```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import os
os.chdir("/content/drive/MyDrive/ML Assignment")
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
data = pd.read_csv("/content/drive/MyDrive/ML Assignment/breast-cancer.csv")
```

```
data.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	con points_
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10

5 rows × 32 columns

```
data.shape
```

```
(569, 32)
```

```
data.isnull()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	conca points_me
0	False	False	False	False	False	False	False	False	False	Fal
1	False	False	False	False	False	False	False	False	False	Fal
2	False	False	False	False	False	False	False	False	False	Fal
3	False	False	False	False	False	False	False	False	False	Fal
4	False	False	False	False	False	False	False	False	False	Fal
...	...	...	...	...	...	...	...	...	...	
564	False	False	False	False	False	False	False	False	False	Fal
565	False	False	False	False	False	False	False	False	False	Fal
566	False	False	False	False	False	False	False	False	False	Fal
567	False	False	False	False	False	False	False	False	False	Fal
568	False	False	False	False	False	False	False	False	False	Fal

569 rows × 32 columns

```
data.drop(columns='id',inplace=True)
data.head()
```

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symme
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	

5 rows × 31 columns

```
data.describe()
```

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_m
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304

8 rows × 30 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   diagnosis                569 non-null    object
 1   radius_mean              569 non-null    float64
 2   texture_mean             569 non-null    float64
 3   perimeter_mean           569 non-null    float64
 4   area_mean                569 non-null    float64
 5   smoothness_mean          569 non-null    float64
 6   compactness_mean         569 non-null    float64
 7   concavity_mean           569 non-null    float64
 8   concave points_mean      569 non-null    float64
 9   symmetry_mean            569 non-null    float64
 10  fractal_dimension_mean   569 non-null    float64
 11  radius_se                569 non-null    float64
 12  texture_se               569 non-null    float64
 13  perimeter_se             569 non-null    float64
 14  area_se                  569 non-null    float64
 15  smoothness_se            569 non-null    float64
 16  compactness_se           569 non-null    float64
 17  concavity_se             569 non-null    float64
 18  concave points_se        569 non-null    float64
 19  symmetry_se              569 non-null    float64
 20  fractal_dimension_se     569 non-null    float64
 21  radius_worst             569 non-null    float64
 22  texture_worst            569 non-null    float64
 23  perimeter_worst          569 non-null    float64
 24  area_worst               569 non-null    float64
 25  smoothness_worst         569 non-null    float64
 26  compactness_worst        569 non-null    float64
 27  concavity_worst          569 non-null    float64
 28  concave points_worst     569 non-null    float64
 29  symmetry_worst           569 non-null    float64
 30  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), object(1)
memory usage: 137.9+ KB
```

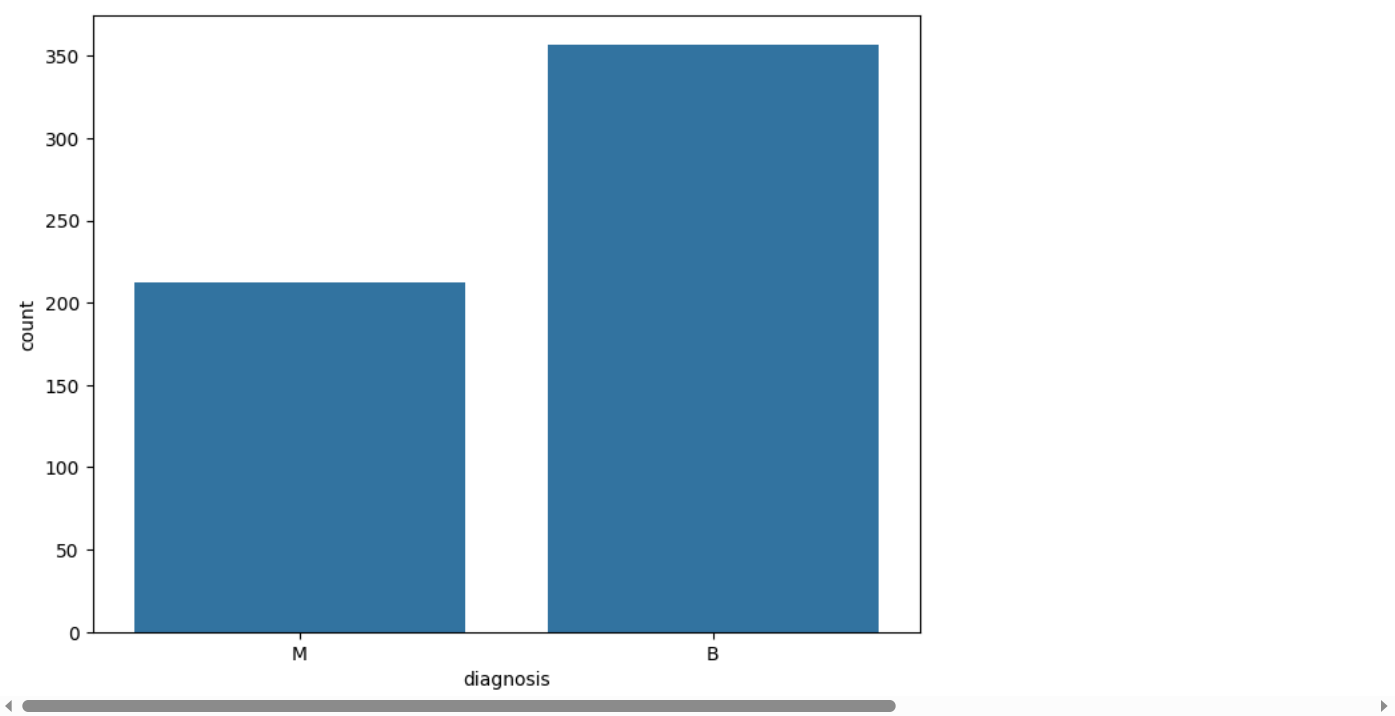
```
data.duplicated().sum()
```

