Breast Cancer

High level question: Breast Cancer is one of the deadly kinds of disease among women; which caused death of 627,000 lives alone. This high death rate because of breast cancer growth needs considerable attention, for early recognition with the goal that prevention should be possible in time. So, we propose develop a machine learning model to detect early using features of which are used by physicians to detect.

Machine learning model prediction: We detect the mass in the breast is benign (benign) or malignant (malignant) using the features below.

Data Content

1. **ID number**
2. **Diagnosis (M = malignant, B = benign)**
3. **radius (mean of distances from centre to points on the perimeter)**
4. **texture (standard deviation of Gray-scale values)**
5. **perimeter**
6. **area**
7. **smoothness (local variation in radius lengths)**
8. **compactness (perimeter^2 / area - 1.0)**
9. **concavity (severity of concave portions of the contour)**
10. **concave points (number of concave portions of the contour)**
11. **symmetry**
12. **fractal dimension ("coastline approximation" - 1)**

Using Pandas library, we import the data file using the pd.read_csv function. There are a total of 569 rows.

```
In [2]:  1 data = pd.read_csv('data.csv')
In [3]:  1 data
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430
...	...	...	...	...	...	...	...	...	...	...
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000

569 rows × 33 columns

Label encoding

Using label encoding we encode the diagnosis column using label encoder function of Sklearn library.

In [4]:

```
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'species'.
data['diagnosis']= label_encoder.fit_transform(data['diagnosis'])

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

Out[4]: array([1, 0])

Dealing with Nan Values and label/target column

The column diagnosis is a label(y) so we place in other variable and remove from dataset. "Unnamed: 32" had Nan so we removed it. Id does not contribute in any way to cancer classification so it was removed.

```
# y includes our labels and x includes our features
y = data.diagnosis                                  # M or B
list = ['Unnamed: 32','id','diagnosis']
x = data.drop(list,axis = 1 )
x.head()
```

:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_
0	17.99	10.38	122.80	1001.0	0.11840	0.2
1	20.57	17.77	132.90	1326.0	0.08474	0.(
2	19.69	21.25	130.00	1203.0	0.10960	0.1
3	11.42	20.38	77.58	386.1	0.14250	0.2
4	20.29	14.34	135.10	1297.0	0.10030	0.1

5 rows × 30 columns

```
data.isnull().values.any()
```

: False

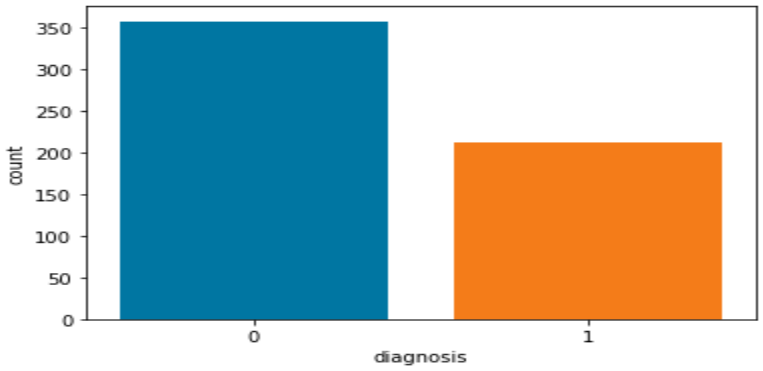
Data Visualization

Ratio of class labels

The number Benign and Malignant classes are in a ratio of 60:40 which is okay ratio for classification labels in dataset.

