In [ ]:
```
# import libraries
import pandas as pd # for data manupulation or analysis
import numpy as np # for numeric calculation
import matplotlib.pyplot as plt # for data visualization
import seaborn as sns # for data visualization
```

In [ ]:
```
#Load breast cancer dataset
from sklearn.datasets import load_breast_cancer
cancer_dataset = load_breast_cancer()
type(cancer_dataset)
```

Out[ ]:
```
sklearn.utils._bunch.Bunch
```

In [ ]:
```
# keys in dataset
cancer_dataset.keys()
```

Out[ ]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_
names', 'filename', 'data_module'])
```

In [ ]:
```
# featurs of each cells in numeric format
cancer_dataset['data']
```

Out[ ]:
```
array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
        1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
        8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
        8.758e-02],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
        7.820e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
        1.240e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
        7.039e-02]])
```

In [ ]:
```
# malignant or benign value
cancer_dataset['target']
```

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

In [ ]:
```
# target value name malignant or benign tumor
cancer_dataset['target_names']
```

Out[ ]:
```
array(['malignant', 'benign'], dtype='<U9')
```

In [ ]:
```
# description of data
print(cancer_dataset['DESCR'])
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimete
r)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
```

        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of the thre
e
        worst/largest values) of these features were computed for each ima
ge,
        resulting in 30 features. For instance, field 0 is Mean Radius, f
ield
        10 is Radius SE, field 20 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign

    :Summary Statistics:

    ===================================== ====== ======
                                           Min   Max
    ===================================== ====== ======
    radius (mean):   6.981 28.11
    texture (mean):   9.71   39.28
    perimeter (mean):   43.79 188.5
    area (mean):   143.5 2501.0
    smoothness (mean):   0.053 0.163
    compactness (mean):   0.019 0.345
    concavity (mean):   0.0   0.427
    concave points (mean):   0.0   0.201
    symmetry (mean):   0.106 0.304
    fractal dimension (mean):   0.05   0.097
    radius (standard error):   0.112 2.873
    texture (standard error):   0.36   4.885
    perimeter (standard error):   0.757 21.98
    area (standard error):   6.802 542.2
    smoothness (standard error):   0.002 0.031
    compactness (standard error):   0.002 0.135
    concavity (standard error):   0.0   0.396
    concave points (standard error):   0.0   0.053
    symmetry (standard error):   0.008 0.079
    fractal dimension (standard error):   0.001 0.03
    radius (worst):   7.93   36.04
    texture (worst):   12.02 49.54
    perimeter (worst):   50.41 251.2
    area (worst):   185.2 4254.0
    smoothness (worst):   0.071 0.223
    compactness (worst):   0.027 1.058
    concavity (worst):   0.0   1.252
    concave points (worst):   0.0   0.291
    symmetry (worst):   0.156 0.664
    fractal dimension (worst):   0.055 0.208
    ===================================== ====== ======

    :Missing Attribute Values: None

    :Class Distribution: 212 - Malignant, 357 - Benign

    :Creator: Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

```
    :Donor: Nick Street

    :Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle
aspirate (FNA) of a breast mass.  They describe
characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using
Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

|details-start|
**References**
|details-split|

- W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extracti
on
  for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
  Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
  San Jose, CA, 1993.
- O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis
and
  prognosis via linear programming. Operations Research, 43(4), pages 570-
577,
  July-August 1995.
- W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techni
ques
  to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77
(1994)
  163-171.

|details-end|
```

In [ ]:
```
# name of features
print(cancer_dataset['feature_names'])
```

```
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
```

In [ ]:
```
# location/path of data file
print(cancer_dataset['filename'])
```

