Install Pycaret Module

```
In [37]: #!pip install pycaret[full]
         #!pip install catboost
         #!pip install xgboost
```

Import Modules

```
In [5]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        import warnings
        from pycaret.classification import *
        %matplotlib inline
        warnings.filterwarnings('ignore')
```

Load the Dataset

```
In [6]: df = pd.read_csv('./data.csv')
        df.head()
```

Out[6]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	texture_worst	perimeter_worst	area_worst	smoothness_worst	compact
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	17.33	184.60	2019.0	0.1622	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	23.41	158.80	1956.0	0.1238	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	25.53	152.50	1709.0	0.1444	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	26.50	98.87	567.7	0.2098	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	16.67	152.20	1575.0	0.1374	

5 rows × 33 columns

```
In [7]: # delete unnecessary columns
        df = df.drop(columns=['id', 'Unnamed: 32'], axis=1)
```

```
In [8]: # statistical info
        df.describe()
```

Out[8]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean	...	radius_worst	texture_worst	perimeter_worst
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798	...	16.269190	25.677223	107.261213
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	...	4.833242	6.146258	33.602542
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	...	7.930000	12.020000	50.410000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	...	13.010000	21.080000	84.110000
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	...	14.970000	25.410000	97.660000
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120	...	18.790000	29.720000	125.400000
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	...	36.040000	49.540000	251.200000

8 rows × 30 columns

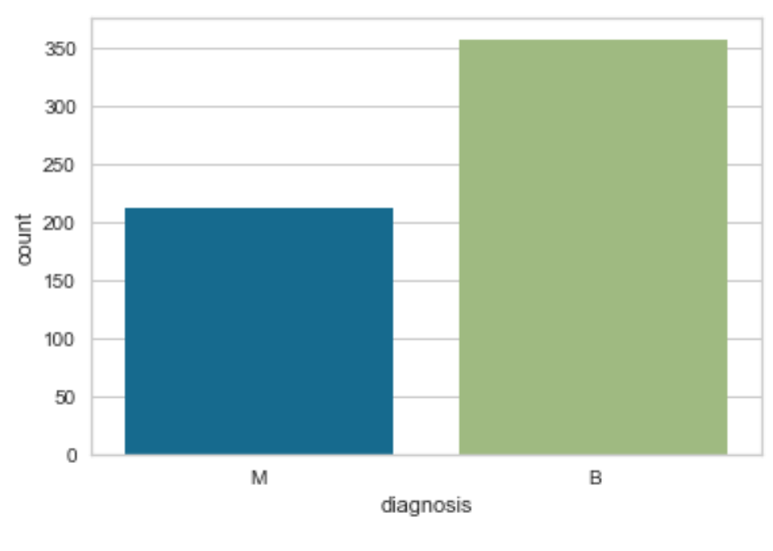
```
In [9]: # datatype info
        df.info()
```

```
<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
```

