In [1]:

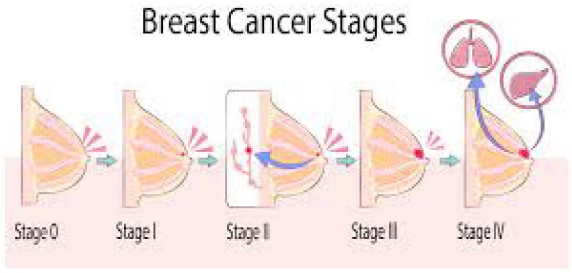
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly.express as px
from pandas_profiling import ProfileReport
from plotly.offline import iplot
!pip install joypy
import joypy

plt.rcParams['figure.figsize'] = 8, 5
plt.style.use("fivethirtyeight")

data = pd.read_csv('../input/breast-cancer-wisconsin-data/data.csv')
```

```
Collecting joypy
  Downloading joypy-0.2.2-py2.py3-none-any.whl (8.3 kB)
Requirement already satisfied: pandas>=0.20.0 in /opt/conda/lib/python3.7/site-packages (from joypy) (1.0.3)
Requirement already satisfied: scipy>=0.11.0 in /opt/conda/lib/python3.7/site-packages (from joypy) (1.4.1)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.7/site-packages (from joypy) (3.2.1)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages (from joypy) (1.18.1)
Requirement already satisfied: pytz>=2017.2 in /opt/conda/lib/python3.7/site-packages (from pandas>=0.20.0->joypy) (2019.3)
Requirement already satisfied: python-dateutil>=2.6.1 in /opt/conda/lib/python3.7/site-packages (from pandas>=0.20.0->joypy) (2.8.
1)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.7/site-packages (from matplotlib->joypy) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python3.7/site-packages (from matplotlib->joypy) (1.2.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /opt/conda/lib/python3.7/site-packages (from matplotlib
->joypy) (2.4.7)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-packages (from python-dateutil>=2.6.1->pandas>=0.20.0->jo
ypy) (1.14.0)
Installing collected packages: joypy
Successfully installed joypy-0.2.2
```

Cancer occurs when changes called mutations take place in genes that regulate cell growth. The mutations let the cells divide and multiply in an uncontrolled way.

Breast cancer is cancer that develops in breast cells. Typically, the cancer forms in either the lobules or the ducts of the breast. Lobules are the glands that produce milk, and ducts are the pathways that bring the milk from the glands to the nipple. Cancer can also occur in the fatty tissue or the fibrous connective tissue within your breast.

The uncontrolled cancer cells often invade other healthy breast tissue and can travel to the lymph nodes under the arms. The lymph nodes are a primary pathway that help the cancer cells move to other parts of the body. Source.



Types of Breast Cancer

There are several types of breast cancer, and they are broken into two main categories: “invasive” and “noninvasive,” or in situ. While invasive cancer has spread from the breast ducts or glands to other parts of the breast, noninvasive cancer has not spread from the original tissue.

These two categories are used to describe the most common types of breast cancer, which include:

1. **Ductal carcinoma in situ. Ductal carcinoma in situ (DCIS) is a noninvasive condition. With DCIS, the cancer cells are confined to the ducts in your breast and haven’t invaded the surrounding breast tissue.**
2. **Lobular carcinoma in situ. Lobular carcinoma in situ (LCIS) is cancer that grows in the milk-producing glands of your breast. Like DCIS, the cancer cells haven’t invaded the surrounding tissue.**
3. **Invasive ductal carcinoma. Invasive ductal carcinoma (IDC) is the most common type of breast cancer. This type of breast cancer begins in your breast’s milk ducts and then invades nearby tissue in the breast. Once the breast cancer has spread to the tissue outside your milk ducts, it can begin to spread to other nearby organs and tissue.**
4. **Invasive lobular carcinoma. Invasive lobular carcinoma (ILC) first develops in your breast’s lobules and has invaded nearby tissue.**

Description of Data

In [2]:

```
#Describing data

data.describe()
```

Out[2]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	...	texture_wo
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.0000

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	...	texture_wor
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	...	25.6772
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	...	6.1462
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	...	12.0200
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	...	21.0800
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	...	25.4100
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	...	29.7200
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	...	49.5400

8 rows × 32 columns

In [3]:

```
#covariance in data

data.cov()
```

Out[3]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_
id	1.563015e+16	3.287883e+07	5.364807e+07	2.222490e+08	4.262946e+09	-22802.053383	631.883924	499127.721282	214217.739286	-75792.6
radius_mean	3.287883e+07	1.241892e+01	4.907582e+00	8.544714e+01	1.224483e+03	0.008454	0.094197	0.190128	0.112475	0.0
texture_mean	5.364807e+07	4.907582e+00	1.849891e+01	3.443976e+01	4.859938e+02	-0.001415	0.053767	0.103692	0.048977	0.0
perimeter_mean	2.222490e+08	8.544714e+01	3.443976e+01	5.904405e+02	8.435772e+03	0.070836	0.714714	1.387234	0.802360	0.1
area_mean	4.262946e+09	1.224483e+03	4.859938e+02	8.435772e+03	1.238436e+05	0.876178	9.264931	19.244924	11.241958	1.4
smoothness_mean	-2.280205e+04	8.454460e-03	-1.414779e-03	7.083607e-02	8.761781e-01	0.000198	0.000490	0.000585	0.000302	0.0
compactness_mean	6.318839e+02	9.419706e-02	5.376681e-02	7.147141e-01	9.264931e+00	0.000490	0.002789	0.003718	0.001703	0.0
concavity_mean	4.991277e+05	1.901276e-01	1.036923e-01	1.387234e+00	1.924492e+01	0.000585	0.003718	0.006355	0.002850	0.0
concave points_mean	2.142177e+05	1.124751e-01	4.897693e-02	8.023604e-01	1.124196e+01	0.000302	0.001703	0.002850	0.001506	0.0
symmetry_mean	-7.579262e+04	1.427317e-02	8.418876e-03	1.219216e-01	1.459596e+00	0.000215	0.000873	0.001094	0.000492	0.0
fractal_dimension_mean	-4.635137e+04	-7.753706e-03	-2.321158e-03	-4.485888e-02	-7.034264e-01	0.000058	0.000211	0.000190	0.000046	0.0
radius_se	4.959431e+06	6.636503e-01	3.290374e-01	4.661401e+00	7.149094e+01	0.001176	0.007286	0.013970	0.007511	0.0
texture_se	-5.190618e+05	-1.891886e-01	9.166951e-01	-1.162988e+00	-1.286717e+01	0.000531	0.001346	0.003352	0.000460	0.0
perimeter_se	3.471365e+07	4.803550e+00	2.449449e+00	3.405303e+01	5.170100e+02	0.008420	0.058612	0.106443	0.055753	0.0
area_se	1.010874e+09	1.179682e+02	5.084087e+01	8.234928e+02	1.280852e+04	0.157742	1.094708	2.239119	1.218501	0.2
smoothness_se	3.632916e+04	-2.355336e-03	8.540990e-05	-1.478818e-02	-1.762210e-01	0.000014	0.000021	0.000024	0.000003	0.0
compactness_se	7.603492e+04	1.300051e-02	1.478660e-02	1.091112e-01	1.339725e+00	0.000080	0.000699	0.000957	0.000341	0.0
concavity_se	2.084665e+05	2.065883e-02	1.860393e-02	1.672962e-01	2.205952e+00	0.000105	0.000910	0.001663	0.000514	0.0
concave points_se	6.076269e+04	8.179563e-03	4.348380e-03	6.105470e-02	8.084602e-01	0.000033	0.000209	0.000336	0.000147	0.0
symmetry_se	-1.788548e+04	-3.038982e-03	3.245070e-04	-1.639643e-02	-2.108964e-01	0.000023	0.000100	0.000117	0.000031	0.0
fractal_dimension_se	8.510281e+03	-3.976249e-04	6.197726e-04	-3.551365e-04	-1.851854e-02	0.000011	0.000071	0.000095	0.000026	0.0
radius_worst	4.979381e+07	1.651375e+01	7.329267e+00	1.138581e+02	1.637521e+03	0.014487	0.136643	0.265181	0.155721	0.0
texture_worst	4.973106e+07	6.433100e+00	2.411015e+01	4.525811e+01	6.218249e+02	0.003118	0.080544	0.146934	0.069819	0.0
perimeter_worst	3.360214e+08	1.142886e+02	5.174593e+01	7.923282e+02	1.134179e+04	0.112879	1.047413	1.954350	1.116016	0.2
area_worst	7.629681e+09	1.888227e+03	8.412838e+02	1.302615e+04	1.921926e+05	1.655299	15.323436	30.682405	17.886881	2.7
smoothness_worst	2.951016e+04	9.624625e-03	7.611070e-03	8.352553e-02	9.925135e-01	0.000259	0.000682	0.000817	0.000401	0.0
compactness_worst	-5.838341e+04	2.292492e-01	1.880100e-01	1.742478e+00	2.161660e+01	0.001045	0.007194	0.009469	0.004075	0.0
concavity_worst	6.051816e+05	3.873864e-01	2.701101e-01	2.858506e+00	3.763442e+01	0.001276	0.008994	0.014704	0.006091	0.0
concave points_worst	2.890528e+05	1.723927e-01	8.349085e-02	1.231848e+00	1.670179e+01	0.000465	0.002831	0.004513	0.002321	0.0
symmetry_worst	-3.420616e+05	3.574576e-02	2.794199e-02	2.842997e-01	3.125809e+00	0.000343	0.001667	0.002020	0.000902	0.0
fractal_dimension_worst	-6.743751e+04	4.497351e-04	9.260129e-03	2.239052e-02	2.375623e-02	0.000127	0.000656	0.000741	0.000258	0.0
Unnamed: 32	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

32 rows × 32 columns

In [4]:

```
#correlation in data

data.corr()
```

Out[4]:

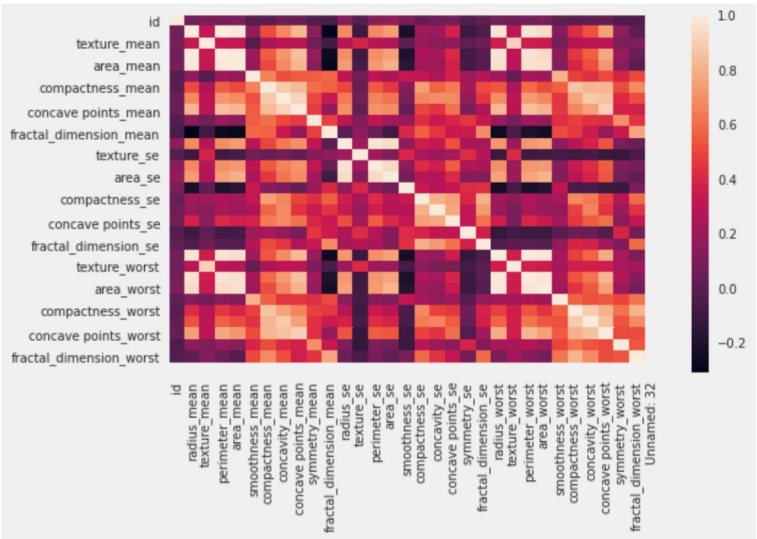
	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	...
id	1.000000	0.074626	0.099770	0.073159	0.096893	-0.012968	0.000096	0.050080	0.044158	-0.022114	...

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	...
radius_mean	0.074626	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764	0.822529	0.147741	...
texture_mean	0.099770	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702	0.302418	0.293464	0.071401	...
perimeter_mean	0.073159	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936	0.716136	0.850977	0.183027	...
area_mean	0.096893	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502	0.685983	0.823269	0.151293	...
smoothness_mean	-0.012968	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123	0.521984	0.553695	0.557775	...
compactness_mean	0.000096	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121	0.831135	0.602641	...
concavity_mean	0.050080	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000	0.921391	0.500667	...
concave points_mean	0.044158	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391	1.000000	0.462497	...
symmetry_mean	-0.022114	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.462497	1.000000	...
fractal_dimension_mean	-0.052511	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783	0.166917	0.479921	...
radius_se	0.143048	0.679090	0.275869	0.691765	0.732562	0.301467	0.497473	0.631925	0.698050	0.303379	...
texture_se	-0.007526	-0.097317	0.386358	-0.086761	-0.066280	0.068406	0.046205	0.076218	0.021480	0.128053	...
perimeter_se	0.137331	0.674172	0.281673	0.693135	0.726628	0.296092	0.548905	0.660391	0.710650	0.313893	...
area_se	0.177742	0.735864	0.259845	0.744983	0.800086	0.246552	0.455653	0.617427	0.690299	0.223970	...
smoothness_se	0.096781	-0.222600	0.006614	-0.202694	-0.166777	0.332375	0.135299	0.098564	0.027653	0.187321	...
compactness_se	0.033961	0.206000	0.191975	0.250744	0.212583	0.318943	0.738722	0.670279	0.490424	0.421659	...
concavity_se	0.055239	0.194204	0.143293	0.228082	0.207660	0.248396	0.570517	0.691270	0.439167	0.342627	...
concave points_se	0.078768	0.376169	0.163851	0.407217	0.372320	0.380676	0.642262	0.683260	0.615634	0.393298	...
symmetry_se	-0.017306	-0.104321	0.009127	-0.081629	-0.072497	0.200774	0.229977	0.178009	0.095351	0.449137	...
fractal_dimension_se	0.025725	-0.042641	0.054458	-0.005523	-0.019887	0.283607	0.507318	0.449301	0.257584	0.331786	...
radius_worst	0.082405	0.969539	0.352573	0.969476	0.962746	0.213120	0.535315	0.688236	0.830318	0.185728	...
texture_worst	0.064720	0.297008	0.912045	0.303038	0.287489	0.036072	0.248133	0.299879	0.292752	0.090651	...
perimeter_worst	0.079986	0.965137	0.358040	0.970387	0.959120	0.238853	0.590210	0.729565	0.855923	0.219169	...
area_worst	0.107187	0.941082	0.343546	0.941550	0.959213	0.206718	0.509604	0.675987	0.809630	0.177193	...
smoothness_worst	0.010338	0.119616	0.077503	0.150549	0.123523	0.805324	0.565541	0.448822	0.452753	0.426675	...
compactness_worst	-0.002968	0.413463	0.277830	0.455774	0.390410	0.472468	0.865809	0.754968	0.667454	0.473200	...
concavity_worst	0.023203	0.526911	0.301025	0.563879	0.512606	0.434926	0.816275	0.884103	0.752399	0.433721	...
concave points_worst	0.035174	0.744214	0.295316	0.771241	0.722017	0.503053	0.815573	0.861323	0.910155	0.430297	...
symmetry_worst	-0.044224	0.163953	0.105008	0.189115	0.143570	0.394309	0.510223	0.409464	0.375744	0.699826	...
fractal_dimension_worst	-0.029866	0.007066	0.119205	0.051019	0.003738	0.499316	0.687382	0.514930	0.368661	0.438413	...
Unnamed: 32	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...

32 rows × 32 columns

In [5]:

```
sns.heatmap(data.corr())
plt.show()
```



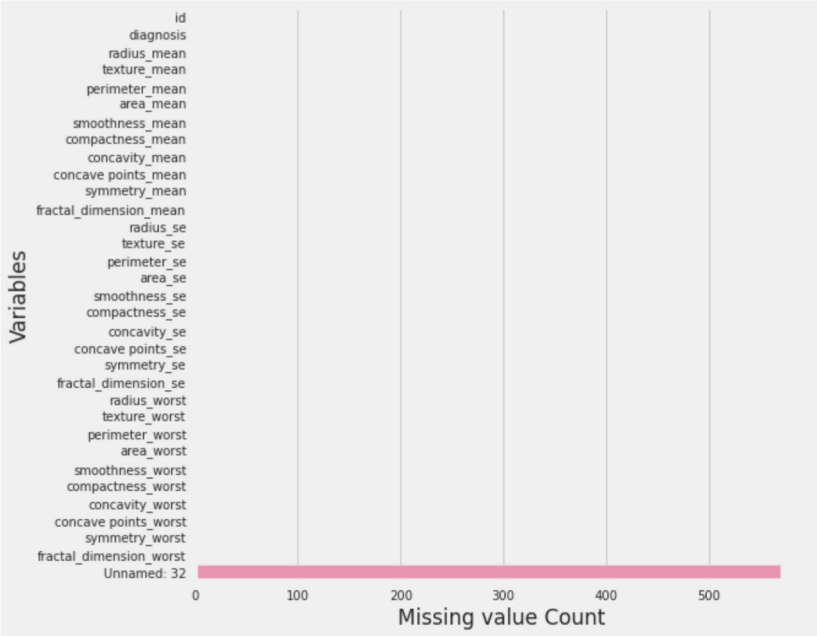
Checking for missing and duplicate values

In [6]:

```
print('Missing Values Plot')
plt.figure(figsize=(8,8))
```

```
sns.barplot(data=data.isnull().sum().reset_index(), y='index', x=0)
plt.ylabel('Variables')
plt.xlabel('Missing value Count')
plt.show()
```

Missing Values Plot



Dropping unnecessary variables

In [7]:

```
drop_var = ['Unnamed: 32', 'id']

data.drop(drop_var, axis=1, inplace=True)
```

Analyzing the features

There are a total of 10 features for each of which the safe, worst and mean values are given.

For the analysis dropping the mean values seems to be right, because it is just a new feature added from the former two.

The features are:-

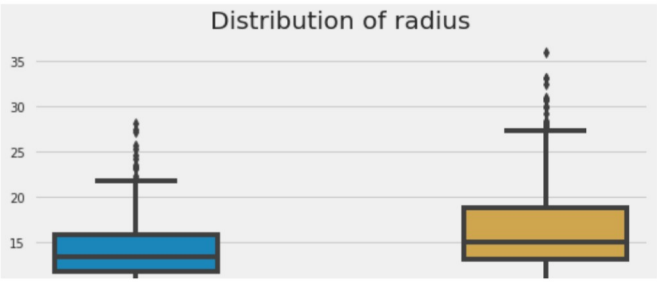
1. **radius**
2. **texture**
3. **perimeter**
4. **area**
5. **smoothness**
6. **compactness**
7. **concavity**
8. **concave points**
9. **symmetry**
10. **fractal_dimension**

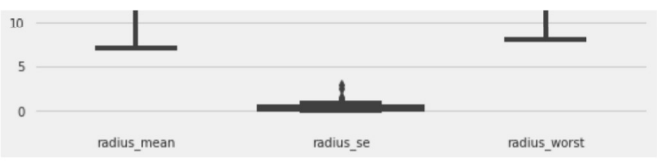
Distribution of the features

In [8]:

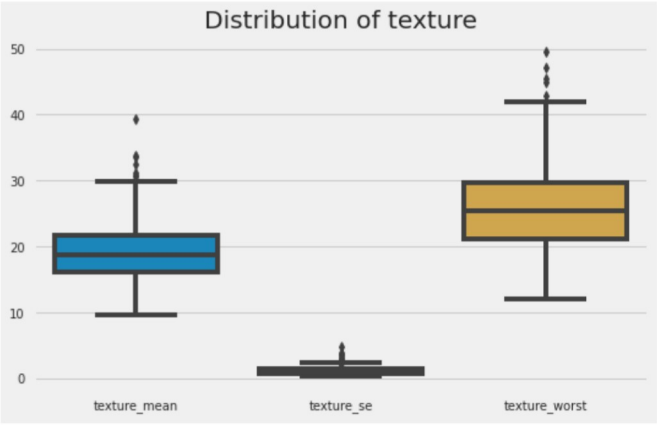
```
features = ['radius','texture','perimeter','area','smoothness','compactness','concavity','concave points','symmetry','fractal dime
nsion']