```
breast_cancer.csv
```

In [ ]:
```
# create datafrmae
cancer_df = pd.DataFrame(np.c_[cancer_dataset['data'],cancer_dataset['tar
             columns = np.append(cancer_dataset['feature_names'], ['targe
```

In [ ]:
```
# Head of cancer DataFrame
cancer_df.head(6)
```

Out[ ]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	me conca poi
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.147
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.070
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.127
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.105
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.104
5	12.45	15.70	82.57	477.1	0.12780	0.17000	0.1578	0.080

6 rows × 31 columns

In [ ]:
```
# Tail of cancer DataFrame
cancer_df.tail(6)
```

Out[ ]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	cor p
563	20.92	25.09	143.00	1347.0	0.10990	0.22360	0.31740	0.1
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.1
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.0
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.0
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.1
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.0

6 rows × 31 columns

In [ ]:
```
# Information of cancer Dataframe
cancer_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  target                   569 non-null    float64
dtypes: float64(31)
memory usage: 137.9 KB
```

In [ ]:
```
# Numerical distribution of data
cancer_df.describe()
```

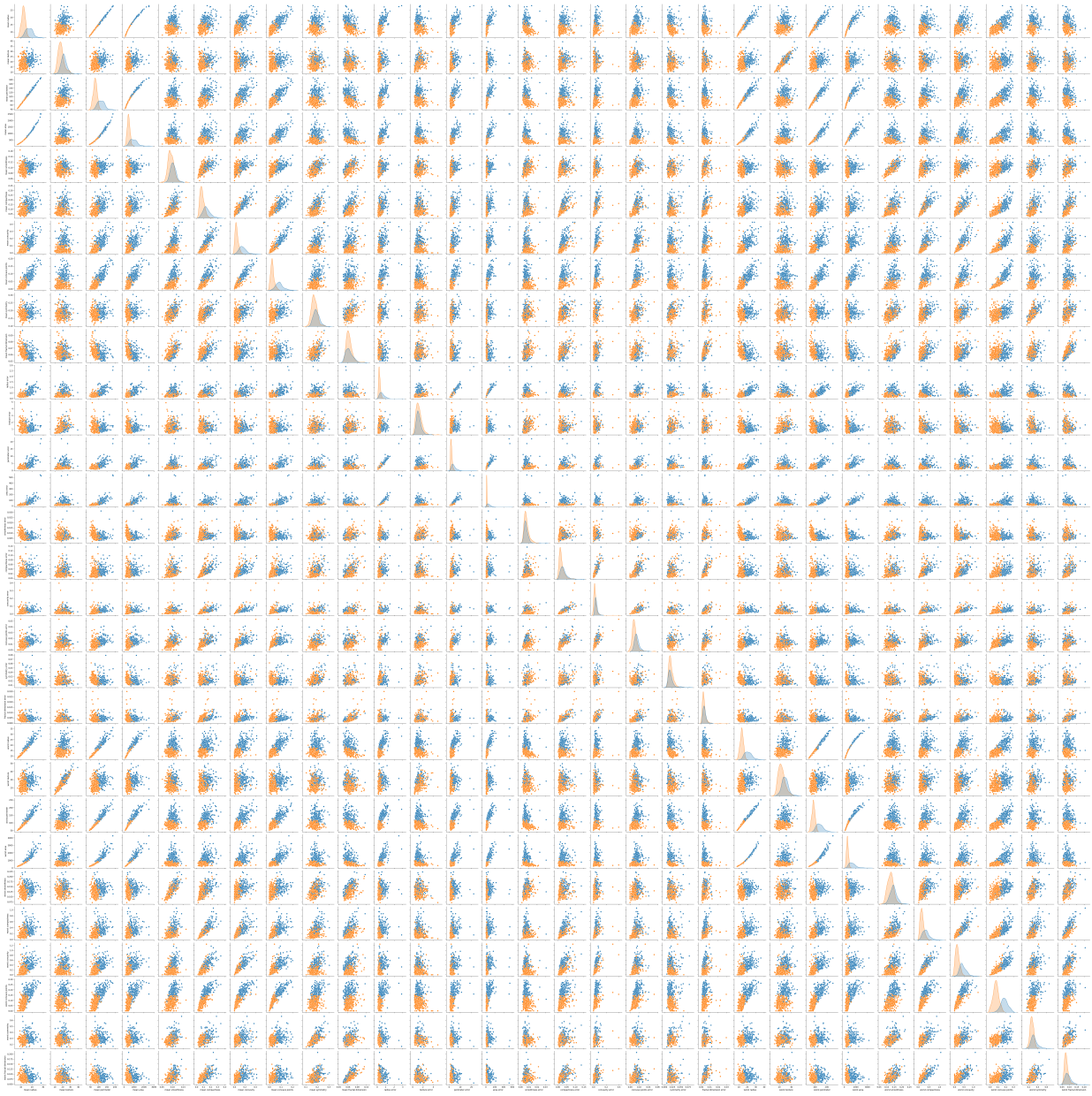
Out[ ]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	me compactn
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000(
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104:
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019:
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345

8 rows × 31 columns

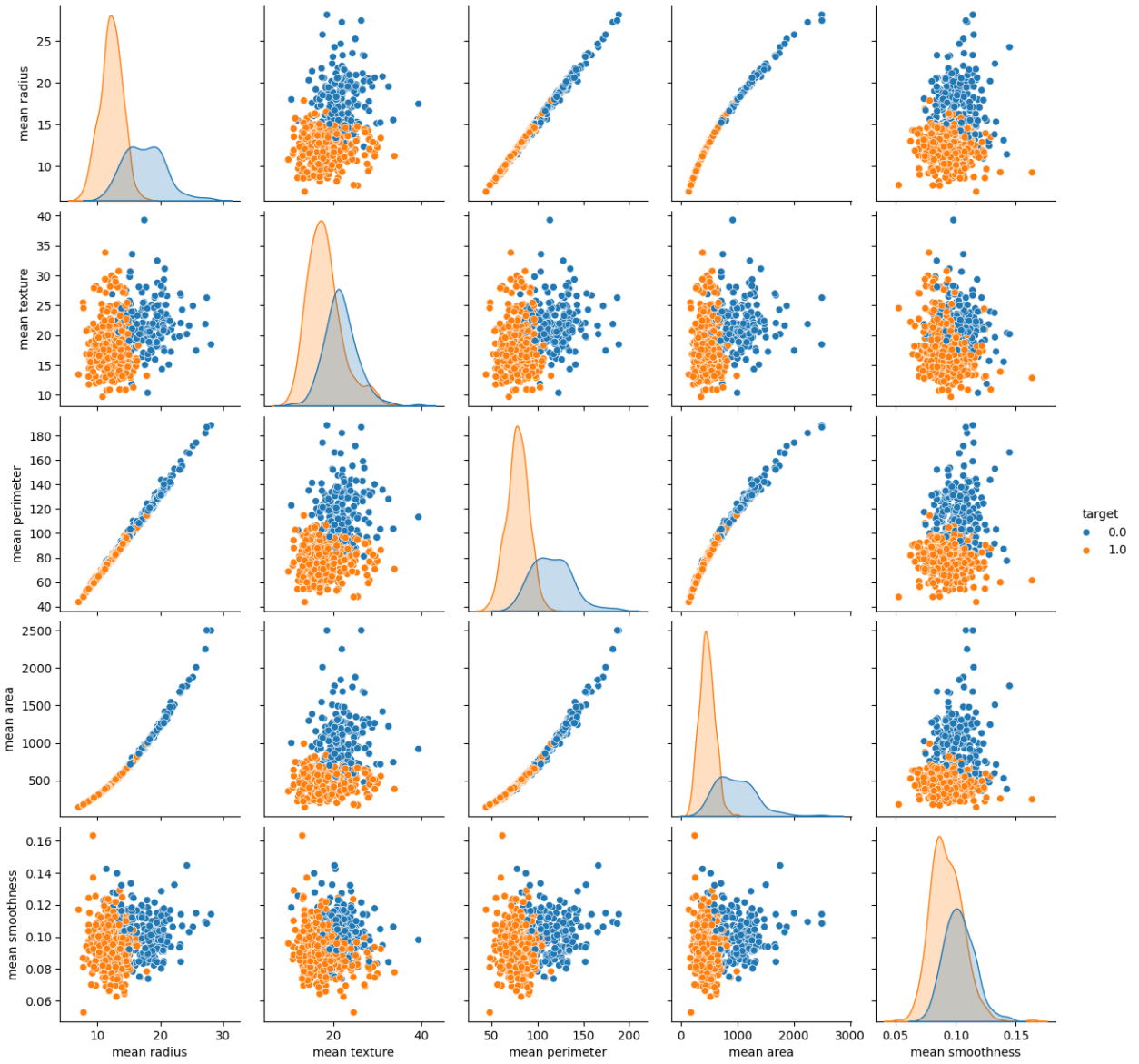
In [ ]:
```
# Paiplot of cancer dataframe
sns.pairplot(cancer_df, hue = 'target')
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x15a13d8d0>
```



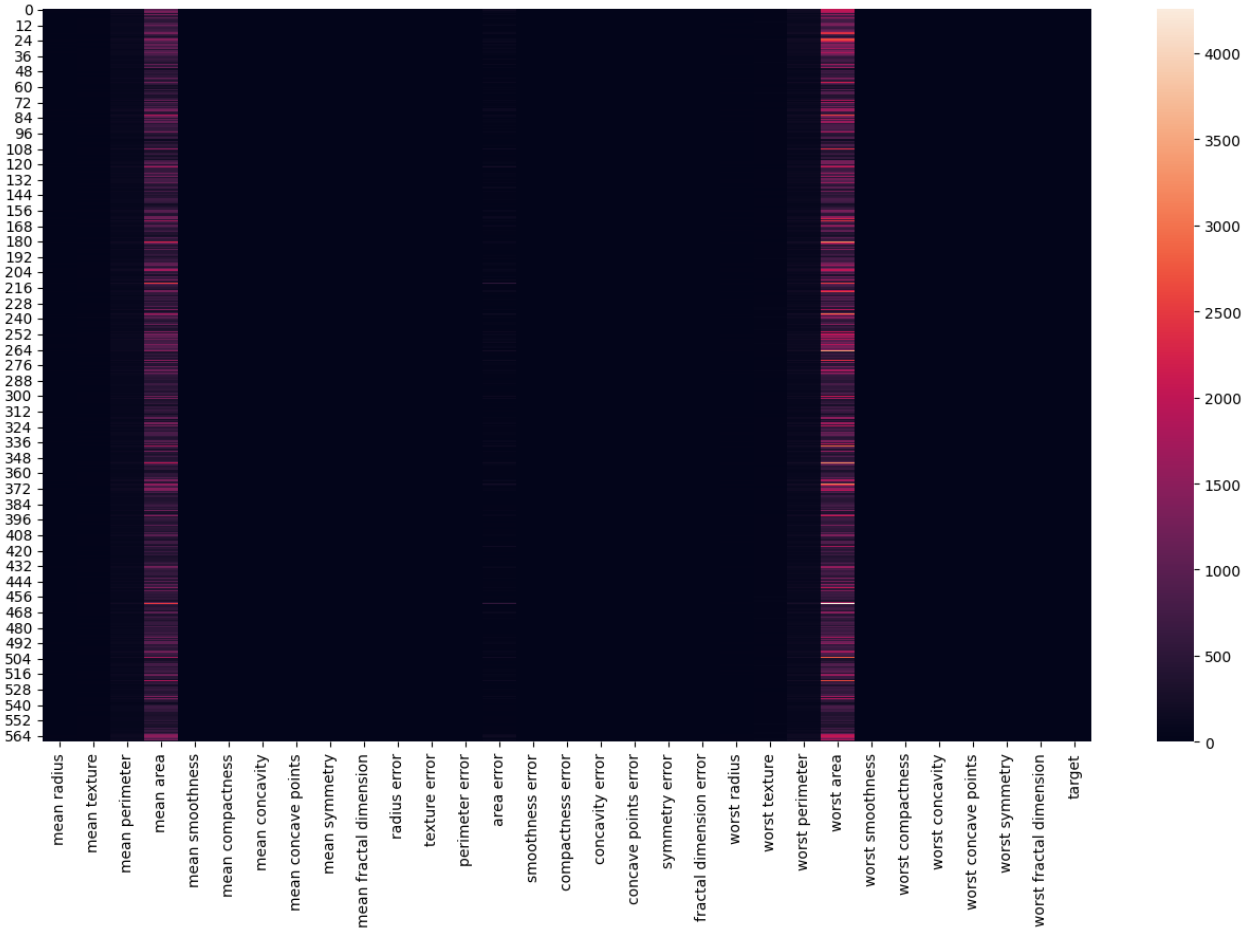
In [ ]:
```
# pair plot of sample feature
sns.pairplot(cancer_df, hue = 'target',
                vars = ['mean radius', 'mean texture', 'mean perimeter', 'me
```

Out[ ]: <seaborn.axisgrid.PairGrid at 0x32f5da790>



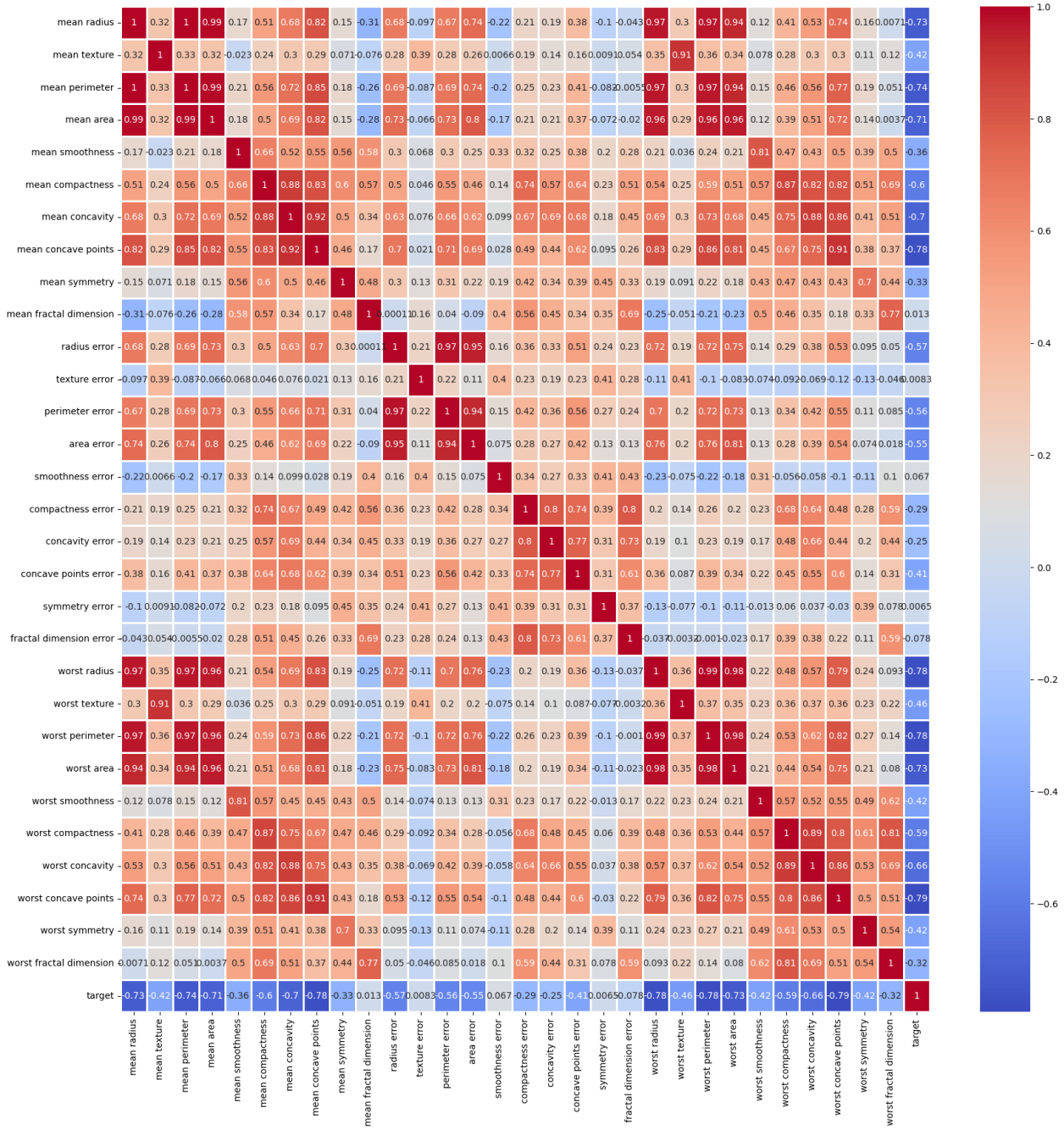
In [ ]:
```
# heatmap of DataFrame
plt.figure(figsize=(16,9))
sns.heatmap(cancer_df)
```

Out[ ]: <Axes: >



In [ ]:
```
# Heatmap of Correlation matrix of breast cancer DataFrame
plt.figure(figsize=(20,20))
sns.heatmap(cancer_df.corr(), annot = True, cmap ='coolwarm', linewidths=
```

Out[ ]: <Axes: >



In [ ]:
```
# create second DataFrame by droping target
cancer_df2 = cancer_df.drop(['target'], axis = 1)
print("The shape of 'cancer_df2' is : ", cancer_df2.shape)
cancer_df2.head(10)
```

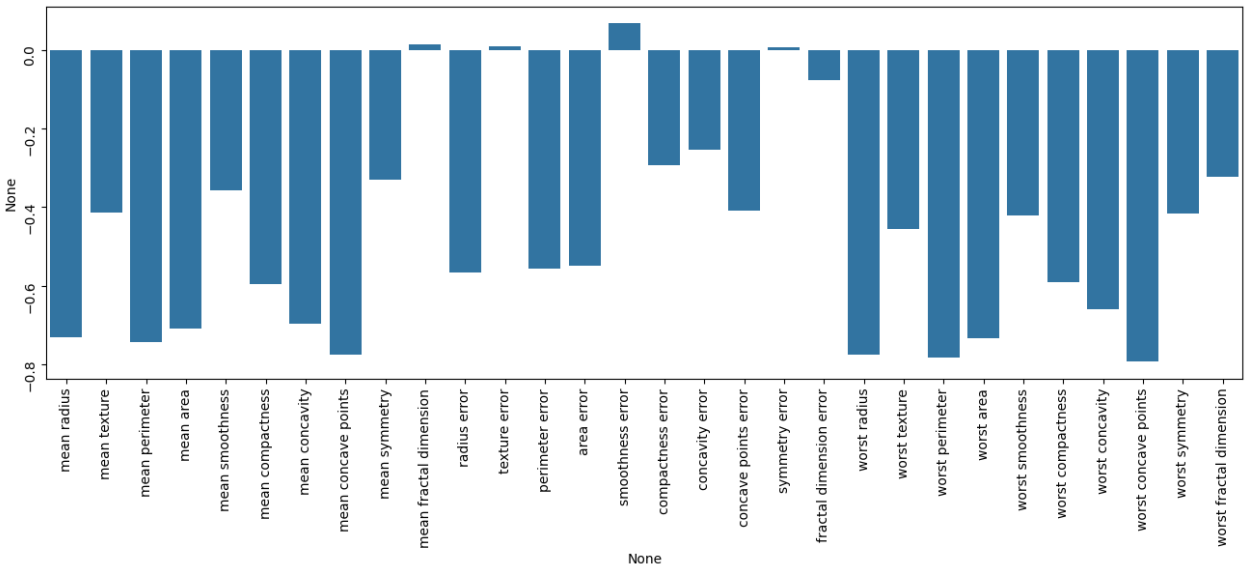
```
The shape of 'cancer_df2' is :  (569, 30)
```

Out[ ]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	me conca poi
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.147
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.070
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.127
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.105
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.104
5	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15780	0.080
6	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.11270	0.074
7	13.71	20.83	90.20	577.9	0.11890	0.16450	0.09366	0.059
8	13.00	21.82	87.50	519.8	0.12730	0.19320	0.18590	0.093
