In [11]:
```
from sklearn.datasets import load_breast_cancer
import pandas as pd
data=load_breast_cancer()
x = pd.DataFrame(data=data.data,columns=data.feature_names)
y = pd.Series(data.target,name='target')
```

In [12]:
```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 30 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
dtypes: float64(30)
memory usage: 133.5 KB
```

In [13]:
```
x.duplicated().sum()
```

Out[13]:
```
0
```

In [14]:
```
x.describe()
```

Out[14]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	te
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.000000	569.0
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798	...	16.269190	25.6
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	...	4.833242	6.1
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	...	7.930000	12.0
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	...	13.010000	21.0
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	...	14.970000	25.4
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120	...	18.790000	29.7
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	...	36.040000	49.5

8 rows × 30 columns

In [22]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [28]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y)
```

In [26]:
```
scaler=StandardScaler()
```

In [29]:
```
x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)
```

Logistic regression

In [31]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

In [33]:
```
logreg=LogisticRegression()
logreg.fit(x_train,y_train)
```

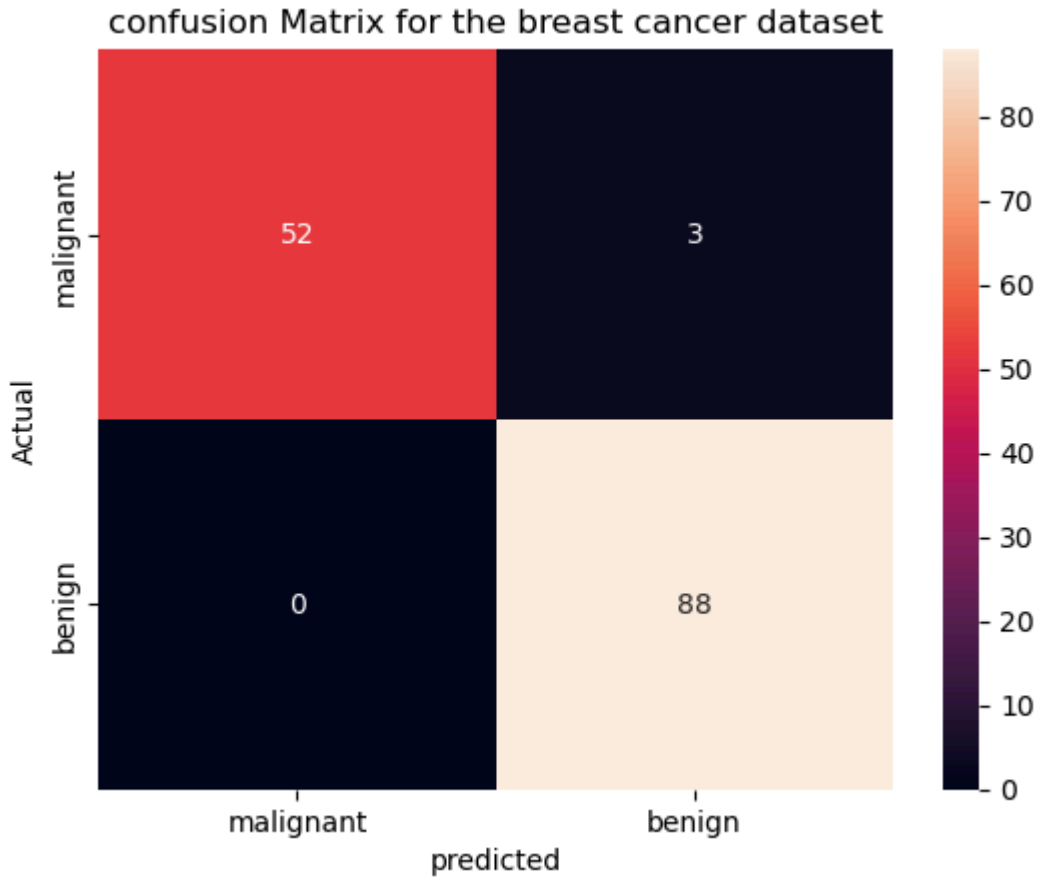
Out[33]:
```
▾ LogisticRegression
LogisticRegression()
```

In [34]:
```
y_pred=logreg.predict(x_test)
```

In [36]:
```
y_pred
```

Out[36]:
```
array([1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1,
       0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,
       0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0,
       0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1])
```

In [43]:
```
import seaborn as sns
con=confusion_matrix(y_test,y_pred)
import matplotlib.pyplot as plt
sns.heatmap(con,annot=True,xticklabels=data.target_names,yticklabels=data.target_names)
plt.xlabel('predicted')
plt.ylabel('Actual')
plt.title('confusion Matrix for the breast cancer dataset')
plt.show()
```



In [49]:
```
cm=confusion_matrix(y_test,y_pred)
print('confusion_matrix:',)
print(cm)
```

