machine was unable to perform this operation.)

[ ]:
```
model(svm,X1_train,y1_train,X1_test,y1_test)
crossval(svm,df_row_reindex_X,df_row_reindex_y)
```

2.9.7 XGBoost

Let us evaluate the XGBoost algorithm on our second data set.

[16]:
```
model(xgb,X1_train,y1_train,X1_test,y1_test)
crossval(xgb,df_row_reindex_X,df_row_reindex_y)
rfe(xgb,df_row_reindex_X,df_row_reindex_y)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=0, num_parallel_tree=1,
              objective='multi:softprob', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=None, subsample=1,
              tree_method='exact', validate_parameters=1,
verbosity=None)Accuracy: 99.99310344827586
              precision    recall  f1-score   support

           1       1.00      1.00      1.00     11478
           2       1.00      0.92      0.96        13
           3       1.00      1.00      1.00        39
           4       1.00      1.00      1.00      2155
           5       1.00      1.00      1.00       809
           6       1.00      1.00      1.00         4
           7       1.00      1.00      1.00         2

    accuracy                           1.00     14500
   macro avg       1.00      0.99      0.99     14500
weighted avg       1.00      1.00      1.00     14500
```

```
[[11478     0     0     0     0     0     0]
 [    0    12     0     1     0     0     0]
 [    0     0    39     0     0     0     0]
 [    0     0     0  2155     0     0     0]
 [    0     0     0     0   809     0     0]
 [    0     0     0     0     0     4     0]
 [    0     0     0     0     0     0     2]]
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=0, num_parallel_tree=1,
              objective='multi:softprob', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=None, subsample=1,
              tree_method='exact', validate_parameters=1,
verbosity=None)Accuracy: 1.00 (+/- 0.00)
Num Features: 9
Selected Features: [ True  True  True  True  True  True  True  True  True]
Feature Ranking: [1 1 1 1 1 1 1 1 1]
```

2.9.8 Gaussian Naive Bayes

Let us evaluate the Gaussian Naive Bayes algorithm on our second data set.

[17]:
```
model(gnb,X1_train,y1_train,X1_test,y1_test)
crossval(gnb,df_row_reindex_X,df_row_reindex_y)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)Accuracy: 82.6551724137931
              precision    recall  f1-score   support

           1       0.95      0.88      0.92     11478
           2       0.01      0.92      0.02        13
           3       0.11      0.59      0.19        39
           4       0.89      0.54      0.67      2155
           5       0.99      0.82      0.90       809
           6       0.40      1.00      0.57         4
           7       0.00      1.00      0.01         2

    accuracy                           0.83     14500
   macro avg       0.48      0.82      0.47     14500
weighted avg       0.94      0.83      0.88     14500

[[10116   463   185   143     4     5   562]
 [    1    12     0     0     0     0     0]
 [    7     1    23     0     1     0     7]
 [  502   491     0  1162     0     0     0]
```

```
 [     0   142     0     0   666     1     0]
 [     0     0     0     0     0     4     0]
 [     0     0     0     0     0     0     2]]
GaussianNB(priors=None, var_smoothing=1e-09)Accuracy: 0.81 (+/- 0.01)
```

2.10 Experiments on Our Third Data Set

[20]:
```
db = pd.read_csv("data-hastalik.csv",sep = ';' )
db.head()
```

[20]:
```
   ATES  BULANTI  BEL-AGRI  SUREKLI-WC  IDRAR-SIRASINDA-AGRI  \
0   355        0         1           0                     0
1   359        0         0           1                     1
2   359        0         1           0                     0
3   360        0         0           1                     1
4   360        0         1           0                     0

   URETRADA-YANMA-SISME-KASINTI  MESANE-ILTIHABI  BOBREK-ILTIHABI
0                             0                0                0
1                             1                1                0
2                             0                0                0
3                             1                1                0
4                             0                0                0
```

2.10.1 Feature selection and splitting the train and test data sets

Let us denote the features we are going to use with X and response variable with y. With the drop function, we get the columns other than the columns ‘MESANE-ILTIHABI’ and ‘BOBREK-ILTIHABI’. For the response variable, we just take the column named ‘MESANE-ILTIHABI’. Then we determine the size of our training and test sets. For this dataset, 80% of all data is used for training and the remaining 20% is used for testing using the function “train_test_split”.