9	12.46	24.04	83.97	475.9	0.11860	0.23960	0.22730	0.085

10 rows × 30 columns

In [ ]:
```
# visualize correlation barplot
plt.figure(figsize = (16,5))
ax = sns.barplot(x = cancer_df2.corrwith(cancer_df.target).index,
                 y = cancer_df2.corrwith(cancer_df.target))
ax.tick_params(labelrotation = 90)
```



In [ ]:
```
# input variable
X = cancer_df.drop(['target'], axis = 1)
X.head(6)
```

Out[ ]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	me conca poi
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.147
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.070
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.127
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.105
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.104
5	12.45	15.70	82.57	477.1	0.12780	0.17000	0.1578	0.080

6 rows × 30 columns

In [ ]:
```
# output variable
y = cancer_df['target']
y.head(6)
```

Out[ ]:
```
0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
5    0.0
Name: target, dtype: float64
```

In [ ]:
```
# split dataset into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2
```

In [ ]:
```
# Feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train_sc = sc.fit_transform(X_train)
X_test_sc = sc.transform(X_test)
```

In [ ]:
```
from sklearn.metrics import confusion_matrix, classification_report, accu
```

Support Vector Machine

In [ ]:
```
# Support vector classifier
from sklearn.svm import SVC
svc_classifier = SVC()
svc_classifier.fit(X_train, y_train)
y_pred_scv = svc_classifier.predict(X_test)
accuracy_score(y_test, y_pred_scv)
```

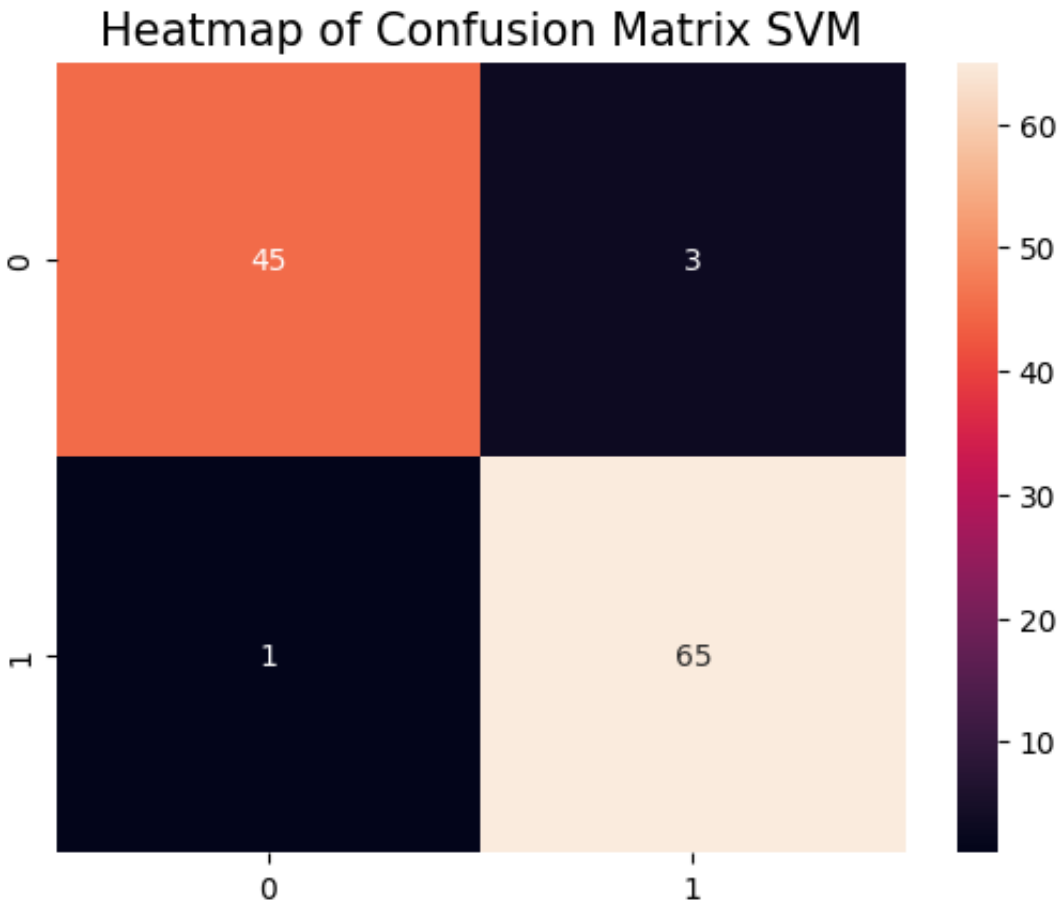
Out[ ]:
```
0.9385964912280702
```

In [ ]:
```
# Train with Standard scaled Data
svc_classifier2 = SVC()
```