```
confusion_matrix:
[[52  3]
 [ 0 88]]
```

In [50]:
```
cr=classification_report(y_test,y_pred)
print('classification report:')
print(cr)
```

```
classification report:
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        55
           1       0.97      1.00      0.98        88

    accuracy                           0.98       143
   macro avg       0.98      0.97      0.98       143
weighted avg       0.98      0.98      0.98       143
```

In [51]:
```
accuracy_score(y_test,y_pred)
```

Out[51]:
```
0.9790209790209791
```

Decision tree classifier

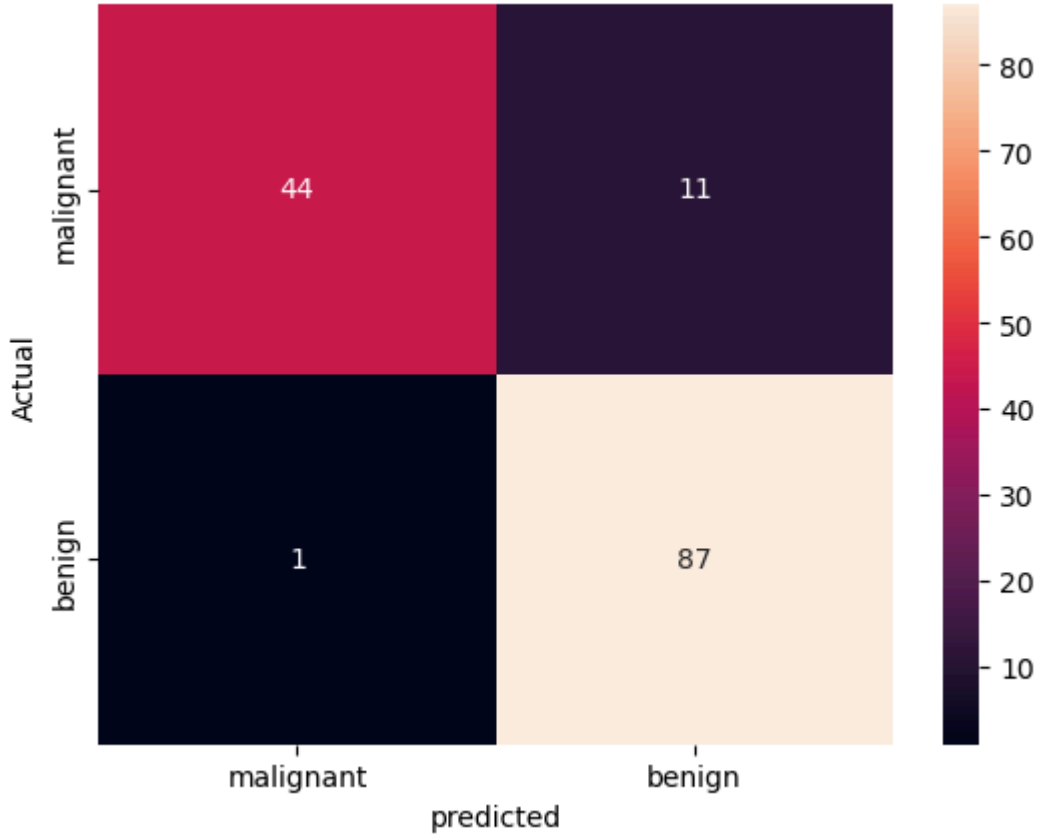
In [54]:
```
from sklearn.tree import DecisionTreeClassifier

tree=DecisionTreeClassifier()
tree.fit(x_train,y_train)
y_pred=tree.predict(x_test)
```

In [57]:
```
con=confusion_matrix(y_test,y_pred)
print('confusion_matrix:',)
print(con)
```

