```
import pandas as pd
df=pd.read_csv("/content/survey lung cancer.csv")
df
```

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING
0	M	69	1	2	2	1	1	2	1	2	2	2
1	M	74	2	1	1	1	2	2	2	1	1	1
2	F	59	1	1	1	2	1	2	1	2	1	2
3	M	63	2	2	2	1	1	1	1	1	2	1
4	F	63	1	2	1	1	1	1	1	2	1	2
...	...	...	...	...	...	...	...	...	...	...	...	...
304	F	56	1	1	1	2	2	2	1	1	2	2
305	M	70	2	1	1	1	1	2	2	2	2	2
306	M	58	2	1	1	1	1	1	2	2	2	2
307	M	67	2	1	2	1	1	2	2	1	2	2
308	M	62	1	1	1	2	1	2	2	2	2	1

309 rows × 16 columns

```
#Selected Features:
#Index(['AGE', 'YELLOW_FINGERS', 'PEER_PRESSURE', 'CHRONIC DISEASE', 'FATIGUE ','ALLERGY ', 'WHEEZING', 'ALCOHOL CONSUMING', 'COUGHING','SWAL
```

```
df.head()
```

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLER
0	M	69	1	2	2	1	1	2	
1	M	74	2	1	1	1	2	2	
2	F	59	1	1	1	2	1	2	
3	M	63	2	2	2	1	1	1	

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
df['GENDER']= label_encoder.fit_transform(df['GENDER'])
df['LUNG_CANCER']= label_encoder.fit_transform(df['LUNG_CANCER'])
```

```
df.head()
```

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLER
0	1	69	1	2	2	1	1	2	
1	1	74	2	1	1	1	2	2	
2	0	59	1	1	1	2	1	2	
3	1	63	2	2	2	1	1	1	

```
df.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
GENDER	309.0	0.524272	0.500221	0.0	0.0	1.0	1.0	1.0
AGE	309.0	62.673139	8.210301	21.0	57.0	62.0	69.0	87.0
SMOKING	309.0	1.563107	0.496806	1.0	1.0	2.0	2.0	2.0
YELLOW_FINGERS	309.0	1.569579	0.495938	1.0	1.0	2.0	2.0	2.0
ANXIETY	309.0	1.498382	0.500808	1.0	1.0	1.0	2.0	2.0
PEER_PRESSURE	309.0	1.501618	0.500808	1.0	1.0	2.0	2.0	2.0
CHRONIC DISEASE	309.0	1.504854	0.500787	1.0	1.0	2.0	2.0	2.0
FATIGUE	309.0	1.673139	0.469827	1.0	1.0	2.0	2.0	2.0
ALLERGY	309.0	1.556634	0.497588	1.0	1.0	2.0	2.0	2.0
WHEEZING	309.0	1.556634	0.497588	1.0	1.0	2.0	2.0	2.0
ALCOHOL CONSUMING	309.0	1.556634	0.497588	1.0	1.0	2.0	2.0	2.0

```
x=df.iloc[:,1:11]
y=df['LUNG_CANCER']
y
```

```
0      1
1      1
2      0
3      0
4      0
      ..
304    1
305    1
306    1
307    1
308    1
Name: LUNG_CANCER, Length: 309, dtype: int64
```

```
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=56)
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x2_train = scaler.fit_transform(x_train)
x2_test= scaler.transform(x_test)
```

```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
y_pred=dt.predict(x_test)
y_pred
```

```
array([0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1])
```

```
from sklearn.metrics import mean_squared_error,r2_score,mean_absolute_error,accuracy_score
```