```
svc_classifier2.fit(X_train_sc, y_train)
y_pred_svc_sc = svc_classifier2.predict(X_test_sc)

accuracy=accuracy_score(y_test, y_pred_svc_sc)
print("="*30)
print(f"Accuracy Score: {accuracy:.4f}")
print("="*30)
cm = confusion_matrix(y_test, y_pred_svc_sc)
plt.title('Heatmap of Confusion Matrix SVM', fontsize = 15)
sns.heatmap(cm, annot = True)
plt.show()
```

```
==============================
Accuracy Score: 0.9649
==============================
```



Logistic Regression

In [ ]:
```
# Logistic Regression
from sklearn.linear_model import LogisticRegression
lr_classifier = LogisticRegression(random_state = 51, penalty = 'l2')
lr_classifier.fit(X_train, y_train)
y_pred_lr = lr_classifier.predict(X_test)
accuracy_score(y_test, y_pred_lr)
```

```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-pac
kages/sklearn/linear_model/_logistic.py:460: ConvergenceWarning: lbfgs fai
led to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  n_iter_i = _check_optimize_result(
```

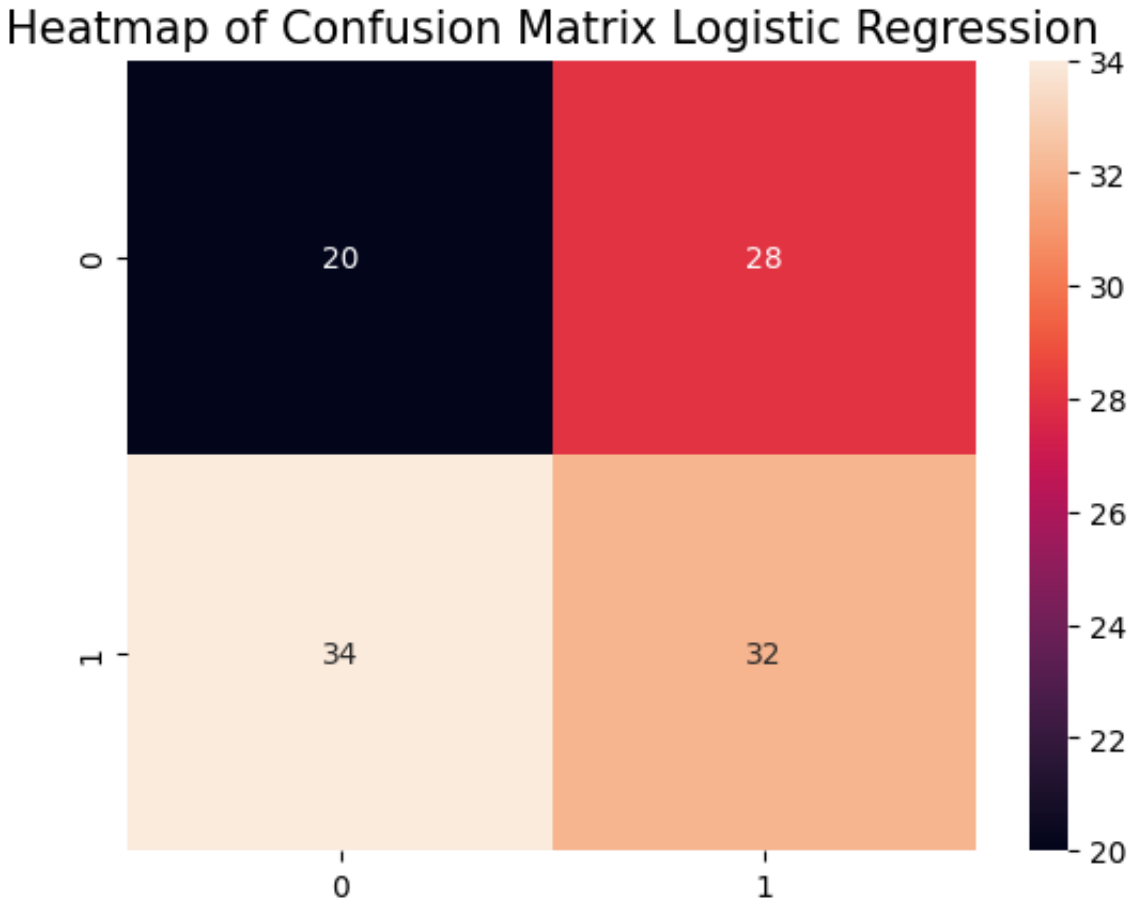
Out[ ]: 0.956140350877193

In [ ]:
```
# Train with Standard scaled Data
lr_classifier2 = LogisticRegression(random_state = 51, penalty = 'l2')
lr_classifier2.fit(X_train_sc, y_train)
y_pred_lr_sc = lr_classifier.predict(X_test_sc)
accuracy=accuracy_score(y_test, y_pred_lr_sc)