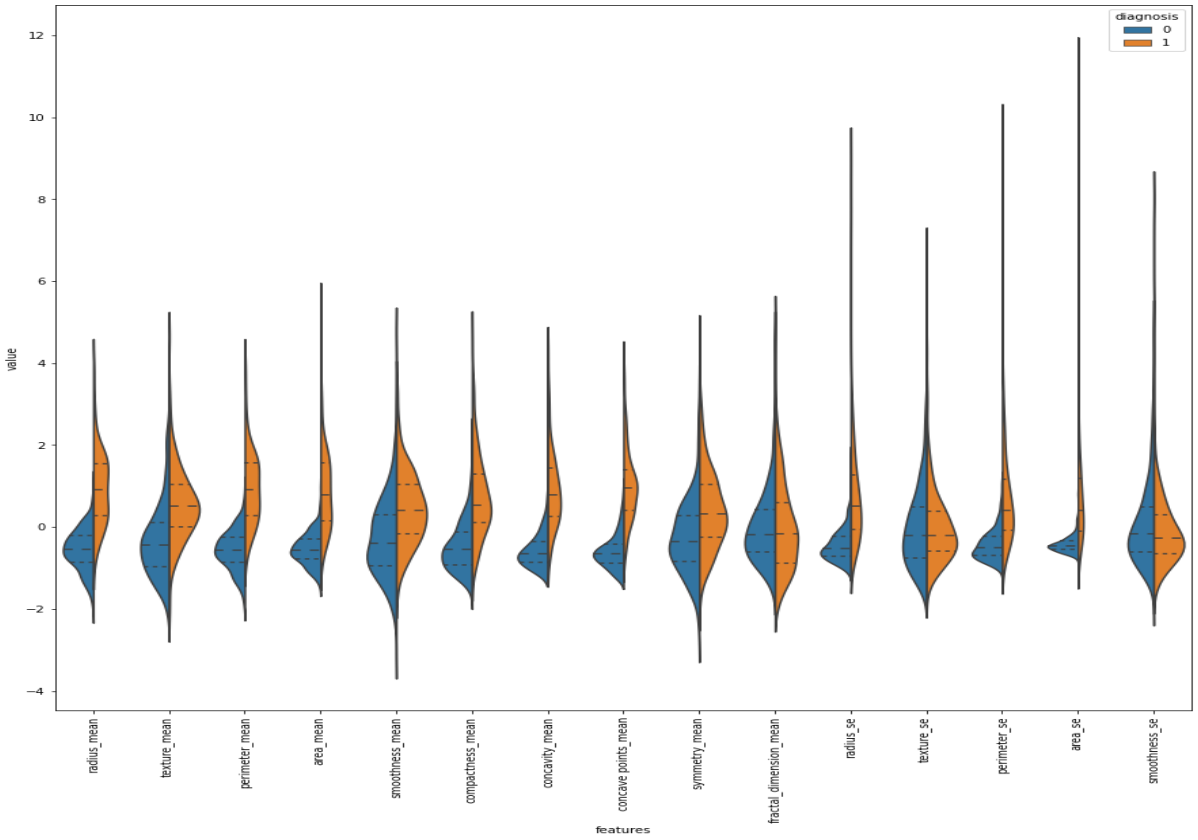
```
ax = sns.countplot(y,label="Count")        # M =
B, M = y.value_counts()
print('Number of Benign: ',B)
print('Number of Malignant : ',M)
```

```
Number of Benign:  357
Number of Malignant :  212
```