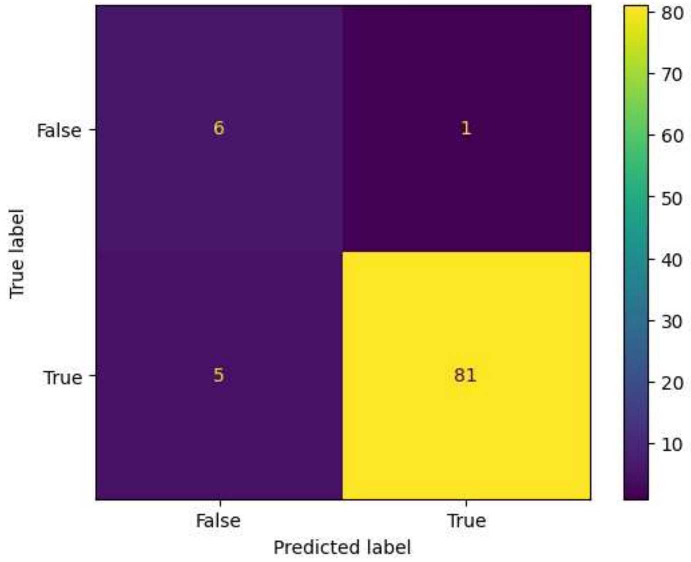
```
print("Mean_Squared_Error :",mean_squared_error(y_test,y_pred))
print("R2_score :",r2_score(y_test,y_pred))
print("Mean_Absolute_Error :",mean_absolute_error(y_test,y_pred))
```

```
Mean_Squared_Error : 0.06451612903225806
R2_score : 0.07308970099667778
Mean_Absolute_Error : 0.06451612903225806
```

```
from sklearn.metrics import confusion_matrix,classification_report
cm = confusion_matrix(y_test,y_pred)
cr= classification_report(y_test,y_pred)
a=accuracy_score(y_test, y_pred)
print(cm)
print(cr)
print(a)
```

```
[[ 6  1]
 [ 5 81]]
              precision    recall  f1-score   support

           0       0.55      0.86      0.67         7
           1       0.99      0.94      0.96        86

    accuracy                           0.94        93
   macro avg       0.77      0.90      0.82        93
weighted avg       0.95      0.94      0.94        93

0.9354838709677419
```

```
import matplotlib.pyplot as plt
from sklearn import metrics
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = [False, True])
cm_display.plot()
plt.show()
```



```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None,min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf
rf.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'` has been depr
  warn(
```

```
▾                          RandomForestClassifier
RandomForestClassifier(max_features='auto', n_estimators=10, n_jobs=1)
```

```
y_pred2=rf.predict(x_test)
y_pred2
```

```
array([1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1])
```

```
from sklearn.feature_selection import SelectFromModel
feature_importances = rf.feature_importances_
selector = SelectFromModel(rf, threshold=0.05)  # You can adjust the threshold as needed

# Apply feature selection
```

```
selector.fit(x_train, y_train)
selected_features = x.columns[selector.get_support()]

# Print the selected features
print("Selected Features:")
print(selected_features)
```

```
Selected Features:
Index(['AGE', 'YELLOW_FINGERS', 'ANXIETY', 'PEER_PRESSURE', 'CHRONIC DISEASE',
       'FATIGUE ', 'ALLERGY ', 'WHEEZING', 'ALCOHOL CONSUMING'],
      dtype='object')
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'` has been deprecated in 1.1
  warn(
```

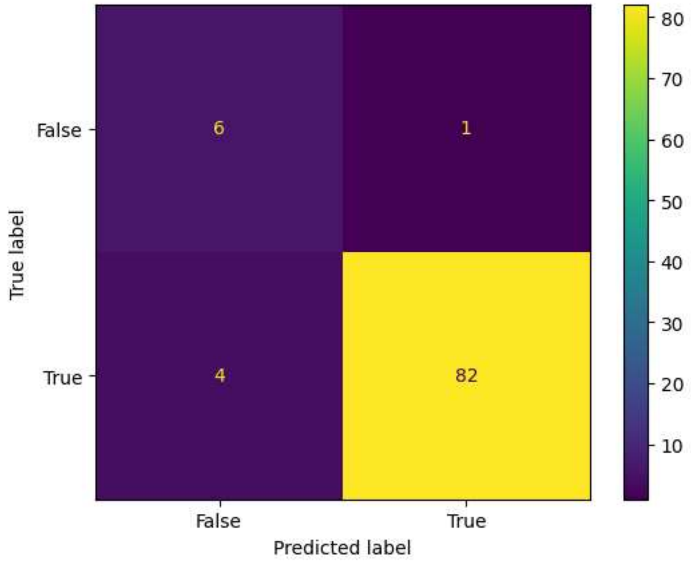
```
print("Mean_Squared_Error :",mean_squared_error(y_test,y_pred2))
print("R2_score :",r2_score(y_test,y_pred2))
print("Mean_Absolute_Error :",mean_absolute_error(y_test,y_pred2))
```

```
Mean_Squared_Error : 0.053763440860215055
R2_score : 0.2275747508305649
Mean_Absolute_Error : 0.053763440860215055
```

```
cm2 = confusion_matrix(y_test,y_pred2)
cr2= classification_report(y_test,y_pred2)
print(cm2)
print(cr2)
a2=accuracy_score(y_test, y_pred2)
print(a2)
```

```
[[ 6  1]
 [ 4 82]]
              precision    recall  f1-score   support

           0       0.60      0.86      0.71         7
           1       0.99      0.95      0.97        86

    accuracy                           0.95        93
   macro avg       0.79      0.91      0.84        93
weighted avg       0.96      0.95      0.95        93

0.946236559139785
```

```
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm2, display_labels = [False, True])
cm_display.plot()
plt.show()
```



```
from sklearn.naive_bayes import GaussianNB
gb = GaussianNB()
gb.fit(x_train, y_train)
```

```
▾ GaussianNB
GaussianNB()
```

```
y_pred3=gb.predict(x_test)
y_pred3
```

```
array([1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1])
```

