SVC (Support Vector Classifier)

Row and columns

```
2]: x.shape
```

```
2]: (399, 27)
```

Data preprocessing

```
21]: classification_yes
     True     249
     False    150
     Name: count, dtype: int64
```

```
11]: from sklearn.model_selection import train_test_split
     x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=0)
```

```
12]: from sklearn.preprocessing import StandardScaler
     sc=StandardScaler()
     x_train=sc.fit_transform(x_train)
     x_test =sc.transform(x_test)
```

Model

```
from sklearn.model_selection import GridSearchCV
param_grid={'kernel':['linear','rbf','sigmoid','poly'],
            'gamma':['auto','scale'],
            'C':[10,100,1000,2000,3000]}
grid=GridSearchCV(SVC(),param_grid,refit=True,verbose=3,n_jobs=-1,scoring='f1_weighted')
grid.fit(x_train,y_train)
```

CLF report

```
16]: clf
```

```
16]: '              precision    recall  f1-score   support\n\n       False       0.96      1.00      0.98        45\n        True       1.00      0.97
     0.99        75\n\n    accuracy                           0.98       120\n   macro avg       0.98      0.99      0.98       120\nweighted avg       0.98
     0.98      0.98       120\n'
```

```
17]: cm
```

```
17]: array([[45,  0],
            [ 2, 73]], dtype=int64)
```

F1 score

```
]: from sklearn.metrics import f1_score
   f1_macro=f1_score(y_test,y_pred,average='weighted')
   print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'C': 10, 'gamma': 'auto', 'kernel': 'sigmoid'}: 0.9834018801410106
```

Decision_tree

Row and columns

```
x.shape
```

```
(399, 27)
```

Data preprocessing

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=0)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test =sc.transform(x_test)
```

```
data['classification_yes'].value_counts()
```

```
classification_yes
True     249
False    150
Name: count, dtype: int64
```

Model

```
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')
grid.fit(x_test,y_test)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
```

CLF report

```
clf
```

```
'              precision    recall  f1-score   support\n\n       False       1.00      1.00      1.00        45\n        True       1.00      1.00
1.00        75\n\n    accuracy                           1.00       120\n   macro avg       1.00      1.00      1.00       120\nweighted avg       1.00
1.00      1.00       120\n'
```

```
cm
```

```
array([[45,  0],
       [ 0, 75]], dtype=int64)
```

F1 score

```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_prediction,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'criterion': 'entropy', 'max_features': 'sqrt', 'splitter': 'random'}: 1.0
```

Random_forest

Row and columns

```
2]: x.shape
```

```
2]: (399, 27)
```

Data preprocessing

```
]: from sklearn.model_selection import train_test_split
   x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=0)
```

```
]: from sklearn.preprocessing import StandardScaler
   sc=StandardScaler()
   x_train=sc.fit_transform(x_train)
   x_test =sc.transform(x_test)
```

Model

```
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)
```

CLF report

```
clf
```

```
'              precision    recall  f1-score   support\n\n       False       1.00      1.00      1.00        45\n        True       1.00      1.00
  1.00        75\n\n    accuracy                           1.00       120\n   macro avg       1.00      1.00      1.00       120\nweighted avg       1.00
  1.00      1.00       120\n'
```

```
cm
```

```
array([[45,  0],
       [ 0, 75]], dtype=int64)
```

F1 score

```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_prediction,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'criterion': 'gini', 'max_features': 'log2', 'n_estimators': 100}: 1.0
```

Logistic_ regression

Row and columns

```
2]: x.shape
```

```
2]: (399, 27)
```

Data preprocessing

```
]: from sklearn.model_selection import train_test_split
   x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=0)
```

```
]: from sklearn.preprocessing import StandardScaler
   sc=StandardScaler()
   x_train=sc.fit_transform(x_train)
   x_test =sc.transform(x_test)
```

Model