Exploratory Data Analysis

```
In [10]: sns.countplot(df['diagnosis'])
```

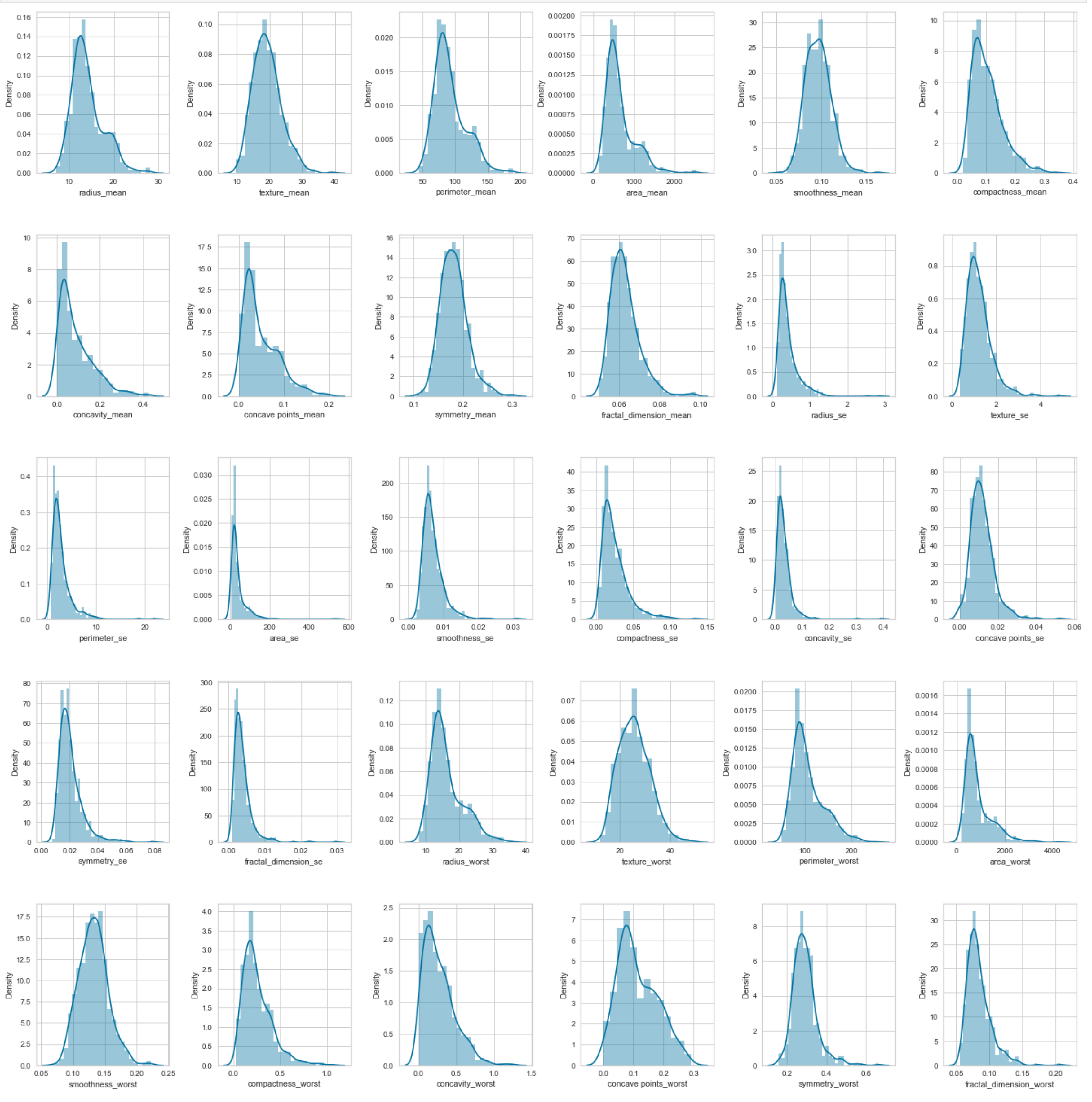
```
Out[10]: <AxesSubplot:xlabel='diagnosis', ylabel='count'>
```



```
In [11]: df_temp = df.drop(columns=['diagnosis'], axis=1)
```