```
0
```

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

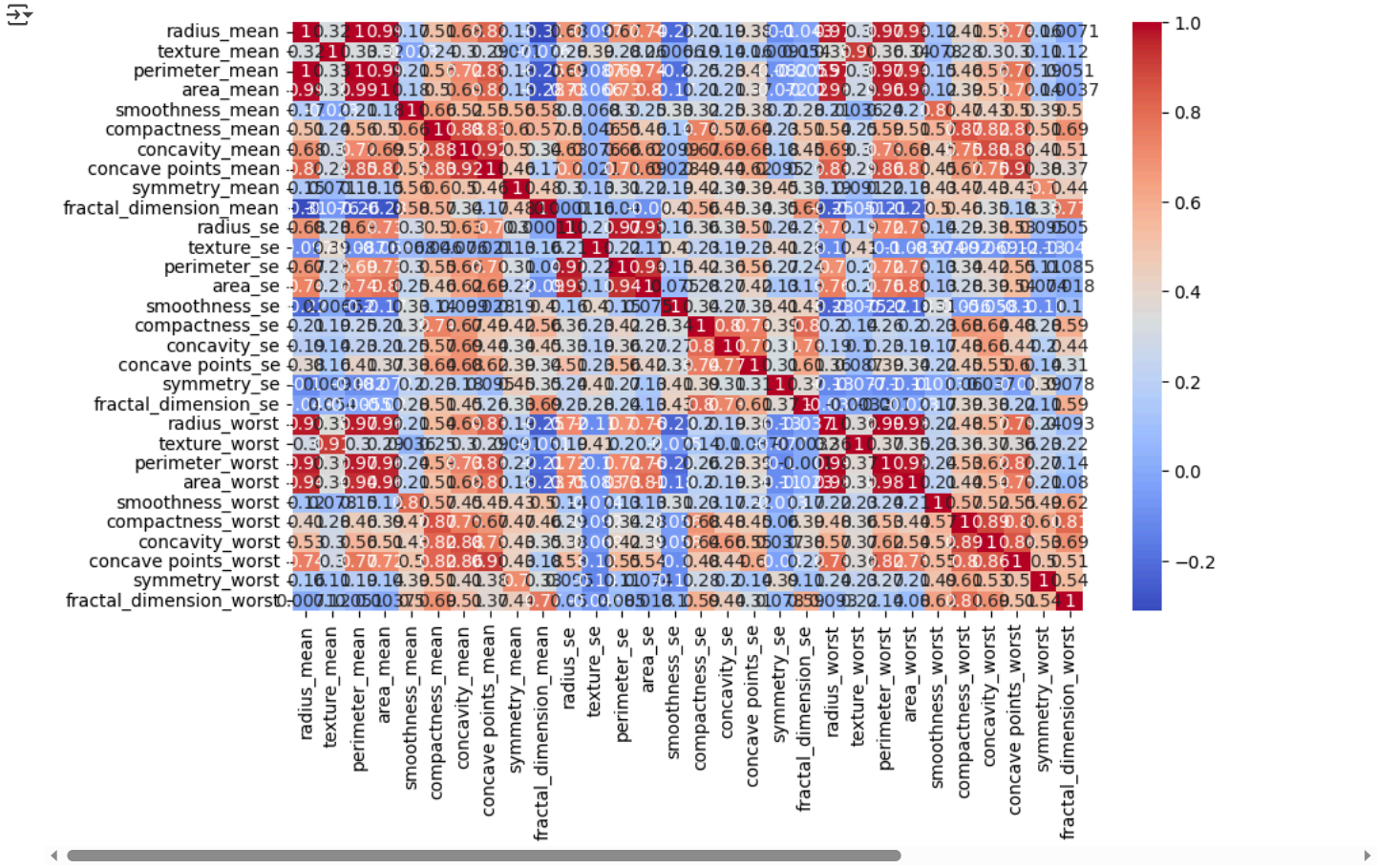
```
B    357
M    212
Name: diagnosis, dtype: int64
```

