Machine Learning Lab program 4

Name – Pranshu Aggarwal
Class – CSE – 3
Enroll – 40576802717

AIM: Estimate the precision recall accuracy f-measure of the decision classifier on a breast cancer dataset using 10-fold cross validation

GitHub link: - https://github.com/pranshuag9/machine-learning-lab/blob/main/lab5/decision%20trees%20using%2010%20fold%20cross%20validation%20.ipynb

Program Snippets: -

1. Loading dataset

Loading dataset

In [2]:
```
import pandas as pd
df = pd.read_csv("cancer.csv")
```

In [3]:
```
df.head(10)
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	co
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1
5	843786	M	12.45	15.70	82.57	477.1	0.12780	0.17000	0.1
6	844359	M	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.1
7	84458202	M	13.71	20.83	90.20	577.9	0.11890	0.16450	0.0
8	844981	M	13.00	21.82	87.50	519.8	0.12730	0.19320	0.1
9	84501001	M	12.46	24.04	83.97	475.9	0.11860	0.23960	0.2

10 rows × 33 columns

In [4]:
```
df.tail(5)
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	co
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.2
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.1
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.0
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.3
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.0

5 rows × 33 columns

2. Data Cleaning/ Preprocessing

dropping id and unnamed: 32 column because they are noise are doesnt affect result

In [38]:
```
df = df.drop(["id"], axis= 1)
```

In [40]:
```
df.columns.values
```

```
array(['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean',
       'concavity_mean', 'concave points_mean', 'symmetry_mean',
       'fractal_dimension_mean', 'radius_se', 'texture_se',
       'perimeter_se', 'area_se', 'smoothness_se', 'compactness_se',
       'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype=object)
```

In [42]:
```
df = df.drop(["Unnamed: 32"], axis=1)
```

In [43]:
```
df.columns.values
```

```
array(['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean',
       'concavity_mean', 'concave points_mean', 'symmetry_mean',
       'fractal_dimension_mean', 'radius_se', 'texture_se',
       'perimeter_se', 'area_se', 'smoothness_se', 'compactness_se',
       'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'], dtype=object)
```

In [66]:
```
# Normalization:
x = (x - np.min(x)) / (np.max(x) - np.min(x))
x
```

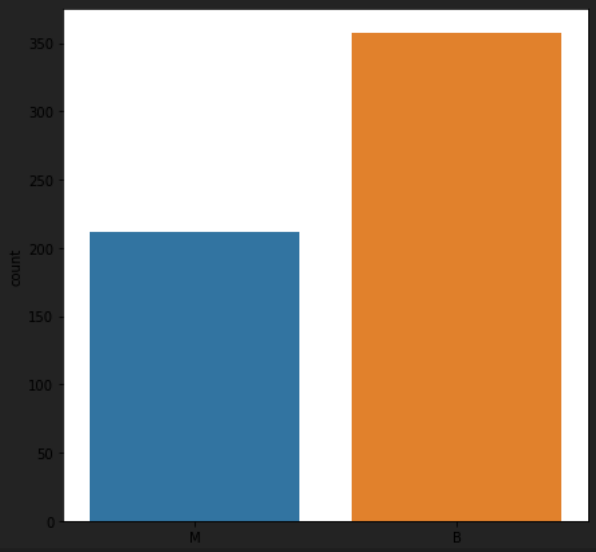
	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	poin
0	0.521037	0.022658	0.545989	0.363733	0.593753	0.792037	0.703140	0.731
1	0.643144	0.272574	0.615783	0.501591	0.289880	0.181768	0.203608	0.348
2	0.601496	0.390260	0.595743	0.449417	0.514309	0.431017	0.462512	0.635
3	0.210090	0.360839	0.233501	0.102906	0.811321	0.811361	0.565604	0.522
4	0.629893	0.156578	0.630986	0.489290	0.430351	0.347893	0.463918	0.518
...	...	...	...	...	...	...	...	...
564	0.690000	0.428813	0.678668	0.566490	0.526948	0.296055	0.571462	0.690
565	0.622320	0.626987	0.604036	0.474019	0.407782	0.257714	0.337395	0.486
566	0.455251	0.621238	0.445788	0.303118	0.288165	0.254340	0.216753	0.263
567	0.644564	0.663510	0.665538	0.475716	0.588336	0.790197	0.823336	0.755
568	0.036869	0.501522	0.028540	0.015907	0.000000	0.074351	0.000000	0.000