for feature in features:
    print("{} distribution".format(feature))
    sns.boxplot(data=data[['{} mean'.format(feature), '{} se'.format(feature), '{} worst'.format(feature)]])
    plt.title('Distribution of {}'.format(feature))
    plt.show()
```

radius distribution

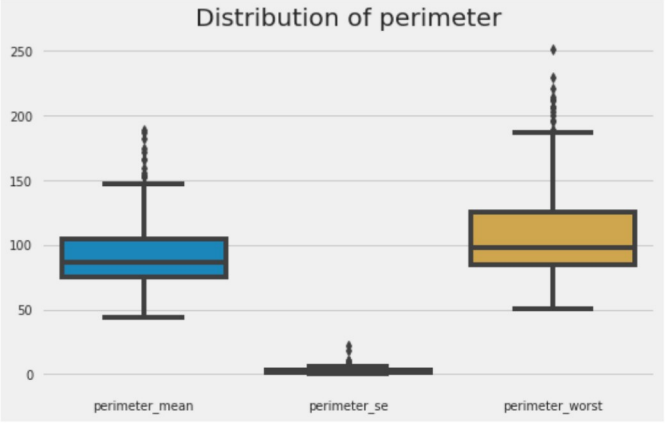




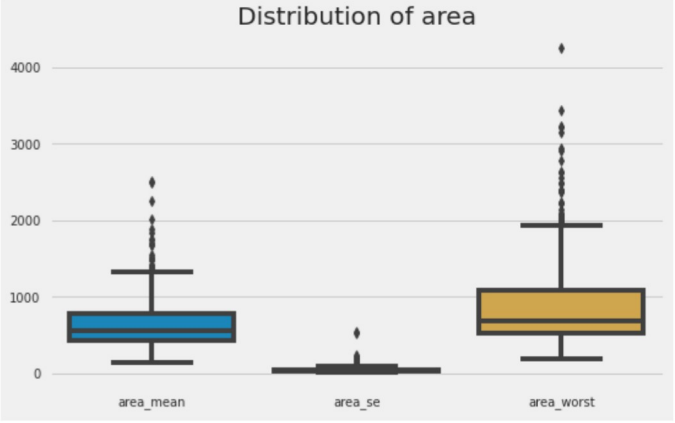
texture distribution



perimeter distribution

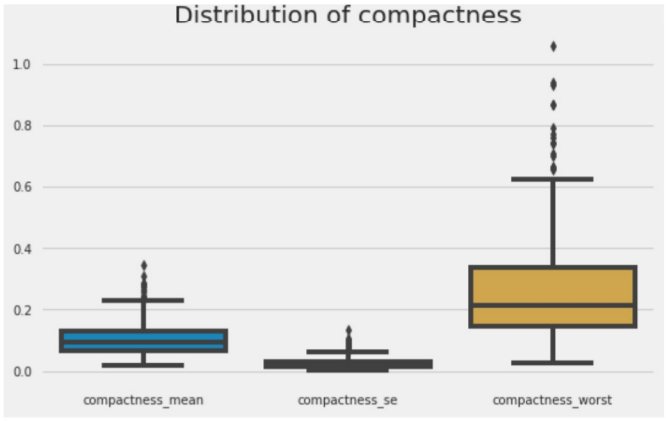


area distribution

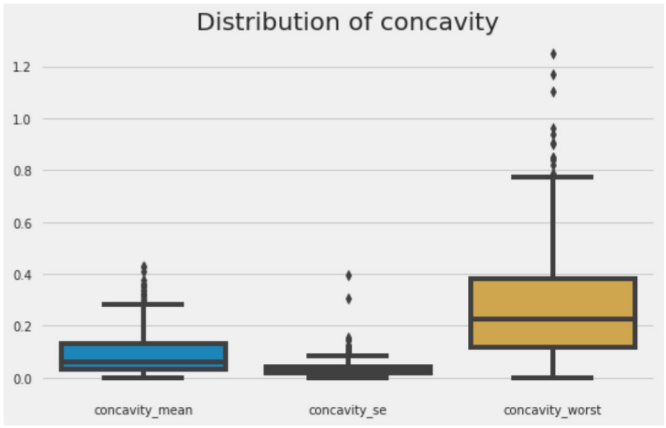


smoothness distribution

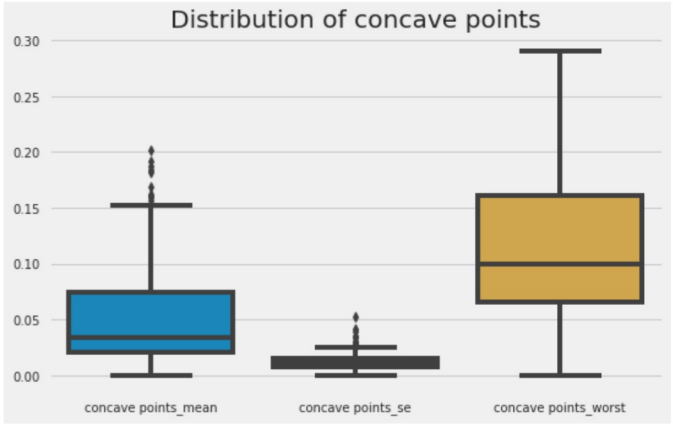
compactness distribution



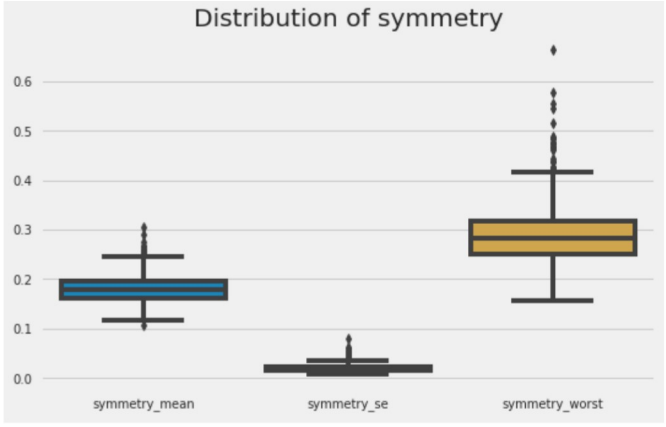
concavity distribution



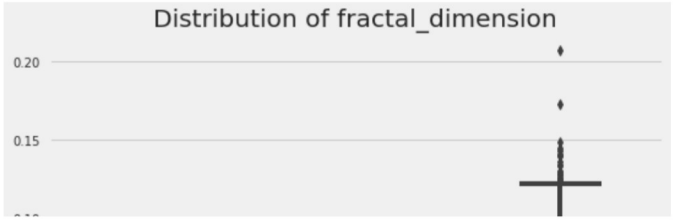
concave points distribution

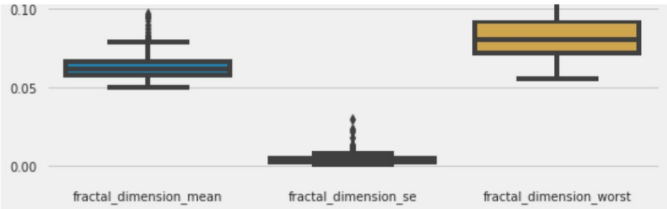


symmetry distribution



fractal_dimension distribution



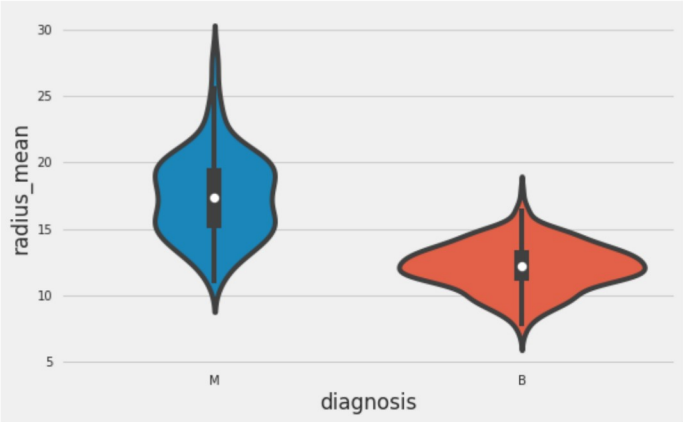


Distribution based on diagnosis

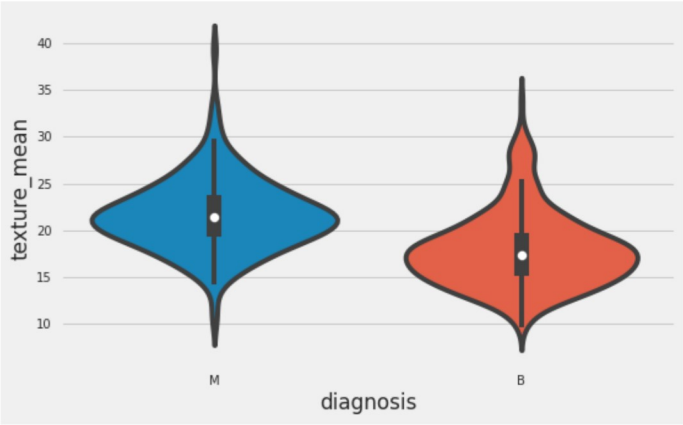
In [9]:

```
for feature in features:
    print("{} distribution based on diagnosis".format(feature))
    sns.violinplot(data=data, x="diagnosis", y="{}_mean".format(feature), size=8)
    plt.show()
```

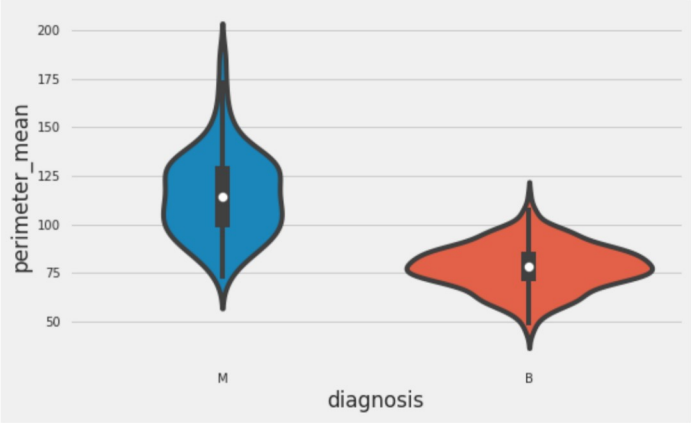
radius distribution based on diagnosis



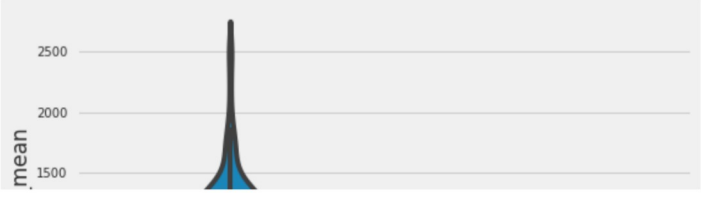
texture distribution based on diagnosis

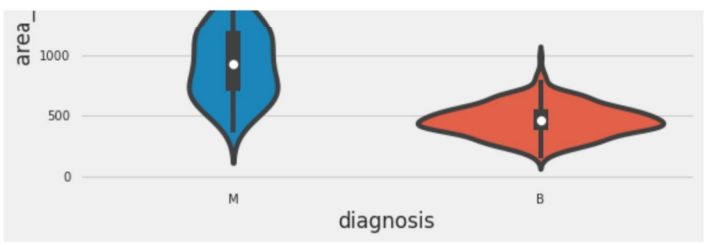


perimeter distribution based on diagnosis

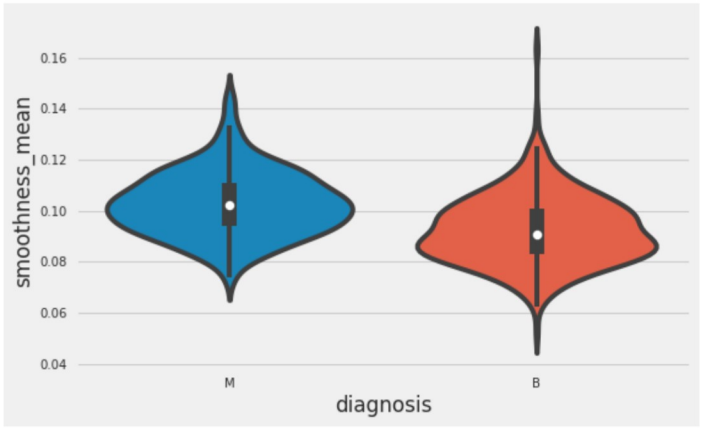


area distribution based on diagnosis

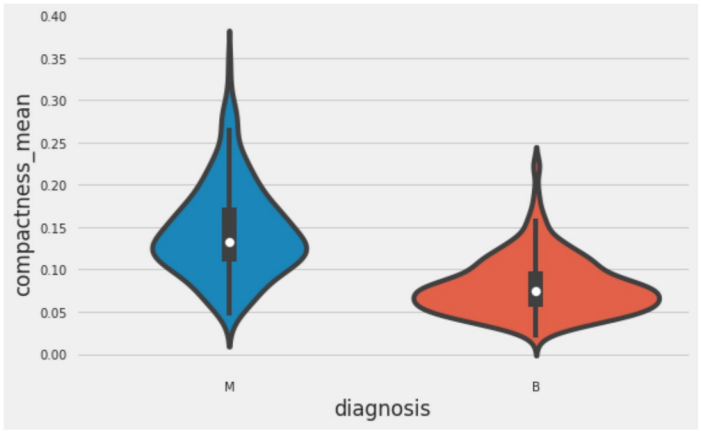




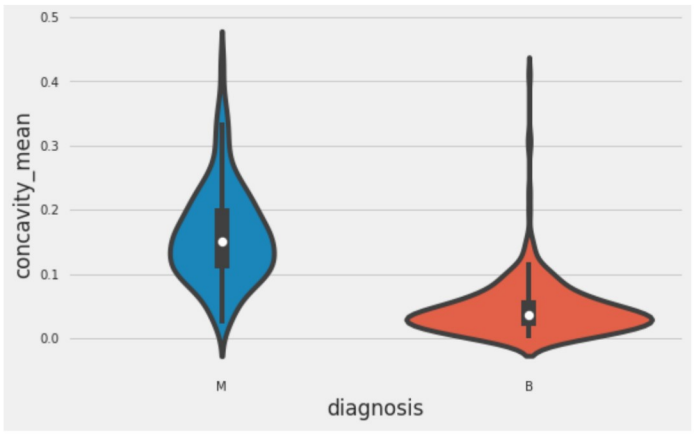