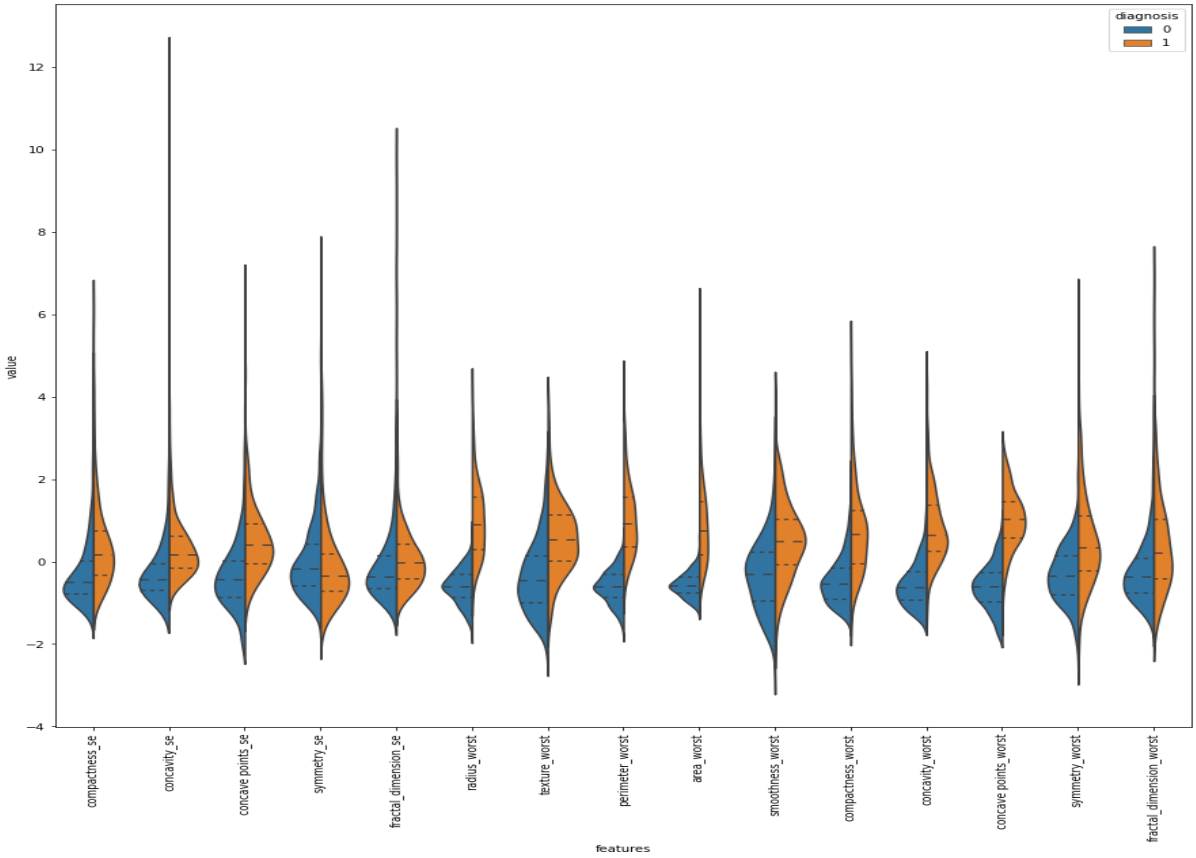


Violin Plot of feature's mean with respect to classification

The violin plot will help us to select features that will help for classification such as if look at "radius_mean" the mean for each classes are quite different so this will help in distinguishing the diagnosis. While if you look at "smoothness_se" the mean is almost same and will not help much in classification.