```
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')
grid.fit(x_test,y_test)
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
```

CLF report

```
clf
```

```
'              precision    recall  f1-score   support\n\n       False       1.00      1.00      1.00        45\n        True       1.00      1.00
1.00        75\n\n    accuracy                           1.00       120\n   macro avg       1.00      1.00      1.00       120\nweighted avg       1.00
1.00      1.00       120\n'
```

F1 score

```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_prediction,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'penalty': 'l2', 'solver': 'newton-cg'}: 1.0
```

KNN(KNeighborsClassifier)

Row and columns

```
2]: x.shape

2]: (399, 27)
```

Data preprocessing

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=0)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test =sc.transform(x_test)
```

Model

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 7, metric = 'minkowski', p = 2)
classifier.fit(x_train, y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\neighbors\_classification.py:238: 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().
  return self._fit(X, y)
```

KNeighborsClassifier

KNeighborsClassifier(n_neighbors=7)

CLF report

```
clf
```

```
'              precision    recall  f1-score   support\n\n       False       0.86      1.00      0.93        51\n        True       1.00      0.90      0.95        82\n\n    accuracy                           0.94       133\n   macro avg       0.93      0.95      0.94       133\nweighted avg       0.95      0.94      0.94       133\n'
```

```
cm
```

```
array([[51,  0],
       [ 8, 74]], dtype=int64)
```

Naive baye_s

1) MultinomialNB

Row and columns

```
2]: x.shape

2]: (399, 27)
```

Data preprocessing

```
: #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(x, y, test_size = 1/3, random_state = 0)
```

```
: from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test =sc.transform(x_test)
```

Model

```
: from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
classifier.fit(X_train, y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1339: DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
: MultinomialNB
MultinomialNB()
```

```
: y_pred = classifier.predict(X_test)
```

CLF report

```
]: clf

]: '              precision    recall  f1-score   support\n\n       False       0.68      0.98      0.81        51\n        True       0.98      0.72
   0.83        82\n\n    accuracy                           0.82       133\n   macro avg       0.83      0.85      0.82       133\nweighted avg       0.87
   0.82      0.82       133\n'
```

```
]: cm

]: array([[50,  1],
       [23, 59]], dtype=int64)
```

F1 SCORE

```
       [23, 59]], dtype=int64)
```

[18]:
```
from sklearn.metrics import f1_score


f1_marco=f1_score(y_test,y_pred,average='weighted')
f1_marco
```

[18]:
```
0.8215780250262184
```

2) BernoulliNB

Row and columns

2]:
```
x.shape
```

2]:
```
(399, 27)
```

Data preprocessing

```
#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(x, y, test_size = 1/3, random_state = 0)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test =sc.transform(x_test)
```

Model

5]:
```
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)
```

CLF report

```
              precision    recall  f1-score   support

       False       0.86      1.00      0.93        51
        True       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]]
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1339: DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

f1score_

```
[ ]
```

```
[21]: from sklearn.metrics import f1_score


      f1_marco=f1_score(y_test,y_pred,average='weighted')
      f1_marco
```

```
[21]: 0.8215780250262184
```

3) ComplementNB

Row and columns

```
2]: x.shape
```

```
2]: (399, 27)
```

Data preprocessing

```
#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(x, y, test_size = 1/3, random_state = 0)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test =sc.transform(x_test)
```

Model

```
#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(x, y, test_size = 1/3, random_state = 0)
```

```
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)
```

CLF report

```
              precision    recall  f1-score   support

       False       0.68      0.98      0.81        51
        True       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]]
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1339: DataConversionWarning: A column-vector y was passed when a 1d array w
cted. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

F1_SCORE

```
cm
```

```
array([[50,  1],
       [23, 59]], dtype=int64)
```

```
from sklearn.metrics import f1_score


f1_marco=f1_score(y_test,y_pred,average='weighted')
f1_marco
```

```
0.8215780250262184
```