print("="*30)
print(f"Accuracy Score: {accuracy:.4f}")
print("="*30)
cm = confusion_matrix(y_test, y_pred_lr_sc)
plt.title('Heatmap of Confusion Matrix Logistic Regression', fontsize = 1
sns.heatmap(cm, annot = True)
plt.show()
```

```
==============================
Accuracy Score: 0.4561
==============================
```

```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-pac
kages/sklearn/base.py:465: UserWarning: X does not have valid feature name
s, but LogisticRegression was fitted with feature names
  warnings.warn(
```



KNN

In [ ]:
```
# K – Nearest Neighbor Classifier
from sklearn.neighbors import KNeighborsClassifier
knn_classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowsk
knn_classifier.fit(X_train, y_train)
y_pred_knn = knn_classifier.predict(X_test)
accuracy_score(y_test, y_pred_knn)
```

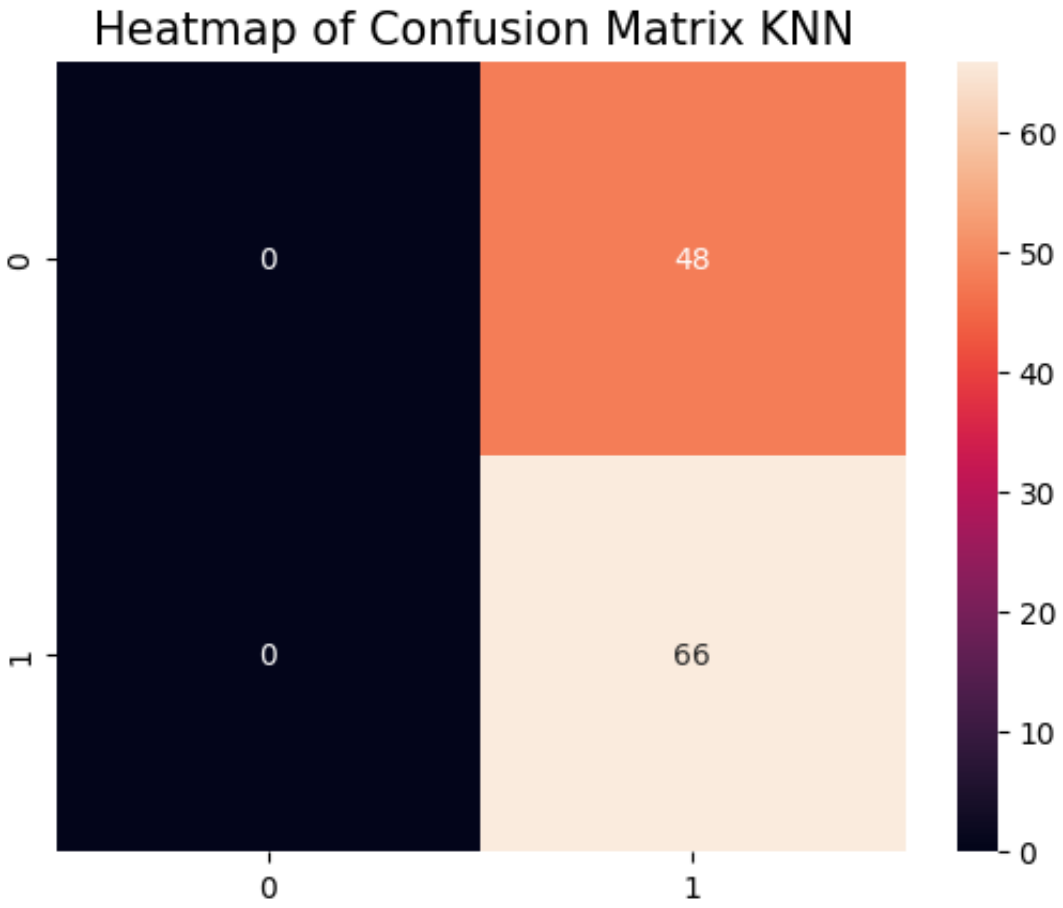
Out[ ]: 0.9385964912280702

In [ ]:
```
# Train with Standard scaled Data
knn_classifier2 = KNeighborsClassifier(n_neighbors = 5, metric = 'minkows
knn_classifier2.fit(X_train_sc, y_train)
y_pred_knn_sc = knn_classifier.predict(X_test_sc)
accuracy=accuracy_score(y_test, y_pred_knn_sc)

print("="*30)
print(f"Accuracy Score: {accuracy:.4f}")
print("="*30)
cm = confusion_matrix(y_test, y_pred_knn_sc)
plt.title('Heatmap of Confusion Matrix KNN ', fontsize = 15)
sns.heatmap(cm, annot = True)
plt.show()
```

```
==============================
Accuracy Score: 0.5789
==============================
```