[50]:
```
X = db.drop(["MESANE-ILTIHABI","BOBREK-ILTIHABI"],axis=1)
y = db['MESANE-ILTIHABI']
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.2 )
```

[51]:
```
A = db.drop(["MESANE-ILTIHABI","BOBREK-ILTIHABI"],axis=1)
b = db['BOBREK-ILTIHABI']
A_train, A_test, b_train, b_test = train_test_split(A, b,test_size=0.2 )
```

2.10.2 Create array for loop

We create two arrays for our third dataset. We will use these arrays for the functions we will write below. The difference between the two arrays is that the rfe function does not work in SVM and GNB algorithms.

[ ]:
```
models = [dt, rf, logreg, svm, xgb, gnb]
```

2.10.3 Evaluate Models

Let us evaluate the all model on our third data set.

[85]:
```
for i in models:
    model(i, X_train, X_test, y_train, y_test)
    model(i, A_train, A_test, b_train, b_test)
    crossval(i, X, y, 5)
    crossval(i, X, y, 10)
    crossval(i, A, b, 5)
    crossval(i, A, b, 10)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')Accuracy: 100.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        10

    accuracy                           1.00        24
   macro avg       1.00      1.00      1.00        24
weighted avg       1.00      1.00      1.00        24

[[14  0]
 [ 0 10]]
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')Accuracy: 100.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      1.00      1.00         9

    accuracy                           1.00        24
   macro avg       1.00      1.00      1.00        24
weighted avg       1.00      1.00      1.00        24

[[15  0]
 [ 0  9]]
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
```

```
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')Accuracy: 0.97 (+/-
0.13)
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')Accuracy: 1.00 (+/-
0.00)
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')Accuracy: 0.86 (+/-
0.36)
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')Accuracy: 0.97 (+/-
0.20)
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=-1, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)Accuracy: 100.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        10

    accuracy                           1.00        24
   macro avg       1.00      1.00      1.00        24
weighted avg       1.00      1.00      1.00        24

[[14  0]
 [ 0 10]]
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
```

```
                         min_impurity_decrease=0.0, min_impurity_split=None,
                         min_samples_leaf=1, min_samples_split=2,
                         min_weight_fraction_leaf=0.0, n_estimators=100,
                         n_jobs=-1, oob_score=False, random_state=0, verbose=0,
                         warm_start=False)Accuracy: 100.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      1.00      1.00         9

    accuracy                           1.00        24
   macro avg       1.00      1.00      1.00        24
weighted avg       1.00      1.00      1.00        24

[[15  0]
 [ 0  9]]
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                         criterion='gini', max_depth=None, max_features='auto',
                         max_leaf_nodes=None, max_samples=None,
                         min_impurity_decrease=0.0, min_impurity_split=None,
                         min_samples_leaf=1, min_samples_split=2,
                         min_weight_fraction_leaf=0.0, n_estimators=100,
                         n_jobs=-1, oob_score=False, random_state=0, verbose=0,
                         warm_start=False)Accuracy: 1.00 (+/- 0.00)
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                         criterion='gini', max_depth=None, max_features='auto',
                         max_leaf_nodes=None, max_samples=None,
                         min_impurity_decrease=0.0, min_impurity_split=None,
                         min_samples_leaf=1, min_samples_split=2,
                         min_weight_fraction_leaf=0.0, n_estimators=100,
                         n_jobs=-1, oob_score=False, random_state=0, verbose=0,
                         warm_start=False)Accuracy: 1.00 (+/- 0.00)
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                         criterion='gini', max_depth=None, max_features='auto',
                         max_leaf_nodes=None, max_samples=None,
                         min_impurity_decrease=0.0, min_impurity_split=None,
                         min_samples_leaf=1, min_samples_split=2,
                         min_weight_fraction_leaf=0.0, n_estimators=100,
                         n_jobs=-1, oob_score=False, random_state=0, verbose=0,
                         warm_start=False)Accuracy: 1.00 (+/- 0.00)
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                         criterion='gini', max_depth=None, max_features='auto',
                         max_leaf_nodes=None, max_samples=None,
                         min_impurity_decrease=0.0, min_impurity_split=None,
                         min_samples_leaf=1, min_samples_split=2,
                         min_weight_fraction_leaf=0.0, n_estimators=100,
                         n_jobs=-1, oob_score=False, random_state=0, verbose=0,
                         warm_start=False)Accuracy: 0.97 (+/- 0.20)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='ovr', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)Accuracy: 100.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        10

    accuracy                           1.00        24
   macro avg       1.00      1.00      1.00        24
weighted avg       1.00      1.00      1.00        24

[[14  0]
 [ 0 10]]