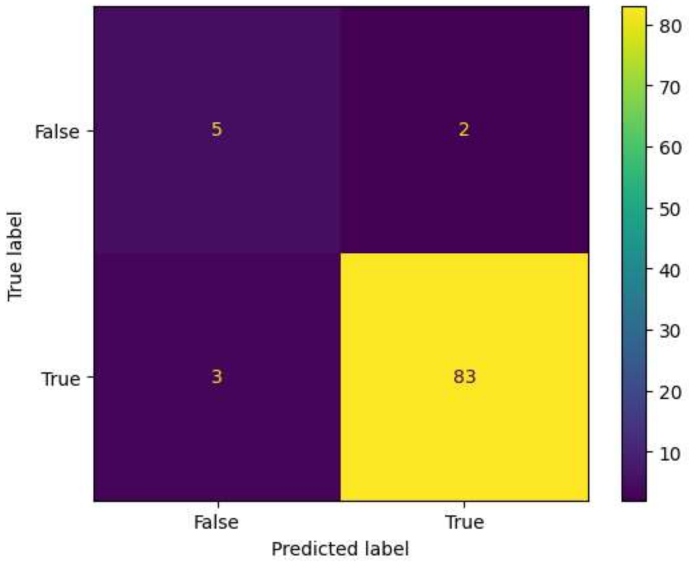
```
print("Mean_Squared_Error :",mean_squared_error(y_test,y_pred3))
print("R2_score :",r2_score(y_test,y_pred3))
print("Mean_Absolute_Error :",mean_absolute_error(y_test,y_pred3))
```

```
Mean_Squared_Error : 0.053763440860215055
R2_score : 0.2275747508305649
Mean_Absolute_Error : 0.053763440860215055
```

```
from mpl_toolkits.mplot3d.axes3d import Axes3D
cm3 = confusion_matrix(y_test,y_pred3)
cr3= classification_report(y_test,y_pred3)
print(cm3)
print(cr3)
a3=accuracy_score(y_test, y_pred3)
print(a3)
```

```
[[ 5  2]
 [ 3 83]]
              precision    recall  f1-score   support

           0       0.62      0.71      0.67         7
           1       0.98      0.97      0.97        86

    accuracy                           0.95        93
   macro avg       0.80      0.84      0.82        93
weighted avg       0.95      0.95      0.95        93

0.946236559139785
```

```
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm3, display_labels = [False, True])
cm_display.plot()
plt.show()
```



```
# Building a Support Vector Machine on train data
from sklearn.svm import SVC
svc = SVC(C=.1, kernel='linear', gamma=1)
svc.fit(x_train, y_train)
```

SVC

```
y_pred4=svc.predict(x_test)
y_pred4
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1])
```