```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-pac
kages/sklearn/base.py:465: UserWarning: X does not have valid feature name
s, but KNeighborsClassifier was fitted with feature names
  warnings.warn(
```



Naive Bayes

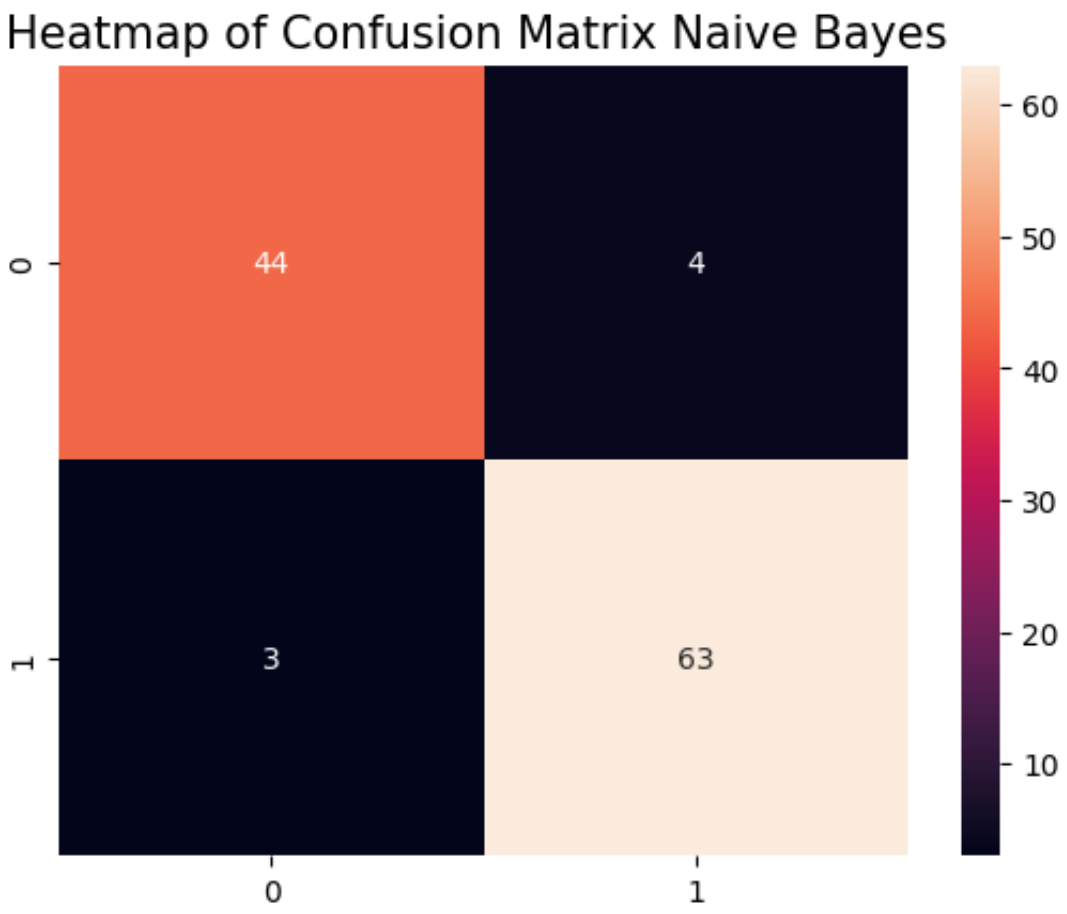
In [ ]:
```
# Naive Bayes Classifier
from sklearn.naive_bayes import GaussianNB
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
y_pred_nb = nb_classifier.predict(X_test)
accuracy_score(y_test, y_pred_nb)
```

Out[ ]:
```
0.9473684210526315
```

In [ ]:
```
# Train with Standard scaled Data
nb_classifier2 = GaussianNB()
nb_classifier2.fit(X_train_sc, y_train)
y_pred_nb_sc = nb_classifier2.predict(X_test_sc)
accuracy=accuracy_score(y_test, y_pred_nb_sc)

print("="*30)
print(f"Accuracy Score: {accuracy:.4f}")
print("="*30)
cm = confusion_matrix(y_test, y_pred_nb_sc)
plt.title('Heatmap of Confusion Matrix Naive Bayes ', fontsize = 15)
sns.heatmap(cm, annot = True)
plt.show()
```

```
==============================
Accuracy Score: 0.9386
==============================
```



Decision Tree

In [ ]:
```
# Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
dt_classifier = DecisionTreeClassifier(criterion = 'entropy', random_stat
dt_classifier.fit(X_train, y_train)
y_pred_dt = dt_classifier.predict(X_test)
accuracy_score(y_test, y_pred_dt)
```

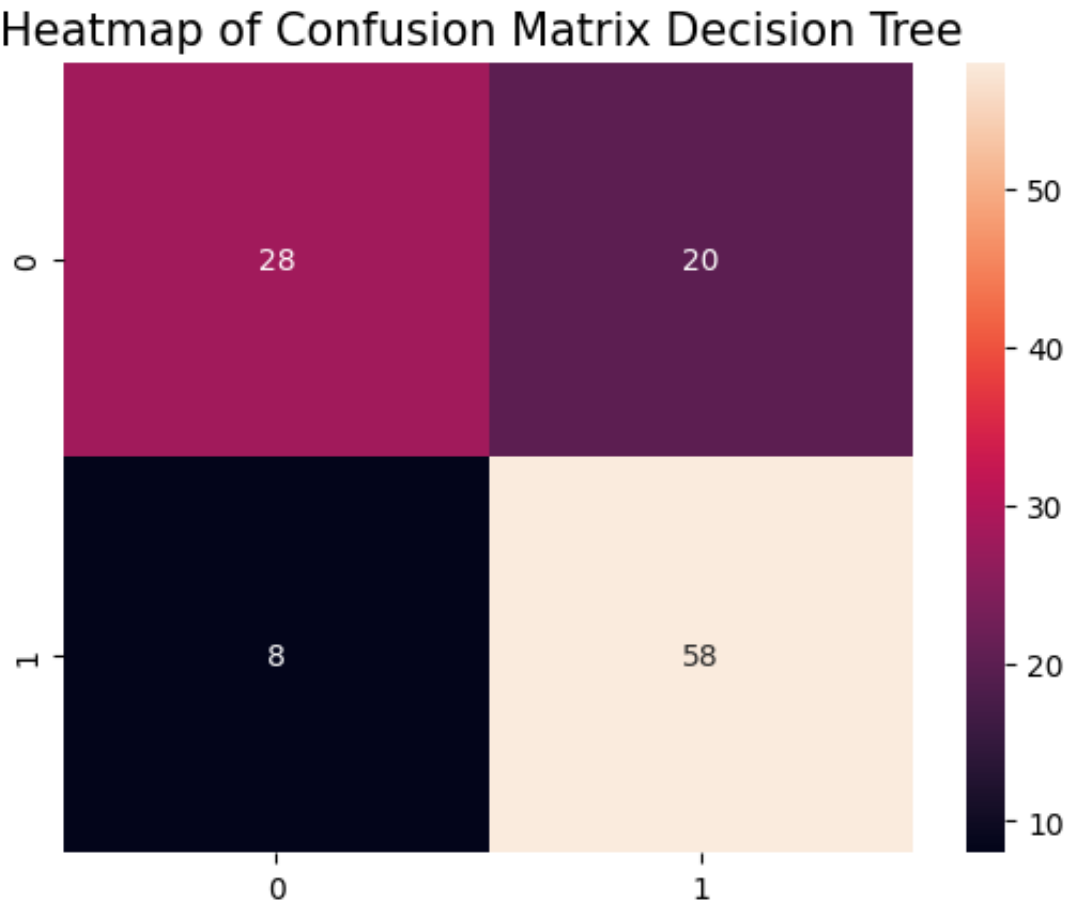
Out[ ]:
```
0.9473684210526315
```

In [ ]:
```
# Train with Standard scaled Data
dt_classifier2 = DecisionTreeClassifier(criterion = 'entropy', random_sta
dt_classifier2.fit(X_train_sc, y_train)
y_pred_dt_sc = dt_classifier.predict(X_test_sc)
accuracy=accuracy_score(y_test, y_pred_dt_sc)