```
plt.figure(figsize=(8,6))
sns.countplot(data = data ,x= data['diagnosis'])
plt.show()
```




```
corr_matrix = data.corr()
```

```
fig, ax = plt.subplots(figsize=(10,6))
sns.heatmap(corr_matrix, annot=True, cmap = 'coolwarm', ax=ax)
plt.show()
```




```
X = data.iloc[:, 1:-1].values
y = data.iloc[:, 0].values
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
print(y)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 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 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 1
 0 1 0 1 1 0 0 0 1 1 0 1 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 0
 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 0 1 1 0 1 1 1 0 0 0 1 0 1 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 0 1 0 1 1 1 0 1
 0 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0 1 0 1 0 0 1 0 1 1 1 1 0 0 1 1 0 0
 0 1 0 0 0 0 0 1 1 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0
 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0 1 1
 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0
 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 0
 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 1
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0]
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Decision Tree Classification

```
from sklearn.tree import DecisionTreeClassifier
DTC = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
```

```
DTC.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

```
y_pred = DTC.predict(X_test)
```

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

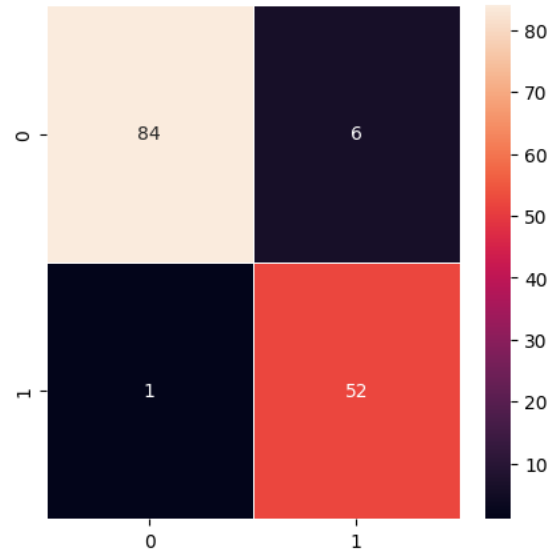
```
[[84  6]
 [ 1 52]]
0.951048951048951
```

```
# Calculate and print the mean accuracy
mean_accuracy = a.mean()
print("Accuracy Scores: ", accuracy_score)
print("Mean Accuracy:", mean_accuracy)
```

```
Accuracy Scores:  <function accuracy_score at 0x78ec7705ba30>
Mean Accuracy: 0.951048951048951
```

```
plt.figure(figsize=(5,5))
sns.heatmap(cm,annot=True , linewidth=0.5)
print('True Negative:',cm[0,0])
print('False Negative:',cm[1,0])
print('False pasitive:',cm[0,1])
print('True positive:',cm[1,1])
```

```
True Negative: 84
False Negative: 1
False pasitive: 6
True positive: 52
```



Random Forest Classification

```
from sklearn.ensemble import RandomForestClassifier
RFC = RandomForestClassifier(n_estimators = 20, criterion = 'entropy', random_state = 42)
RFC.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=20, random_state=42)
```

```
y_pred1 = RFC.predict(X_test)
```

```
cm1 = confusion_matrix(y_test, y_pred1)
print(cm1)
b=accuracy_score(y_test, y_pred1)
print(b)
```

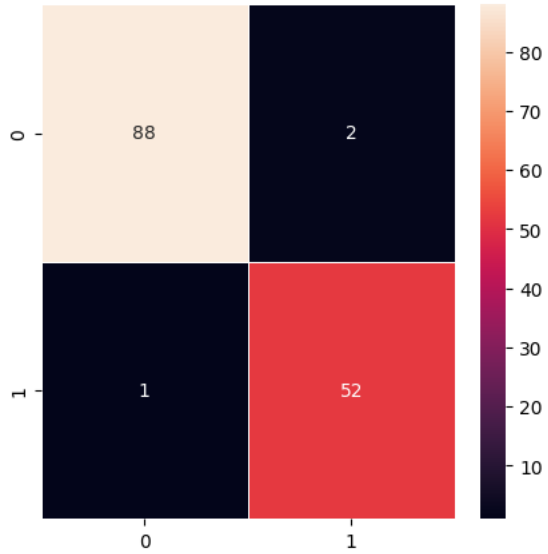
```
[[88  2]
 [ 1 52]]
0.9790209790209791
```

```
# Calculate and print the mean accuracy
mean_accuracy = b.mean()
print("Accuracy Scores: ", accuracy_score)
print("Mean Accuracy:", mean_accuracy)
```

```
Accuracy Scores:  <function accuracy_score at 0x78ec7705ba30>
Mean Accuracy: 0.9790209790209791
```

```
plt.figure(figsize=(5,5))
sns.heatmap(cm1,annot=True , linewidth=0.5)
print('True Negative:',cm1[0,0])
print('False Negative:',cm1[1,0])
print('False pasitive:',cm1[0,1])
print('True positive:',cm1[1,1])
```

```
True Negative: 88
False Negative: 1
False pasitive: 2
True positive: 52
```



Logistic Regression

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(random_state = 0)
LR.fit(X_train, y_train)
```

```
LogisticRegression(random_state=0)
```

```
y_pred2 = LR.predict(X_test)
```

```
cm2 = confusion_matrix(y_test, y_pred2)
print(cm2)
c=accuracy_score(y_test, y_pred2)
print(c)
```

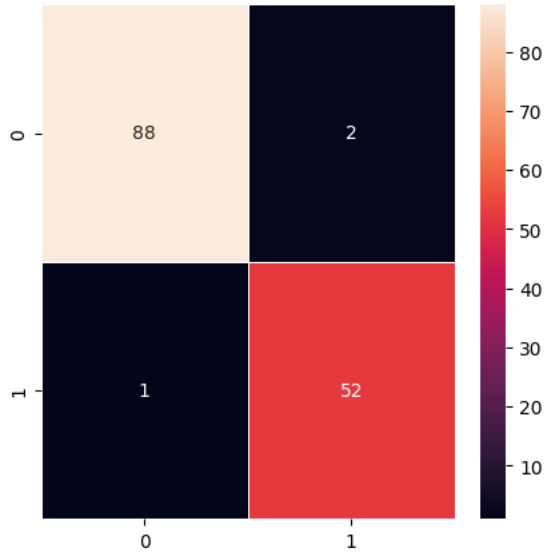
```
[[86  4]
 [ 4 49]]
0.9440559440559441
```

```
# Calculate and print the mean accuracy
mean_accuracy = c.mean()
print("Accuracy Scores: ", accuracy_score)
print("Mean Accuracy:", mean_accuracy)
```

```
Accuracy Scores:  <function accuracy_score at 0x78ec7705ba30>
Mean Accuracy: 0.9440559440559441
```

```
plt.figure(figsize=(5,5))
sns.heatmap(cm1,annot=True , linewidth=0.5)
print('True Negative:',cm2[0,0])
print('False Negative:',cm2[1,0])
print('False pasitive:',cm2[0,1])
print('True positive:',cm2[1,1])
```

```
True Negative: 86
False Negative: 4
False pasitive: 4
True positive: 49
```




```
model=['Decision Tree Classification','Random Forest Classification','LogisticRegression']
accuracy=[a,b,c]
```