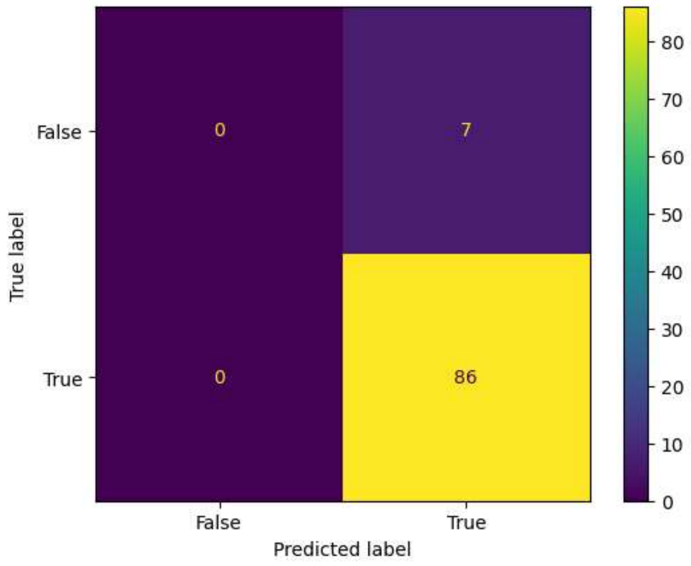
```
print("Mean_Squared_Error :",mean_squared_error(y_test,y_pred4))
print("R2_score :",r2_score(y_test,y_pred4))
print("Mean_Absolute_Error :",mean_absolute_error(y_test,y_pred4))
```

```
Mean_Squared_Error : 0.07526881720430108
R2_score : -0.08139534883720922
Mean_Absolute_Error : 0.07526881720430108
```

```
cm4 = confusion_matrix(y_test,y_pred4)
cr4= classification_report(y_test,y_pred4)
print(cm4)
print(cr4)
a4=accuracy_score(y_test, y_pred4)
print(a4)
```

```
[[ 0  7]
 [ 0 86]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         7
           1       0.92      1.00      0.96        86

    accuracy                           0.92        93
   macro avg       0.46      0.50      0.48        93
weighted avg       0.86      0.92      0.89        93

0.9247311827956989
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
  _warn_prf(average, modifier, msg_start, len(result))
```

```
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm4, display_labels = [False, True])
cm_display.plot()
plt.show()
```



```
from sklearn.neighbors import KNeighborsClassifier
knn= KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train, y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
y_pred5=knn.predict(x_test)
y_pred5
```

```
array([1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1])
```

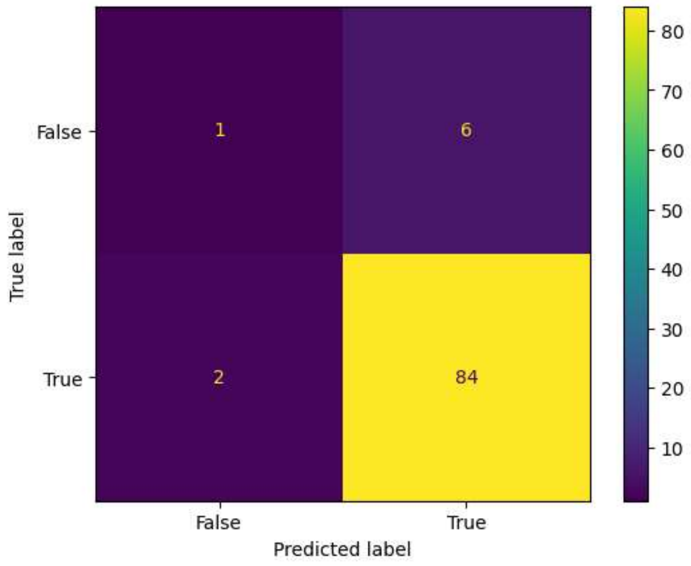
```
print("Mean_Squared_Error :",mean_squared_error(y_test,y_pred5))
print("R2_score :",r2_score(y_test,y_pred5))
print("Mean_Absolute_Error :",mean_absolute_error(y_test,y_pred5))
```

```
Mean_Squared_Error : 0.08602150537634409
R2_score : -0.23588039867109623
Mean_Absolute_Error : 0.08602150537634409
```

```
cm5 = confusion_matrix(y_test,y_pred5)
cr5= classification_report(y_test,y_pred5)
print(cm5)
print(cr5)
a5=accuracy_score(y_test, y_pred5)
print(a5)
```

```
[[ 1  6]
 [ 2 84]]
              precision    recall  f1-score   support

           0       0.33      0.14      0.20         7
           1       0.93      0.98      0.95        86

    accuracy                           0.91        93
   macro avg       0.63      0.56      0.58        93
weighted avg       0.89      0.91      0.90        93

0.9139784946236559
```

```
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm5, display_labels = [False, True])
cm_display.plot()
plt.show()
```



```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(random_state=5)
lr.fit(x_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾       LogisticRegression
LogisticRegression(random_state=5)
```

```
y_pred6=lr.predict(x_test)
y_pred6
```

```
array([1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1])
```

```
print("Mean_Squared_Error :",mean_squared_error(y_test,y_pred6))
print("R2_score :",r2_score(y_test,y_pred6))
print("Mean_Absolute_Error :",mean_absolute_error(y_test,y_pred6))
```

```
Mean_Squared_Error : 0.053763440860215055
R2_score : 0.2275747508305649
Mean_Absolute_Error : 0.053763440860215055
```

```
cm6 = confusion_matrix(y_test,y_pred6)
cr6= classification_report(y_test,y_pred6)
print(cm6)
print(cr6)
a6=accuracy_score(y_test, y_pred6)
print(a6)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm6, display_labels = [False, True])
cm_display.plot()
plt.show()
```