```
confusion_matrix:
[[44 11]
 [ 1 87]]
```

In [59]:
```
sns.heatmap(con,annot=True,xticklabels=data.target_names,yticklabels=data.target_names)
plt.xlabel('predicted')
plt.ylabel('Actual')
plt.show()
```



In [62]:
```
cr=classification_report(y_test,y_pred)
print('classification report:')
print(cr)
```

```
classification report:
              precision    recall  f1-score   support

           0       0.98      0.80      0.88        55
           1       0.89      0.99      0.94        88

    accuracy                           0.92       143
   macro avg       0.93      0.89      0.91       143
weighted avg       0.92      0.92      0.91       143
```

In [63]:
```
accuracy_score(y_test,y_pred)
```

Out[63]:
```
0.916083916083916
```

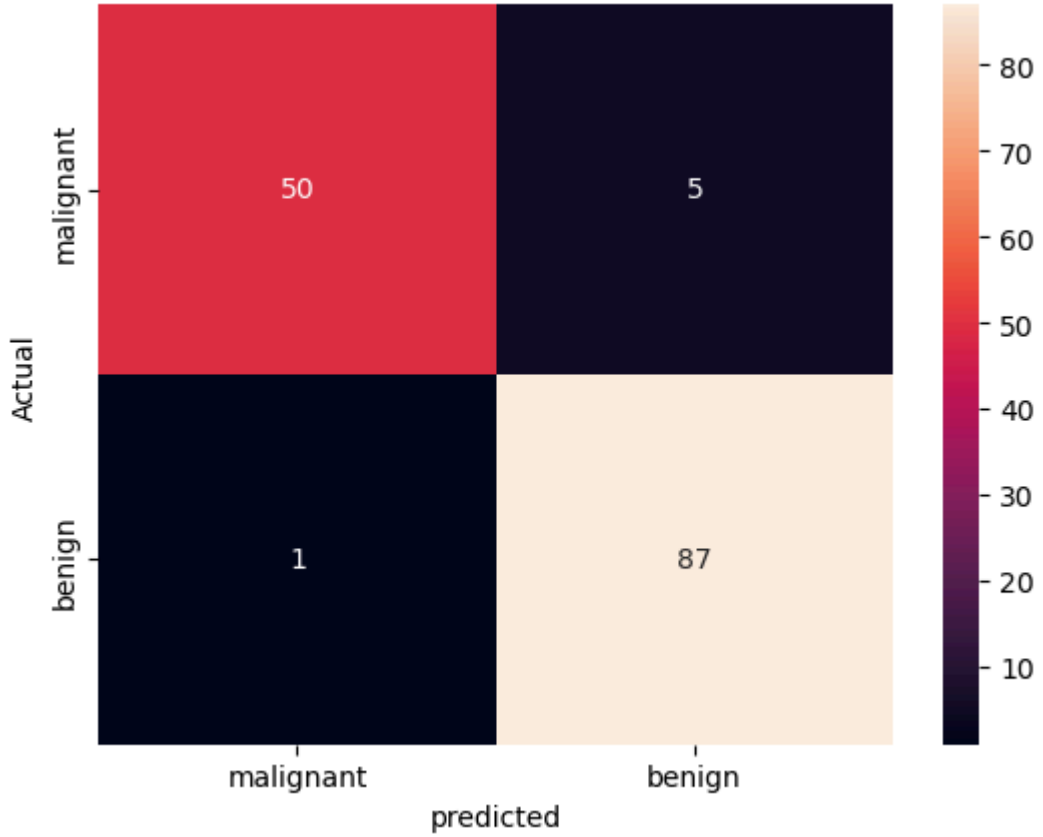
Random forest classifier

In [66]:
```
from sklearn.ensemble import RandomForestClassifier
forest=RandomForestClassifier()
forest.fit(x_train,y_train)
y_pred=forest.predict(x_test)
```

In [67]:
```
con=confusion_matrix(y_test,y_pred)
print('confusion_matrix:',)
print(con)
```

```
confusion_matrix:
[[50  5]
 [ 1 87]]
```

In [69]:
```
sns.heatmap(con,annot=True,xticklabels=data.target_names,yticklabels=data.target_names)
plt.xlabel('predicted')
plt.ylabel('Actual')
plt.show()
```



In [70]:
```
cr=classification_report(y_test,y_pred)
print('classification report:')
print(cr)
```