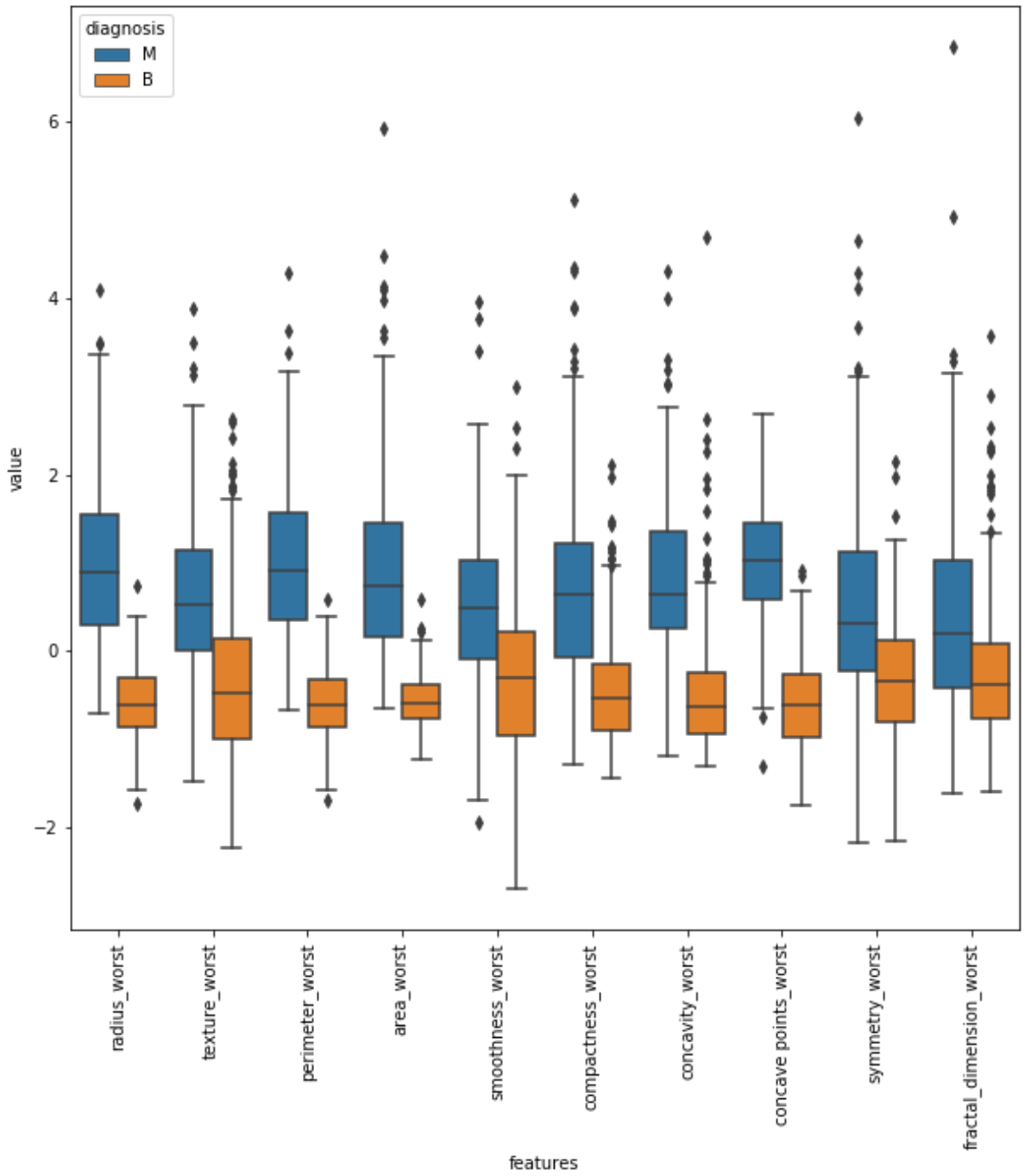


The features were divided in groups of 15 so that chart could be seen clearly.