[[5 2]

```
from sklearn.ensemble import AdaBoostClassifier
ad=AdaBoostClassifier(n_estimators=10, random_state=7)
```

0   0.62   0.71   0.67   7

```
ad.fit(x_train,y_train)
y_pred7=ad.predict(x_test)
y_pred7
```

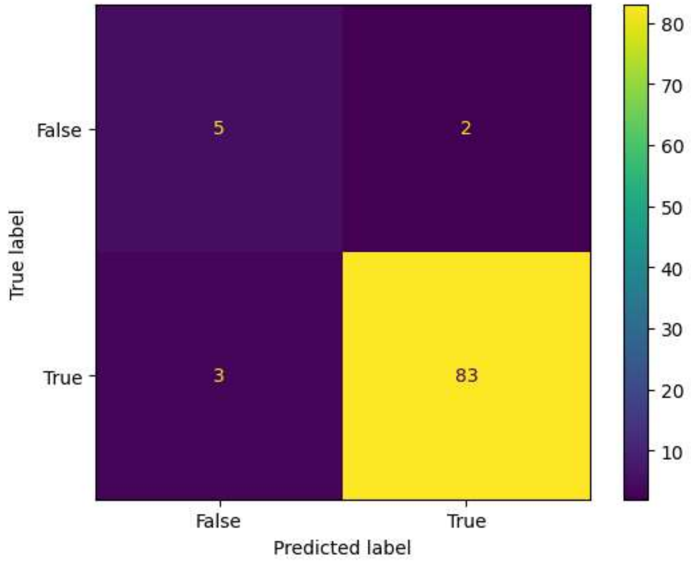
```
array([1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1])
```

```
print("Mean_Squared_Error :",mean_squared_error(y_test,y_pred7))
print("R2_score :",r2_score(y_test,y_pred7))
print("Mean_Absolute_Error :",mean_absolute_error(y_test,y_pred7))
```

```
Mean_Squared_Error : 0.053763440860215055
R2_score : 0.2275747508305649
Mean_Absolute_Error : 0.053763440860215055
```

```
cm7= confusion_matrix(y_test,y_pred7)
cr7= classification_report(y_test,y_pred7)
print(cm7)
print(cr7)
a7=accuracy_score(y_test, y_pred7)
print(a7)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm7, display_labels = [False, True])
cm_display.plot()
plt.show()
```

```
[[ 5  2]
 [ 3 83]]
              precision    recall  f1-score   support

           0       0.62      0.71      0.67         7
           1       0.98      0.97      0.97        86

    accuracy                           0.95        93
   macro avg       0.80      0.84      0.82        93
weighted avg       0.95      0.95      0.95        93

0.946236559139785
```



```
import xgboost as xgb
xg= xgb.XGBClassifier(objective ='reg:linear', colsample_bytree = 0.3, learning_rate = 0.05,
                max_depth = 10, alpha = 1, n_estimators = 100)
xg.fit(x_train, y_train)
```

```
[03:09:15] WARNING: ../src/objective/regression_obj.cu:213: reg:linear is now deprecated in favor of reg:squarederror.
▾                               XGBClassifier
XGBClassifier(alpha=1, base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.3, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.05, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=10, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
```

```
y_pred8=xg.predict(x_test)
y_pred8
```