```
classification report:
              precision    recall  f1-score   support

           0       0.98      0.91      0.94        55
           1       0.95      0.99      0.97        88

    accuracy                           0.96       143
   macro avg       0.96      0.95      0.96       143
weighted avg       0.96      0.96      0.96       143
```

In [80]:
```
accuracy_score(y_test,y_pred)
```

Out[80]:
```
0.972027972027972
```

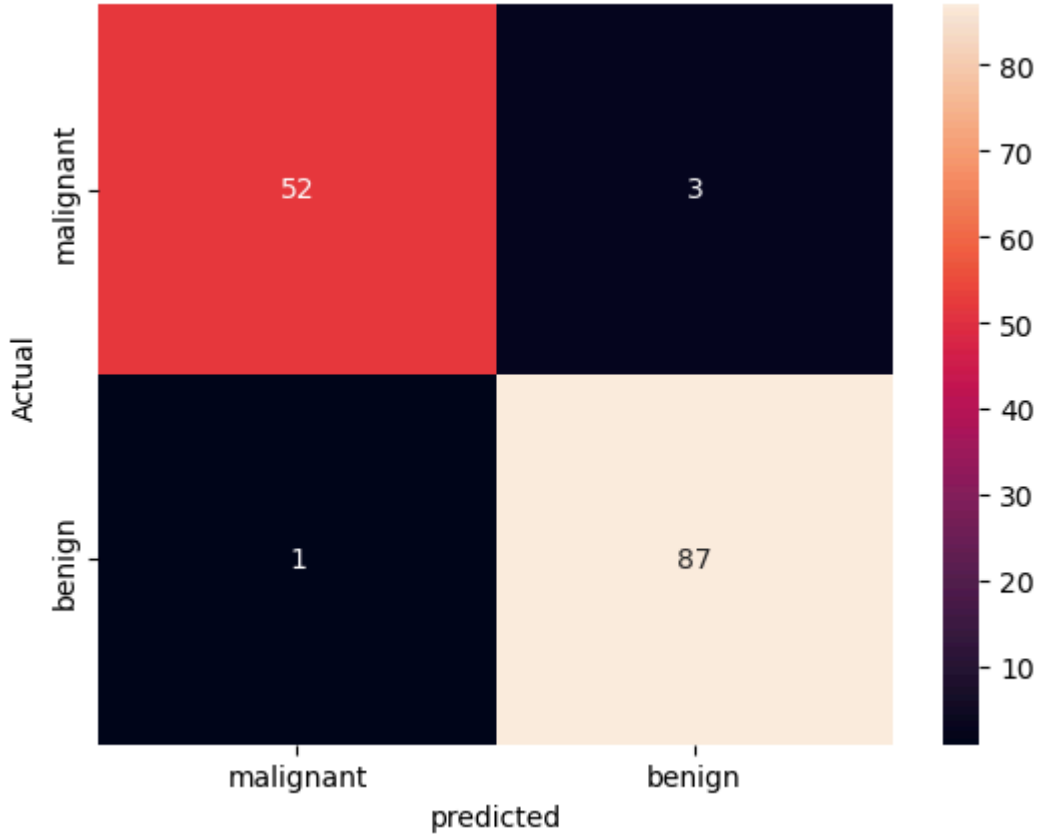
Support vector machine (SVM)

In [76]:
```
from sklearn.svm import SVC
svm=SVC()
svm.fit(x_train,y_train)
y_pred=svm.predict(x_test)
```

In [77]:
```
con=confusion_matrix(y_test,y_pred)
print('confusion_matrix:',)
print(con)
```

```
confusion_matrix:
[[52  3]
 [ 1 87]]
```

In [78]:
```
sns.heatmap(con,annot=True,xticklabels=data.target_names,yticklabels=data.target_names)
plt.xlabel('predicted')
plt.ylabel('Actual')
plt.show()
```



In [81]:
```
cr=classification_report(y_test,y_pred)
print('classification report:')
print(cr)
```