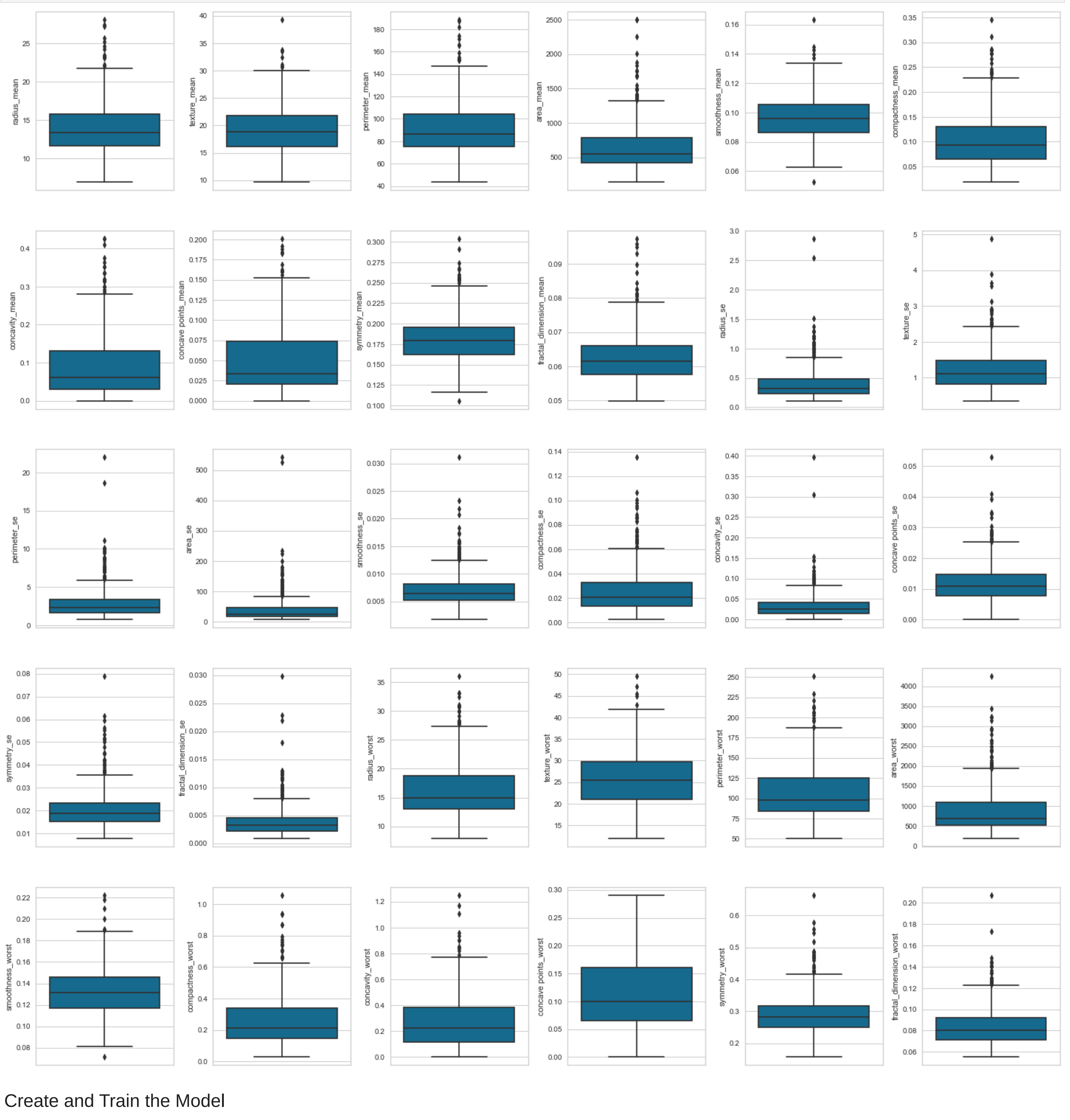
```
In [12]: # create dist plot
         fig, ax = plt.subplots(ncols=6, nrows=5, figsize=(20, 20))
         index = 0
         ax = ax.flatten()

         for col in df_temp.columns:
             sns.distplot(df[col], ax=ax[index])
             index+=1
         plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```



```
In [13]: # create box plot
         fig, ax = plt.subplots(ncols=6, nrows=5, figsize=(20, 20))
         index = 0
         ax = ax.flatten()

         for col in df_temp.columns:
             sns.boxplot(y=col, data=df, ax=ax[index])
             index+=1
         plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```



Create and Train the Model

```
In [14]: # setup the data
         clf = setup(df, target='diagnosis')
```