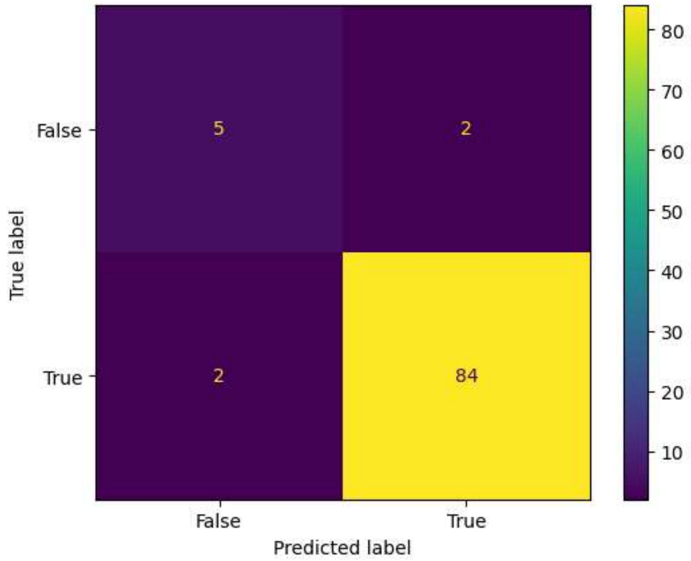
```
array([1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1])
```

```
print("Mean_Squared_Error :",mean_squared_error(y_test,y_pred8))
print("R2_score :",r2_score(y_test,y_pred8))
print("Mean_Absolute_Error :",mean_absolute_error(y_test,y_pred8))
```

```
Mean_Squared_Error : 0.043010752688172046
R2_score : 0.3820598006644519
Mean_Absolute_Error : 0.043010752688172046
```

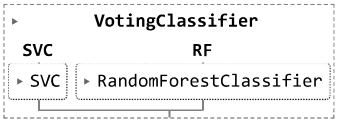
```
cm8= confusion_matrix(y_test,y_pred8)
cr8= classification_report(y_test,y_pred8)
print(cm8)
print(cr8)
a8=accuracy_score(y_test, y_pred8)
print(a8)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm8, display_labels = [False, True])
cm_display.plot()
plt.show()
```

```
[[ 5  2]
 [ 2 84]]
              precision    recall  f1-score   support

           0       0.71      0.71      0.71         7
           1       0.98      0.98      0.98        86

    accuracy                           0.96        93
   macro avg       0.85      0.85      0.85        93
weighted avg       0.96      0.96      0.96        93

0.956989247311828
```



```
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
```

```
estimator = []
estimator.append(('SVC', SVC(gamma ='auto', probability = True)))
estimator.append(('RF', RandomForestClassifier()))
from sklearn.ensemble import VotingClassifier
vot_hard = VotingClassifier( estimators = estimator,voting ='hard')
vot_hard.fit(x_train, y_train)
```



```
y_pred9 = vot_hard.predict(x_test)
y_pred9
```