569 rows × 10 columns

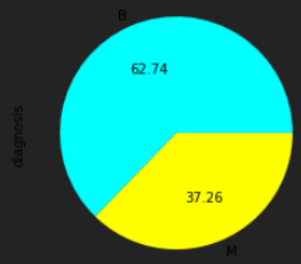
3. Data Visualization

In [48]:
```
plt.figure(figsize=(7,7))
sns.countplot(df['diagnosis'])
plt.show()
```

```
D:\Softwares\Anaconda\envs\tensorflow-gpu\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



In [55]:
```
df.diagnosis.value_counts().plot(kind = "pie" , colors = ("cyan","yellow"), autopct = "%.2f")
plt.show()
```



4. Dividing data into train/test splits

In [67]:
```
from sklearn.model_selection import train_test_split

#for checking testing results
from sklearn.metrics import classification_report, confusion_matrix

#for visualizing tree
from sklearn.tree import plot_tree

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)

print("Training split input- ", x_train.shape)
print("Testing split input- ", x_test.shape)
```

```
Training split input-  (455, 10)
Testing split input-  (114, 10)
```

5. Using Decision classifier method with 10 fold cross validation on breast cancer dataset

In [74]:
```
from sklearn.model_selection import cross_val_score,KFold

kfold = KFold(n_splits=10, random_state=True, shuffle=True) #for 10 folds
y_pred = dt.predict(x_test)
score = cross_val_score(dt, x, y, cv = kfold)
```

6. Accuracy using 10 folds

In [75]:
```
print("decesion Tree Accuracy: {0:.2%}".format(accuracy_score(y_pred, y_test)))
print("Cross validation score: {0:.2%}".format(np.mean(score)))
```

```
decesion Tree Accuracy: 92.98%
Cross validation score: 92.27%
```

In [76]:
```
y_pred = dt.predict(x_test)
print("Classification report - \n", classification_report(y_test,y_pred))
```

```
Classification report -
              precision    recall  f1-score   support

           B       0.95      0.93      0.94        67
           M       0.90      0.94      0.92        47

    accuracy                           0.93       114
   macro avg       0.93      0.93      0.93       114
weighted avg       0.93      0.93      0.93       114
```

7. Confusion matrix

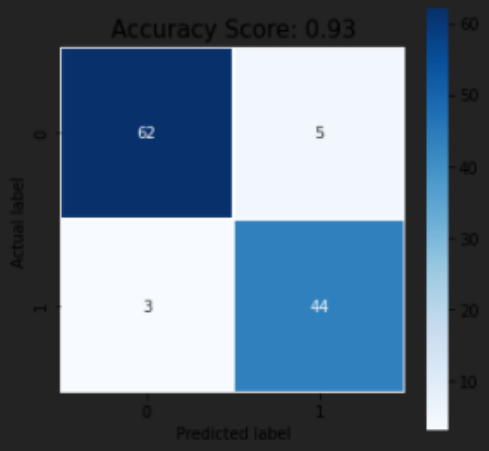
In [77]:
```
cm=confusion_matrix(y_test,y_pred)
cm
```

```
array([[62,  5],
       [ 3, 44]], dtype=int64)
```

In [73]:
```
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=1.0, annot=True,square = True,  cmap = 'Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

all_sample_title = 'Accuracy Score: {0}'.format(round(dt.score(x_test, y_test),2))
plt.title(all_sample_title, size = 15)
```

```
Text(0.5, 1.0, 'Accuracy Score: 0.93')
```



8. Precision :-

```
In [88]: true_negative,false_positive,false_negative,true_positive =cm.ravel()
         (true_negative,false_positive,false_negative,true_positive)

Out[88]: (62, 5, 3, 44)

In [93]: precision = true_positive/(true_positive+false_positive)
         print("Precision of decision classifier : {0:.2%}".format(precision))

         Precision of decision classifier : 89.80%
```

9. Recall & f1 score:-

```
In [90]: recall = true_positive/(true_positive+false_negative)
         print("recall of decision classifier : {0:.2%}".format(recall))

         recall of decision classifier : 93.62%

In [91]: f1= (2*precision*recall)/(precision+recall)
         print("f1 score of decision classifier : {0:.2%}".format(f1))

         f1 score of decision classifier : 91.67%
```