	Description	Value
0	Session id	3332
1	Target	diagnosis
2	Target type	Binary
3	Target mapping	B: 0, M: 1
4	Original data shape	(569, 31)
5	Transformed data shape	(569, 31)
6	Transformed train set shape	(398, 31)
7	Transformed test set shape	(171, 31)
8	Numeric features	30
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Fold Generator	StratifiedKFold
14	Fold Number	10
15	CPU Jobs	-1
16	Use GPU	False
17	Log Experiment	False
18	Experiment Name	clf-default-name
19	USI	45b4

```
In [15]: # train and test the models
         compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
ada	Ada Boost Classifier	0.9773	0.9926	0.9586	0.9812	0.9677	0.9503	0.9527	0.0570
et	Extra Trees Classifier	0.9672	0.9969	0.9314	0.9820	0.9534	0.9283	0.9320	0.1040
rf	Random Forest Classifier	0.9647	0.9950	0.9386	0.9691	0.9504	0.9232	0.9270	0.0830
gbc	Gradient Boosting Classifier	0.9647	0.9942	0.9452	0.9615	0.9509	0.9235	0.9262	0.0720
lightgbm	Light Gradient Boosting Machine	0.9647	0.9974	0.9386	0.9686	0.9510	0.9236	0.9265	0.1900
qda	Quadratic Discriminant Analysis	0.9622	0.9967	0.9590	0.9437	0.9502	0.9198	0.9212	0.0260
lda	Linear Discriminant Analysis	0.9597	0.9940	0.8981	0.9933	0.9414	0.9110	0.9157	0.0280
ridge	Ridge Classifier	0.9572	0.0000	0.8843	1.0000	0.9367	0.9048	0.9104	0.0150
lr	Logistic Regression	0.9497	0.9962	0.9257	0.9447	0.9321	0.8923	0.8961	1.0020
nb	Naive Bayes	0.9496	0.9908	0.9043	0.9600	0.9275	0.8892	0.8941	0.0200
dt	Decision Tree Classifier	0.9423	0.9400	0.9319	0.9249	0.9239	0.8776	0.8832	0.0160
knn	K Neighbors Classifier	0.9372	0.9679	0.8848	0.9475	0.9121	0.8634	0.8678	0.4230
svm	SVM - Linear Kernel	0.8672	0.0000	0.8129	0.8916	0.8219	0.7211	0.7516	0.0150
dummy	Dummy Classifier	0.6282	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0340

```
Processing:   0%|          | 0/61 [00:00<?, ?it/s]
Out[15]: AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0,
                            n_estimators=50, random_state=3332)
```

```
In [34]: # select the best model
         model = create_model('lr')
         #Linear Regression
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9500	1.0000	0.8667	1.0000	0.9286	0.8904	0.8958
1	0.9500	0.9973	1.0000	0.8824	0.9375	0.8961	0.9010
2	0.9750	1.0000	0.9333	1.0000	0.9655	0.9459	0.9473
3	0.9250	0.9920	0.9333	0.8750	0.9032	0.8421	0.8433
4	0.9750	0.9973	0.9333	1.0000	0.9655	0.9459	0.9473
5	0.9500	1.0000	0.8667	1.0000	0.9286	0.8904	0.8958
6	0.9500	1.0000	0.8667	1.0000	0.9286	0.8904	0.8958
7	0.9250	0.9920	1.0000	0.8333	0.9091	0.8462	0.8563
8	0.9231	0.9829	0.8571	0.9231	0.8889	0.8302	0.8315
9	0.9744	1.0000	1.0000	0.9333	0.9655	0.9451	0.9466
Mean	0.9497	0.9962	0.9257	0.9447	0.9321	0.8923	0.8961
Std	0.0195	0.0054	0.0565	0.0609	0.0258	0.0413	0.0404

```
Processing:   0%|          | 0/4 [00:00<?, ?it/s]
```