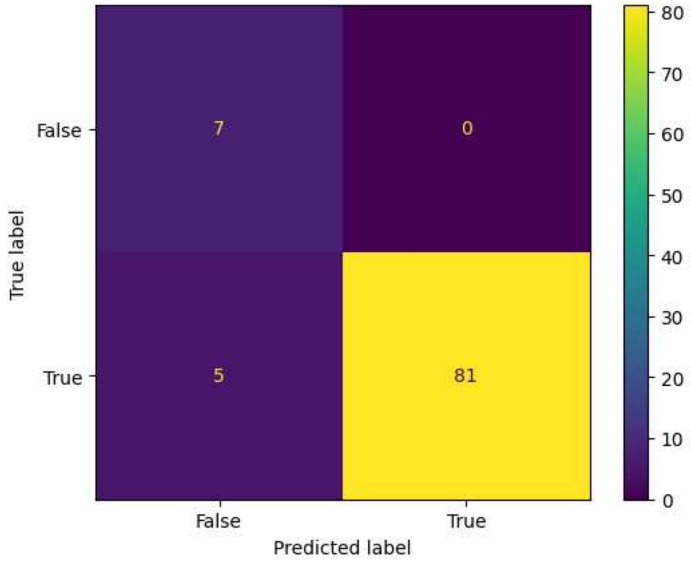
```
array([0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1])
```

```
print("Mean_Squared_Error :",mean_squared_error(y_test,y_pred9))
print("R2_score :",r2_score(y_test,y_pred9))
print("Mean_Absolute_Error :",mean_absolute_error(y_test,y_pred9))
```

```
Mean_Squared_Error : 0.053763440860215055
R2_score : 0.2275747508305649
Mean_Absolute_Error : 0.053763440860215055
```

```
cm9= confusion_matrix(y_test,y_pred9)
cr9= classification_report(y_test,y_pred9)
print(cm9)
print(cr9)
a9=accuracy_score(y_test, y_pred9)
print(a9)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm9, display_labels = [False, True])
cm_display.plot()
plt.show()
```

```
[[ 7  0]
 [ 5 81]]
              precision    recall  f1-score   support

           0       0.58      1.00      0.74         7
           1       1.00      0.94      0.97        86

    accuracy                           0.95        93
   macro avg       0.79      0.97      0.85        93
weighted avg       0.97      0.95      0.95        93

0.946236559139785
```

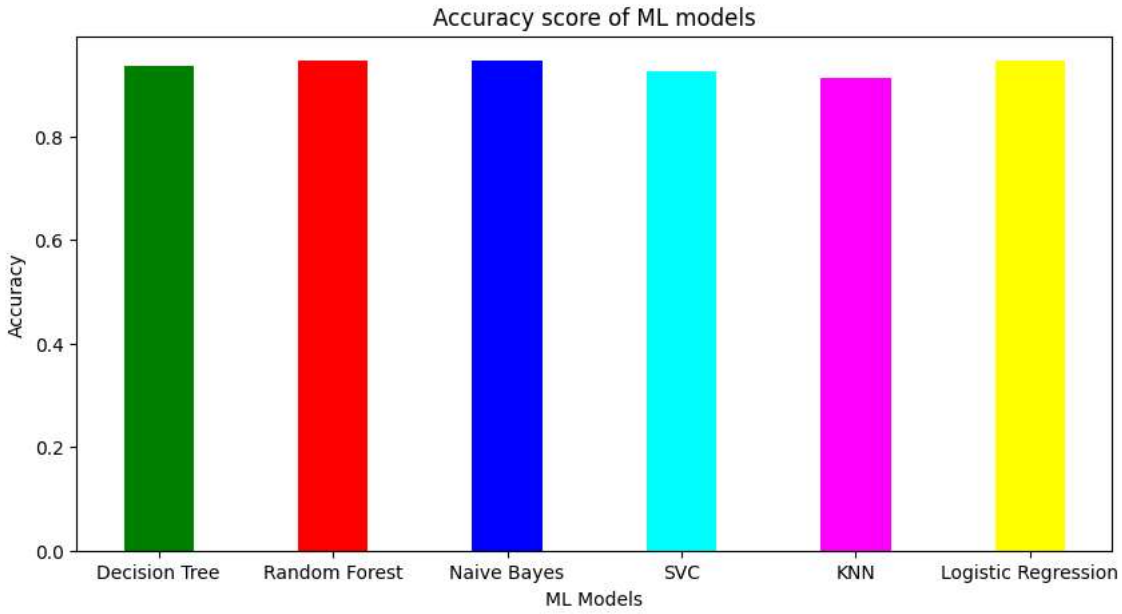


```
ml=["Decision Tree","Random Forest","Naive Bayes","SVC","KNN","Logistic Regression"]
ac=[a,a2,a3,a4,a5,a6]
```

```
ml
```

```
['Decision Tree',
 'Random Forest',
 'Naive Bayes',
 'SVC',
 'KNN',
 'Logistic Regression']
```

```
fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(ml,ac,
        width = 0.4,color=['g','r','b','cyan','magenta','yellow'])

plt.xlabel("ML Models")
plt.ylabel("Accuracy")
plt.title("Accuracy score of ML models")
plt.show()
```


Accuracy score of ML models
Accuracy
Decision Tree
Random Forest
Naive Bayes
SVC
KNN
Logistic Regression
ML Models