Box Plots

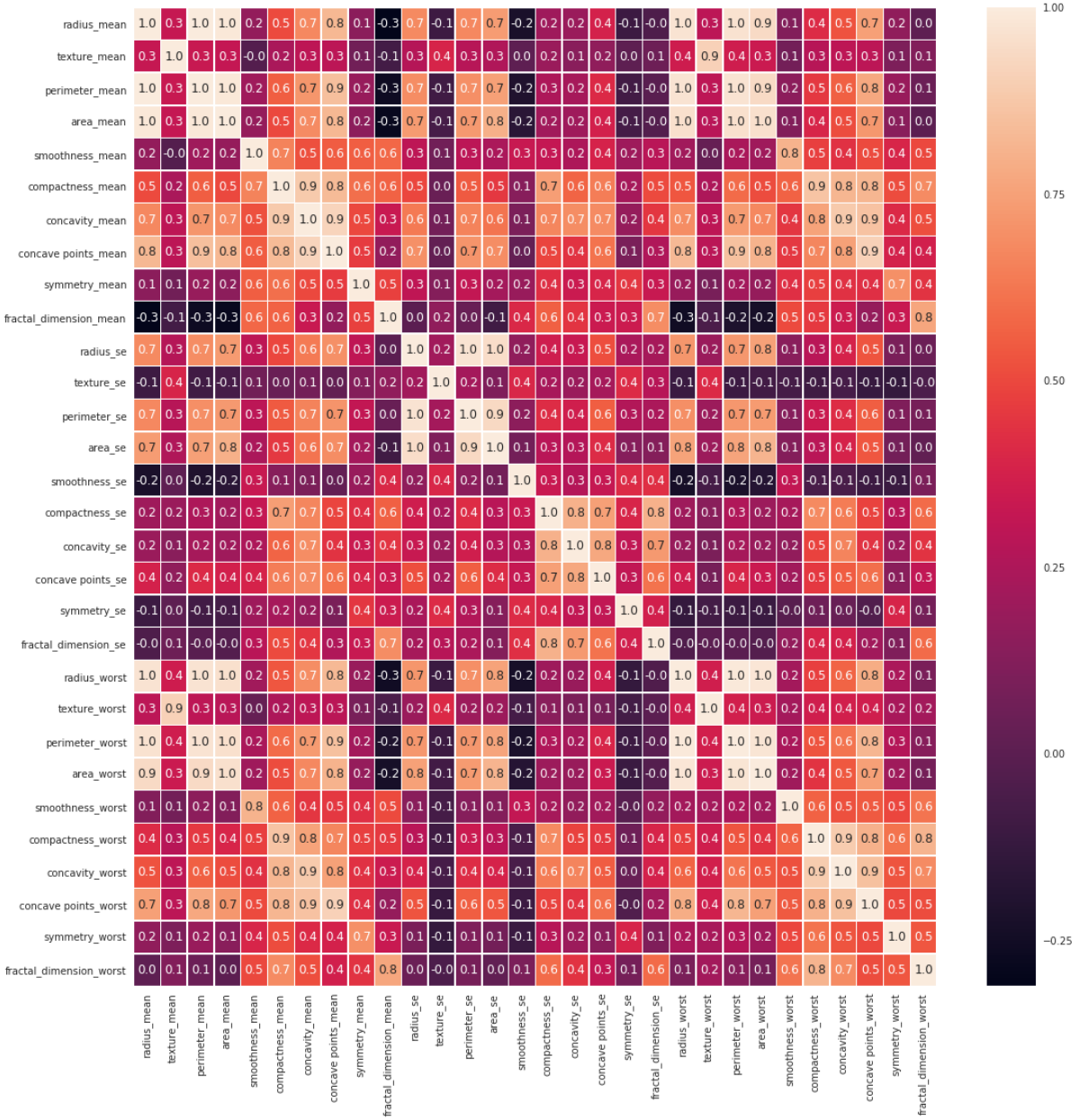
As an alternative of violin plot, box plot can be used as they are also useful in terms of seeing outliers.



Feature selection with correlation

As it can be seen in map heat figure “radius_mean”, “perimeter_mean” and “area_mean” are correlated with each other so we will use only “area_mean. We choose the features based on a value of 0.3 such that if features have Pearson relation of more than 0.3 or less than -0.3 than we call those features highly correlated. So, lets find other correlated features and look accuracy with random forest classifier.

“Area_worst” and “area_mean” are correlated with each other. Therefore, using this method, I only use “area_mean”, “area_se”, “area_worst”, “concavity_worst”, “concavity_se”, “texture_mean”.