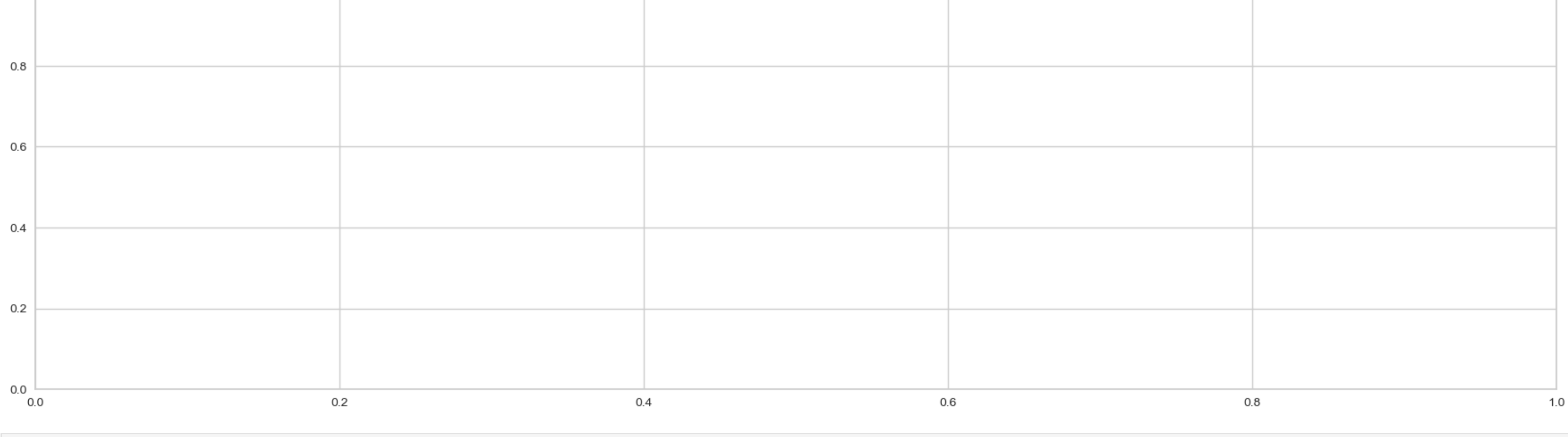
```
In [33]: # hyperparameter tuning
         best_model = tune_model(model)
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9500	0.9947	0.8667	1.0000	0.9286	0.8904	0.8958
1	0.9500	0.9947	1.0000	0.8824	0.9375	0.8961	0.9010
2	0.9750	0.9973	0.9333	1.0000	0.9655	0.9459	0.9473
3	0.9500	0.9947	0.9333	0.9333	0.9333	0.8933	0.8933
4	0.9750	1.0000	0.9333	1.0000	0.9655	0.9459	0.9473
5	0.9500	1.0000	0.8667	1.0000	0.9286	0.8904	0.8958
6	0.9750	1.0000	0.9333	1.0000	0.9655	0.9459	0.9473
7	0.9250	0.9947	1.0000	0.8333	0.9091	0.8462	0.8563
8	0.9231	0.9714	0.8571	0.9231	0.8889	0.8302	0.8315
9	0.9744	1.0000	1.0000	0.9333	0.9655	0.9451	0.9466
Mean	0.9547	0.9947	0.9324	0.9505	0.9388	0.9030	0.9062
Std	0.0190	0.0081	0.0529	0.0565	0.0254	0.0404	0.0390

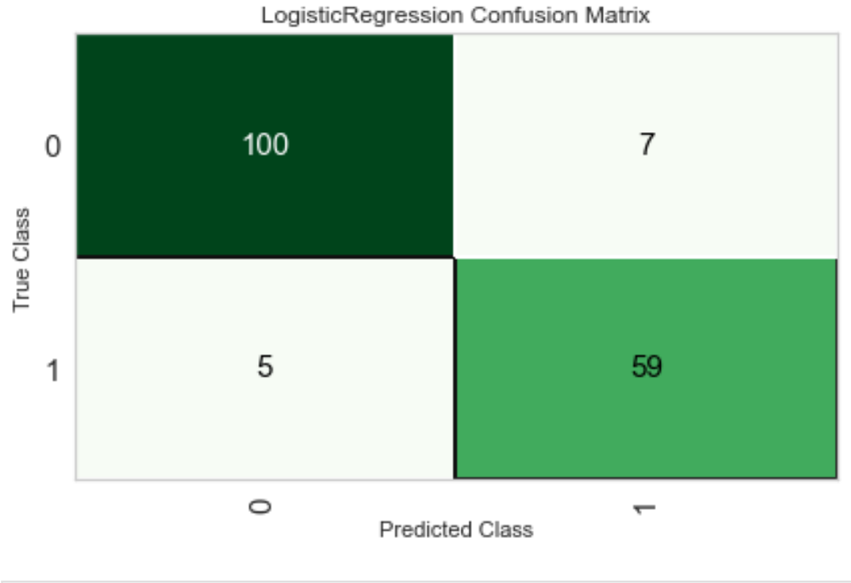
```
Processing:   0%|          | 0/7 [00:00<?, ?it/s]
Fitting 10 folds for each of 10 candidates, totalling 100 fits
```

```
In [35]: evaluate_model(best_model)
```

```
interactive(children=(ToggleButtons(description='Plot Type:', icons=('',), options=(('Pipeline Plot', 'pipelin…
```



```
In [36]: # plot the results
         plot_model(estimator=best_model, plot='confusion_matrix')
```



```
In [ ]:
```