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='ovr', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)Accuracy: 100.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      1.00      1.00         9

    accuracy                           1.00        24
   macro avg       1.00      1.00      1.00        24
weighted avg       1.00      1.00      1.00        24

[[15  0]
 [ 0  9]]
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='ovr', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)Accuracy: 1.00 (+/- 0.00)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='ovr', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)Accuracy: 1.00 (+/- 0.00)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='ovr', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)Accuracy: 1.00 (+/- 0.00)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
```

```
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='ovr', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)Accuracy: 1.00 (+/- 0.00)
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)Accuracy: 87.5
              precision    recall  f1-score   support

           0       1.00      0.79      0.88        14
           1       0.77      1.00      0.87        10

    accuracy                           0.88        24
   macro avg       0.88      0.89      0.87        24
weighted avg       0.90      0.88      0.88        24

[[11  3]
 [ 0 10]]
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)Accuracy: 91.66666666666666
              precision    recall  f1-score   support

           0       1.00      0.87      0.93        15
           1       0.82      1.00      0.90         9

    accuracy                           0.92        24
   macro avg       0.91      0.93      0.91        24
weighted avg       0.93      0.92      0.92        24

[[13  2]
 [ 0  9]]
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)Accuracy: 0.54 (+/- 0.17)
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)Accuracy: 0.63 (+/- 0.33)
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)Accuracy: 0.74 (+/- 0.33)
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
```

```
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)Accuracy: 0.84 (+/- 0.39)
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)Accuracy: 100.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        10

    accuracy                           1.00        24
   macro avg       1.00      1.00      1.00        24
weighted avg       1.00      1.00      1.00        24

[[14  0]
 [ 0 10]]
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)Accuracy: 100.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      1.00      1.00         9

    accuracy                           1.00        24
   macro avg       1.00      1.00      1.00        24
weighted avg       1.00      1.00      1.00        24

[[15  0]
 [ 0  9]]
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)Accuracy: 0.97 (+/- 0.10)
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
```

```
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)Accuracy: 0.97 (+/- 0.15)
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)Accuracy: 0.86 (+/- 0.36)
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)Accuracy: 0.97 (+/- 0.15)
GaussianNB(priors=None, var_smoothing=1e-09)Accuracy: 70.83333333333334
              precision    recall  f1-score   support

           0       1.00      0.50      0.67        14
           1       0.59      1.00      0.74        10

    accuracy                           0.71        24
   macro avg       0.79      0.75      0.70        24
weighted avg       0.83      0.71      0.70        24

[[ 7  7]
 [ 0 10]]
GaussianNB(priors=None, var_smoothing=1e-09)Accuracy: 91.66666666666666
              precision    recall  f1-score   support

           0       0.88      1.00      0.94        15
           1       1.00      0.78      0.88         9

    accuracy                           0.92        24
   macro avg       0.94      0.89      0.91        24
weighted avg       0.93      0.92      0.91        24

[[15  0]
 [ 2  7]]
GaussianNB(priors=None, var_smoothing=1e-09)Accuracy: 0.82 (+/- 0.19)
GaussianNB(priors=None, var_smoothing=1e-09)Accuracy: 0.82 (+/- 0.32)
GaussianNB(priors=None, var_smoothing=1e-09)Accuracy: 0.92 (+/- 0.33)
GaussianNB(priors=None, var_smoothing=1e-09)Accuracy: 0.95 (+/- 0.25)
```

2.10.4 Recursive Feature Elimination

Evaluate rfe on our third data set

[67]:
```
model = [dt, rf, logreg, xgb]
for i in model:
    print(i)
    rfe(i, X, y)
    rfe(i, A, b)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
Num Features: 2
Selected Features: [False False False  True  True False]
Feature Ranking: [2 5 4 1 1 3]
Num Features: 2
Selected Features: [ True False  True False False False]
Feature Ranking: [1 5 1 4 3 2]
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=-1, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
Num Features: 2
Selected Features: [False False False  True  True False]
Feature Ranking: [3 5 2 1 1 4]
Num Features: 2
Selected Features: [ True False  True False False False]
Feature Ranking: [1 2 1 5 4 3]
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='ovr', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
Num Features: 2
Selected Features: [False False False  True  True False]
Feature Ranking: [5 3 2 1 1 4]
Num Features: 2
Selected Features: [False  True  True False False False]
Feature Ranking: [5 1 1 3 4 2]
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
```

```
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
Num Features: 2
Selected Features: [False False False  True  True False]
Feature Ranking: [2 3 4 1 1 5]
Num Features: 2
Selected Features: [ True False  True False False False]
Feature Ranking: [1 5 1 2 3 4]
```