Dropping Columns

```
drop_list1 = ['perimeter_mean','radius_mean','compactness_mean','concave points_mean','radius_se','perimeter_se',
              'radius_worst','perimeter_worst','compactness_worst','concave points_worst','compactness_se',
              'concave points_se','texture_worst','area_worst']
x_1 = x.drop(drop_list1,axis = 1 )
x_1.head()
```

	texture_mean	area_mean	smoothness_mean	concavity_mean	symmetry_mean	fractal_dimension_mean	texture_se	area_se	smoothness_se	concavity
0	10.38	1001.0	0.11840	0.3001	0.2419	0.07871	0.9053	153.40	0.006399	0.05

Making the machine learning model

Here we use train_test_split to spit the data into 80/20 split of training data and test data.

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor

# X, y = load training data

X_train, X_test, y_train, y_test = train_test_split(x_1, y, test_size=0.2)


############## Unscaled test/train set ##############
X_train, X_test, y_train, y_test = train_test_split(
    x_1,
    y,
    test_size=0.2,
    random_state=69
)

```

Training and testing the model accuracy

I have included multiple models to compare the accuracy of different classifiers. It is shown that feature selection benefits the random classifier the most. Accuracy is almost 95%. I tried to show importance of feature selection and data visualization. Default data includes 33 features but after feature selection we drop this number from 33 to 16 with accuracy above 95%

```
classification_models = [
    DecisionTreeClassifier(),#max_depth=5),
    RandomForestClassifier(),#max_depth=5, n_estimators=10, max_features=1),
    AdaBoostClassifier(),
    GaussianNB(),
    QuadraticDiscriminantAnalysis()]

scores = []
for model in classification_models:
    model.fit(X_train_scaled, y_train.values.ravel())
    score = model.score(X_test_scaled, y_test)
    model_name = type(model).__name__
    if model_name=='SVC' and model.kernel=='rbf': model_name+=' RBF kernel'
    scores.append((model_name,(f'{100*score:.2f}%')))
# Make it pretty
scores_df = pd.DataFrame(scores,columns=['Classifier','Accuracy Score'])
scores_df.sort_values(by='Accuracy Score',axis=0,ascending=False)
```

	Classifier	Accuracy Score
1	RandomForestClassifier	95.61%
4	QuadraticDiscriminantAnalysis	94.74%
2	AdaBoostClassifier	93.86%
0	DecisionTreeClassifier	92.11%
3	GaussianNB	90.35%

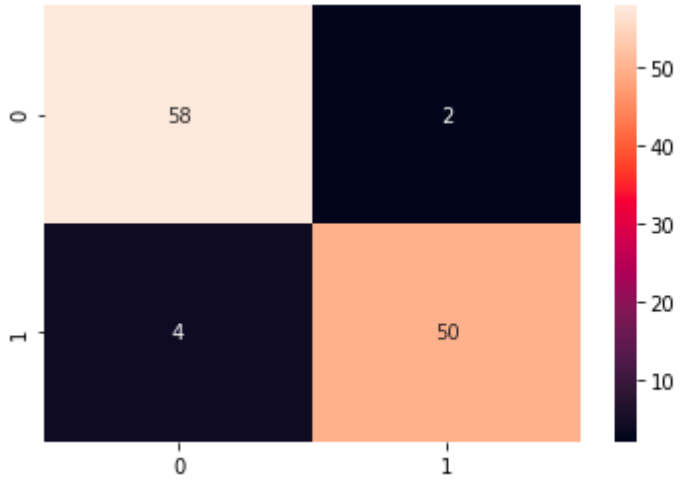
Confusion matrix

As it can be seen in confusion matrix, we make few wrong, only 6 are wrong.

```
from sklearn.metrics import f1_score,confusion_matrix
from sklearn.metrics import accuracy_score
ac = accuracy_score(y_test,model.predict(X_test_scaled))
print('Accuracy is: ',ac)
cm = confusion_matrix(y_test,model.predict(X_test_scaled))
sns.heatmap(cm,annot=True,fmt="d")
```

```
Accuracy is:  0.9473684210526315
```

```
<AxesSubplot:>
```



Acknowledgements

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.