```
classification report:
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        55
           1       0.97      0.99      0.98        88

    accuracy                           0.97       143
   macro avg       0.97      0.97      0.97       143
weighted avg       0.97      0.97      0.97       143
```

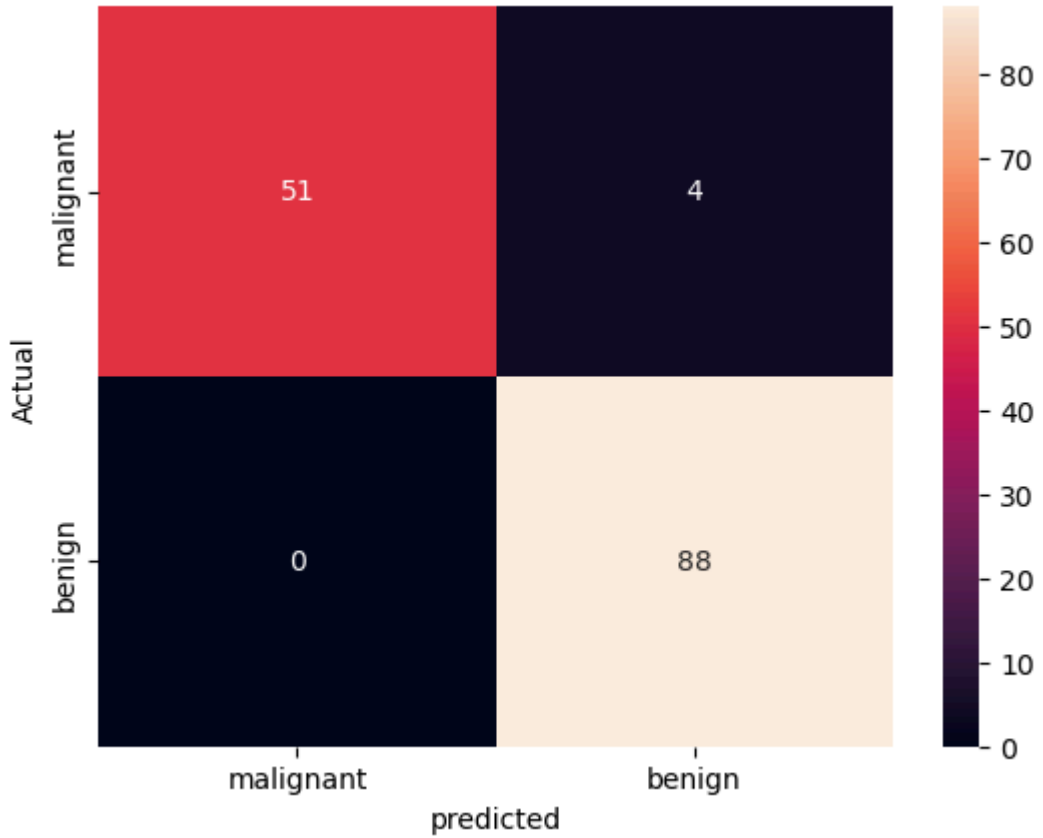
In [82]:
```
accuracy_score(y_test,y_pred)
```

Out[82]: 0.972027972027972

K-Nearst neighbors (K-NN)

In [86]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
y_pred=knn.predict(x_test)
```

In [88]:
```
sns.heatmap(con,annot=True,xticklabels=data.target_names,yticklabels=data.target_names)
plt.xlabel('predicted')
plt.ylabel('Actual')
plt.show()
```



In [90]:
```
con=confusion_matrix(y_test,y_pred)
print('confusion_matrix:',)
print(con)
```

```
confusion_matrix:
[[51  4]
 [ 0 88]]
```

In [91]:
```
cr=classification_report(y_test,y_pred)
print('classification report:')
print(cr)
```

```
classification report:
              precision    recall  f1-score   support

           0       1.00      0.93      0.96        55
           1       0.96      1.00      0.98        88

    accuracy                           0.97       143
   macro avg       0.98      0.96      0.97       143
weighted avg       0.97      0.97      0.97       143
```

In [92]:
```
accuracy_score(y_test,y_pred)
```

Out[92]:
```
0.972027972027972
```

Model comparison

After analyzing the classification report, confusion matrix, and accuracy score of each model the K-NN model is the best-performing model with an accuracy of 0.972 and the decision Tree classifier is the worst algorithm as compared to other models