smoothness distribution based on diagnosis



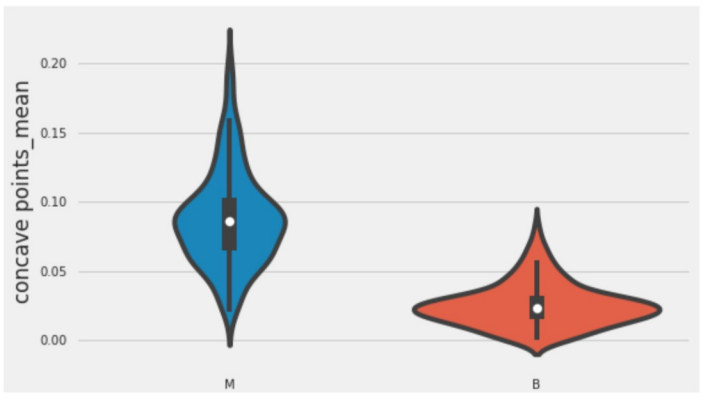
compactness distribution based on diagnosis



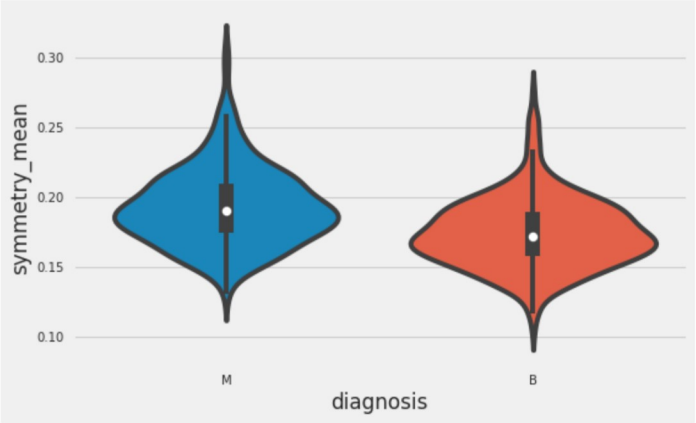
concavity distribution based on diagnosis



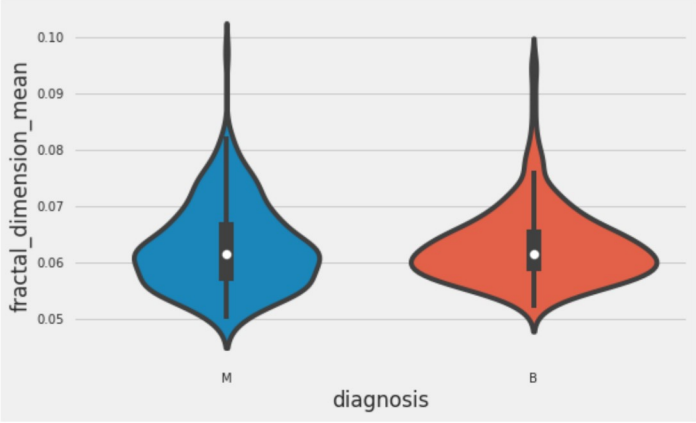
concave points distribution based on diagnosis



symmetry distribution based on diagnosis



fractal_dimension distribution based on diagnosis



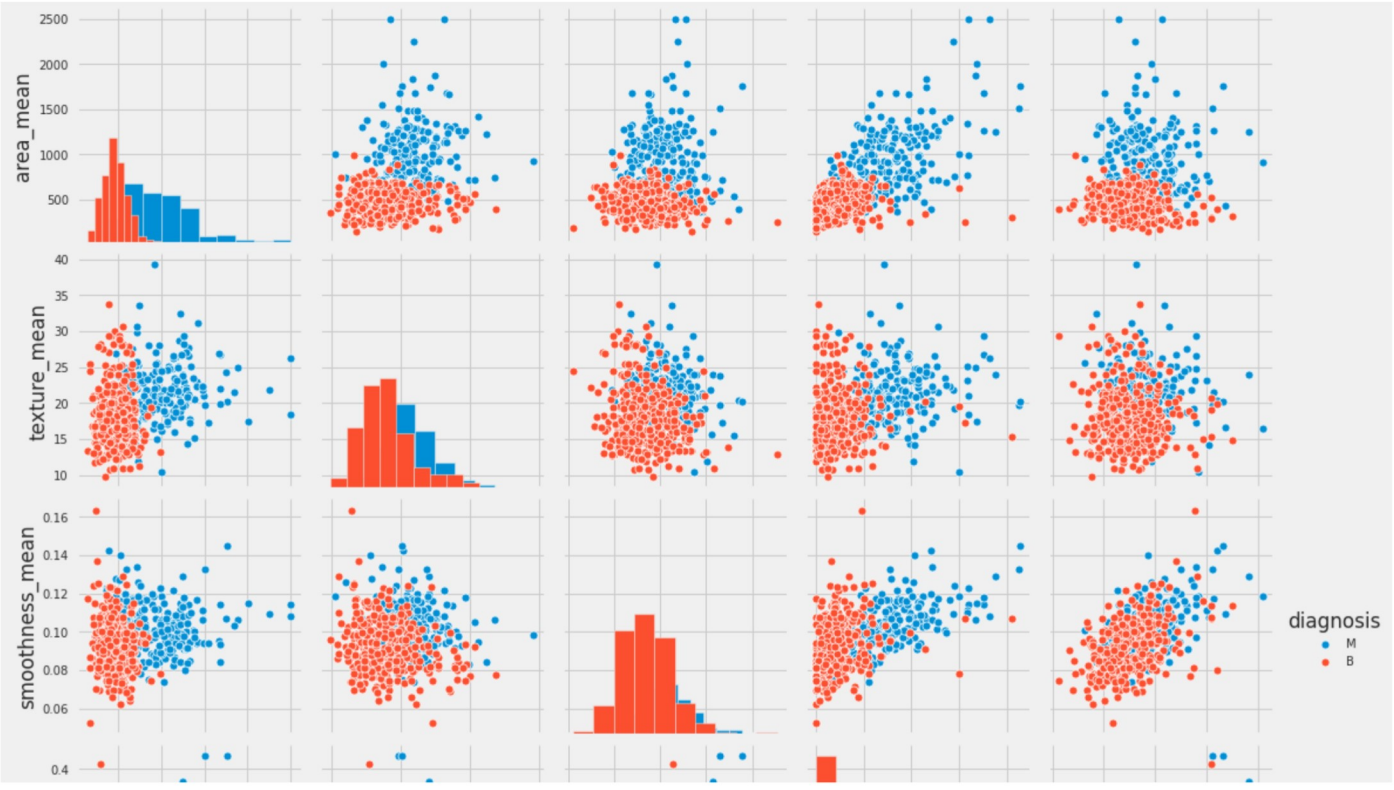
The most outliers can be noticed in the benign plot of the concavity feature.

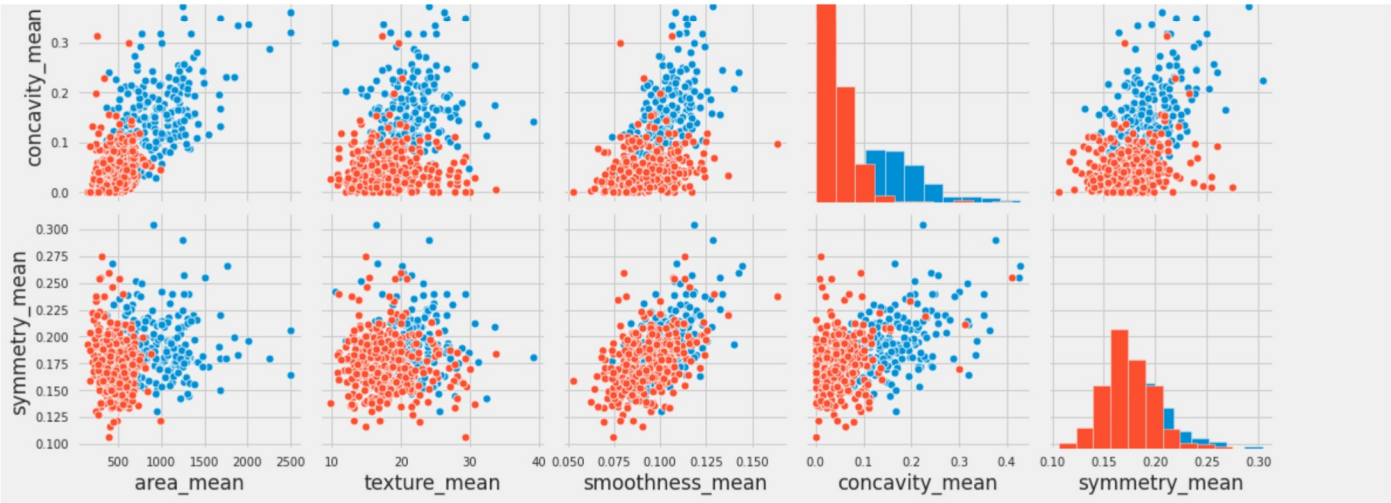
Correlation of the variables

In [10]:

```
print('Pairplot')
sns.pairplot(data=data[['diagnosis','area_mean','texture_mean','smoothness_mean','concavity_mean','symmetry_mean']], hue="diagnosis", height=3, diag_kind="hist")
plt.show()
```

Pairplot





Classification Model

In [11]:

```
#separating features and labels

X = data.drop('diagnosis',axis=1)
y = data['diagnosis']
```

In [12]:

```
#scaling the data

from sklearn import preprocessing

X_scaled = preprocessing.scale(X)
```

In [13]:

```
#splitting the data

from sklearn.model_selection import train_test_split, cross_val_score

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.4, random_state=13)
```

In [14]:

```
#creating model and fitting the data

from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train, y_train)
```

Out[14]:

```
LogisticRegression()
```

In [15]:

```
#checking the cross val score

scores = cross_val_score(model, X_scaled, y, cv=5)
print(np.mean(scores))
```

```
0.9806862288464524
```

In [16]:

```
#prediction

pred = model.predict(X_test)
```

In [17]:

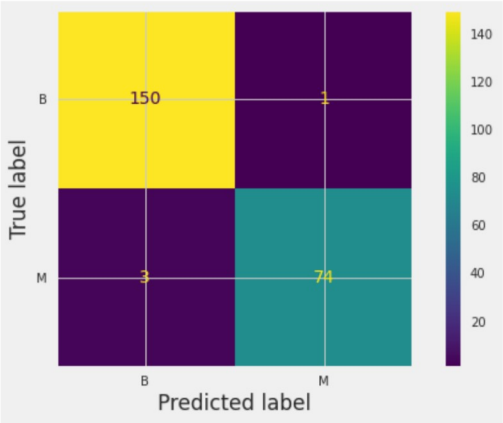
```
#checking the classificaton report

from sklearn.metrics import classification_report, confusion_matrix

print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           B       0.98      0.99      0.99       151
           M       0.99      0.96      0.97        77

    accuracy                           0.98       228
   macro avg       0.98      0.98      0.98       228
weighted avg       0.98      0.98      0.98       228
```

In [18]:

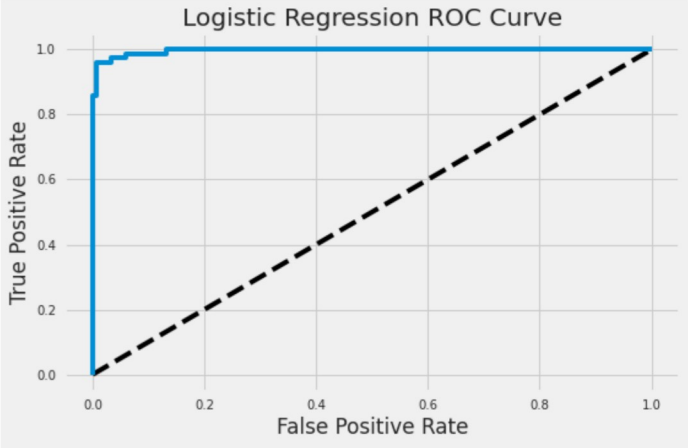
```
#confusion matrix
```

```
from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(model, X_test, y_test)
plt.show()
```



In [19]:

```
#checking roc curve

from sklearn.metrics import roc_curve

model = LogisticRegression()
model.fit(X_train, y_train)
y_pred_prob = model.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob, pos_label='M')
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label='Logistic Regression')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Logistic Regression ROC Curve')
plt.show()
```



Accuracy: 98%

Now the accuracy percentage seems to be getting overweight. But in this case the target feature has almost balanced data for each of the labels.

So overfitting should not be a problem.

Some more data would have helped in understanding. Will check if I find any similar dataset.

For now this is the end of the analysis. Do leave an upvote if you like it and let me know in the comments if there is anything wrong or how to improve it.

In [ ]: