In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
```

- **IMPORT AND VISUALIZE THE DATA**

In [2]:
```
data = pd.read_csv("./breast-cancer-wisconsin-data/data.csv")
data.head()
```

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	com
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	

5 rows × 33 columns

Here we have dataset with 2 different classes. M (malignant), B (benign). We have 30 different features in each row that describes our data. Our dataset is linearly seperable using our 30 features. we have 569 instances:
212 M
357 B

In [3]:
```
data.keys()   # name of all the columns in our data
```

Out[3]:
```
Index(['id', '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 [4]:
```
data.describe()
```

Out[4]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compa
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	

8 rows × 32 columns

In [5]:
```
# we don't need id, diagnosis and also last column which is NaN
X = data.iloc[:, 2:-1]
```

In [6]:
```
X.shape
```

Out[6]:
```
(569, 30)
```

In [7]:
```
Y = data.iloc[:, 1]
Y.shape
```

Out[7]:
```
(569,)
```

In [8]:
```
Y = [1 if i=='M' else 0 for i in Y]
```

In [9]:
```
Y[1:25]
```

Out[9]:
```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1]
```

In [10]:
```
data.tail()
```

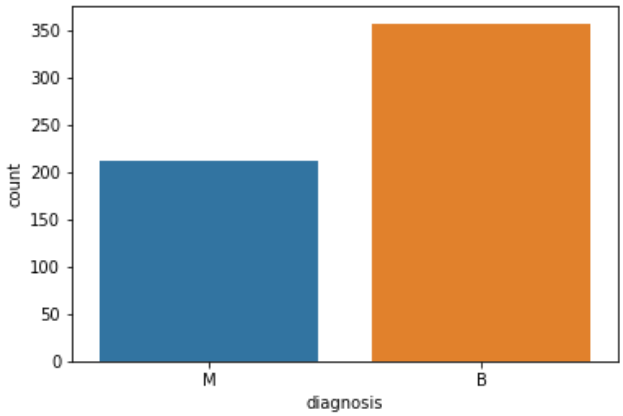
Out[10]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	com
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	
566	926954	M	16.60	28.08	108.30	858.1	0.08455	
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	
568	92751	B	7.76	24.54	47.92	181.0	0.05263	

5 rows × 33 columns

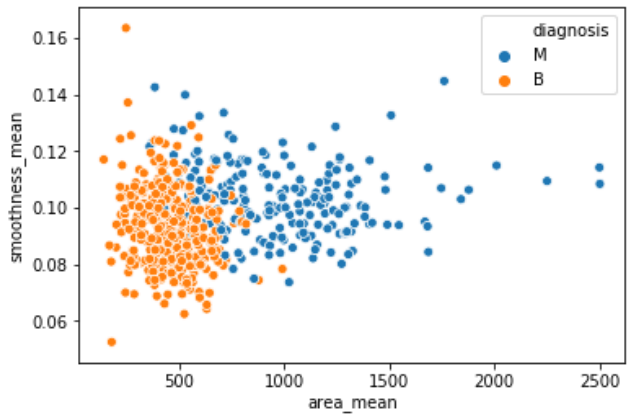
In [11]:
```
sns.countplot(data['diagnosis'])
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb5f7d81dd8>
```



In [12]:
```
sns.scatterplot(x = 'area_mean', y = 'smoothness_mean', hue = 'diagnosis', d
ata = data)
```

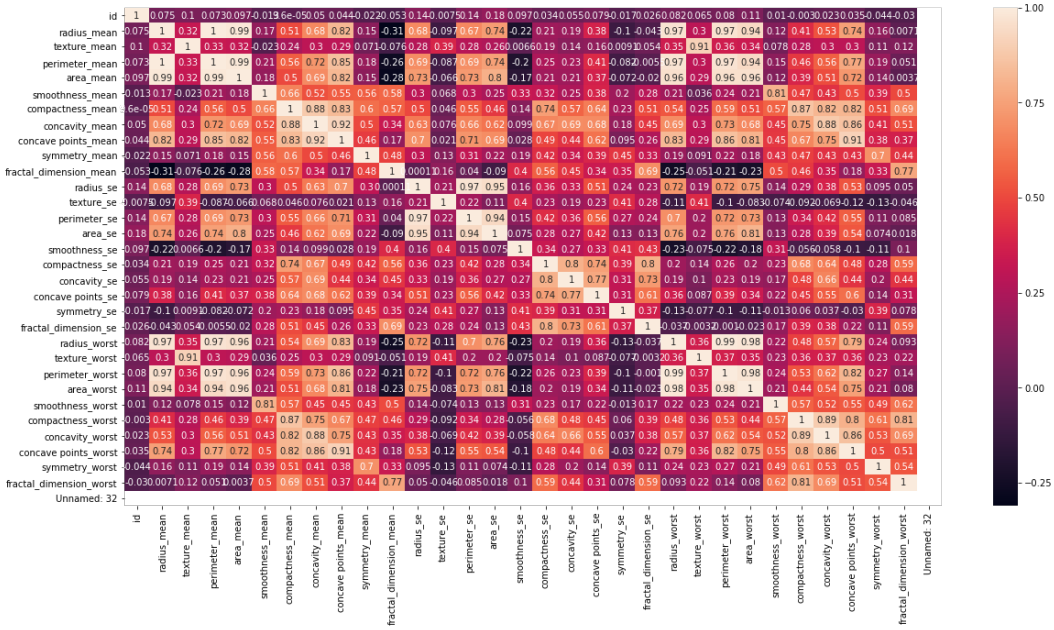
Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb5f5cd0390>
```



As it is visible above our two features have very different scales and we need to scale the features to normalize data for better results

In [13]:
```
plt.figure(figsize=(20, 10))
sns.heatmap(data.corr(), annot= True)
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb5f5c62278>
```



- **TRAIN AND TEST MODEL**

Support Vector Machine is a binary classifier that can detect two classes.

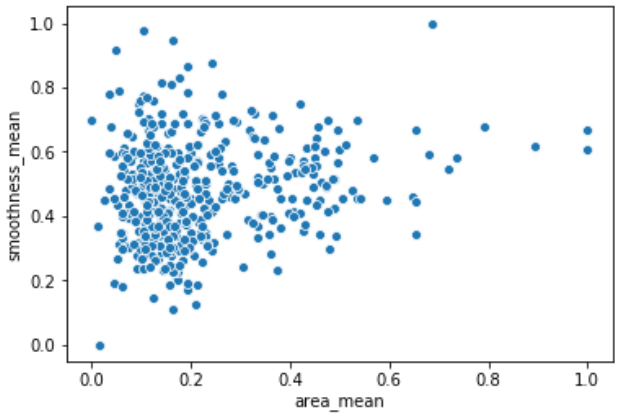
In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, ran
dom_state=0)
```

In [15]:
```
min_train = X_train.min()
range_train = (X_train-min_train).max() # find biggest difference between mi
n value and any point of dataset
X_train= (X_train - min_train)/range_train
```

In [16]:
```
min_test = X_test.min()
range_test = (X_test-min_test).max()
X_test= (X_test - min_test)/range_test
```

In [17]:
```
sns.scatterplot(x = 'area_mean', y = 'smoothness_mean', data = X_train)
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb5ebdd3b38>
```



In [18]:
```
print(X_train.shape)
print(X_test.shape)
```

```
(455, 30)
(114, 30)
```

In [19]:
```
len(y_train)
```

Out[19]:
```
455
```

In [20]:
```
classifier = SVC(C=1, gamma=1, kernel='rbf')
classifier.fit(X_train, y_train)
```

Out[20]:
```
SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=1, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

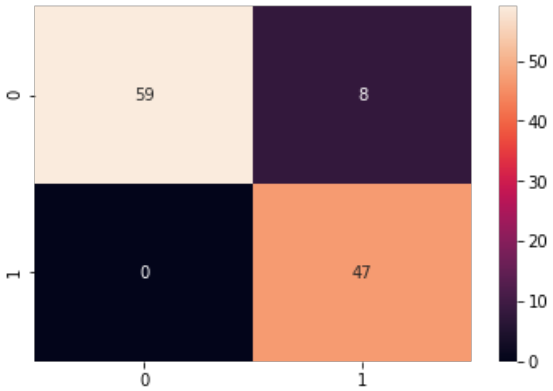
In [21]:
```
y_predict = classifier.predict(X_test)
y_predict
```

Out[21]:
```
array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1,
       0, 1, 1, 0])
```

In [22]:
```
cm = confusion_matrix(y_test, y_predict)
```

In [23]:
```
sns.heatmap(cm, annot= True)
```

Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb5ec53b5f8>
```



While we have more type 1 error, number of type two errors has decreased to 0

In [24]:
```
print("print accuracy of svm alg: ", classifier.score(X_test,y_test))
```

```
print accuracy of svm alg:  0.9298245614035088
```

In [ ]:
```

```