print("="*30)
print(f"Accuracy Score: {accuracy:.4f}")
print("="*30)
cm = confusion_matrix(y_test, y_pred_dt_sc)
plt.title('Heatmap of Confusion Matrix Decision Tree ', fontsize = 15)
sns.heatmap(cm, annot = True)
plt.show()
```

```
==============================
Accuracy Score: 0.7544
==============================
```

```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-pac
kages/sklearn/base.py:465: UserWarning: X does not have valid feature name
s, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```



Random Forest

In [ ]:
```
# Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
rf_classifier = RandomForestClassifier(n_estimators = 20, criterion = 'en
rf_classifier.fit(X_train, y_train)
y_pred_rf = rf_classifier.predict(X_test)
accuracy_score(y_test, y_pred_rf)
```

Out[ ]: 0.9736842105263158

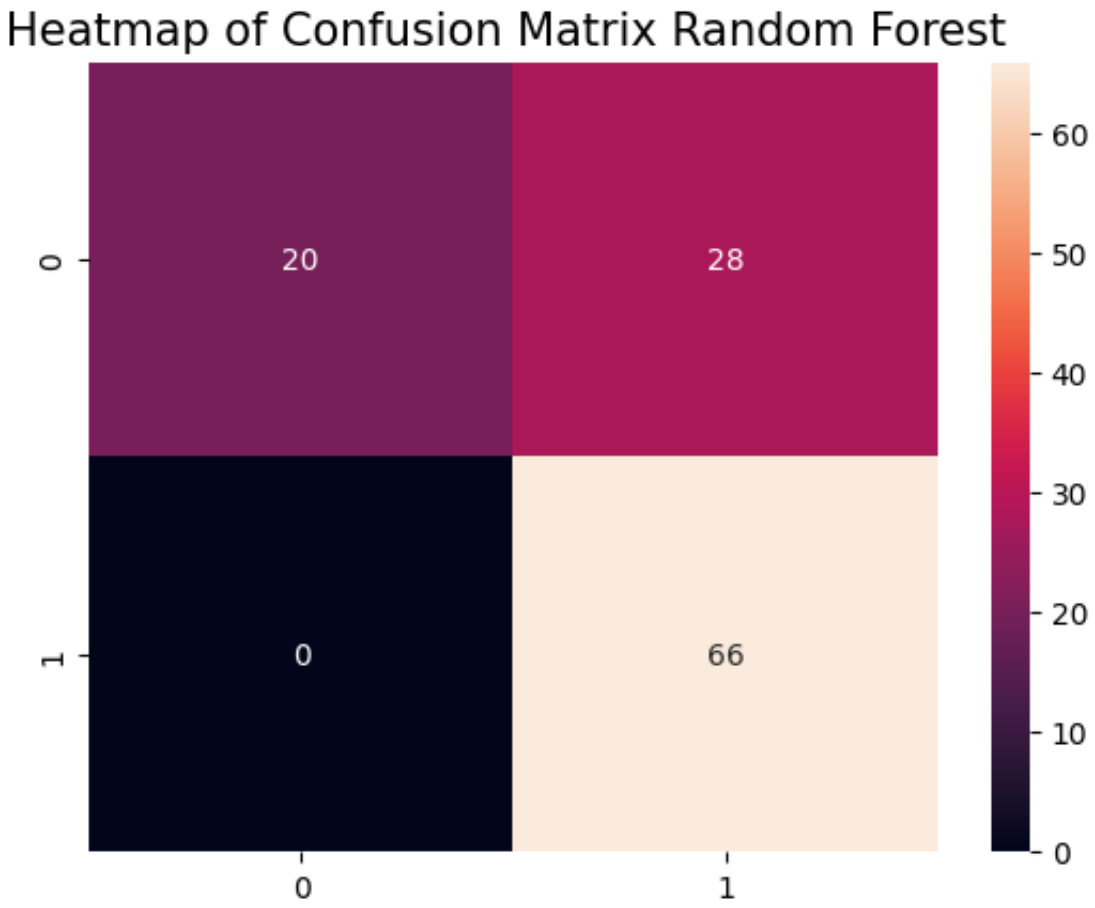
In [ ]:
```
# Train with Standard scaled Data
rf_classifier2 = RandomForestClassifier(n_estimators = 20, criterion = 'e
rf_classifier2.fit(X_train_sc, y_train)
y_pred_rf_sc = rf_classifier.predict(X_test_sc)
accuracy=accuracy_score(y_test, y_pred_rf_sc)

print("="*30)
print(f"Accuracy Score: {accuracy:.4f}")
print("="*30)
cm = confusion_matrix(y_test, y_pred_rf_sc)
```

```
plt.title('Heatmap of Confusion Matrix Random Forest ', fontsize = 15)
sns.heatmap(cm, annot = True)
plt.show()
```

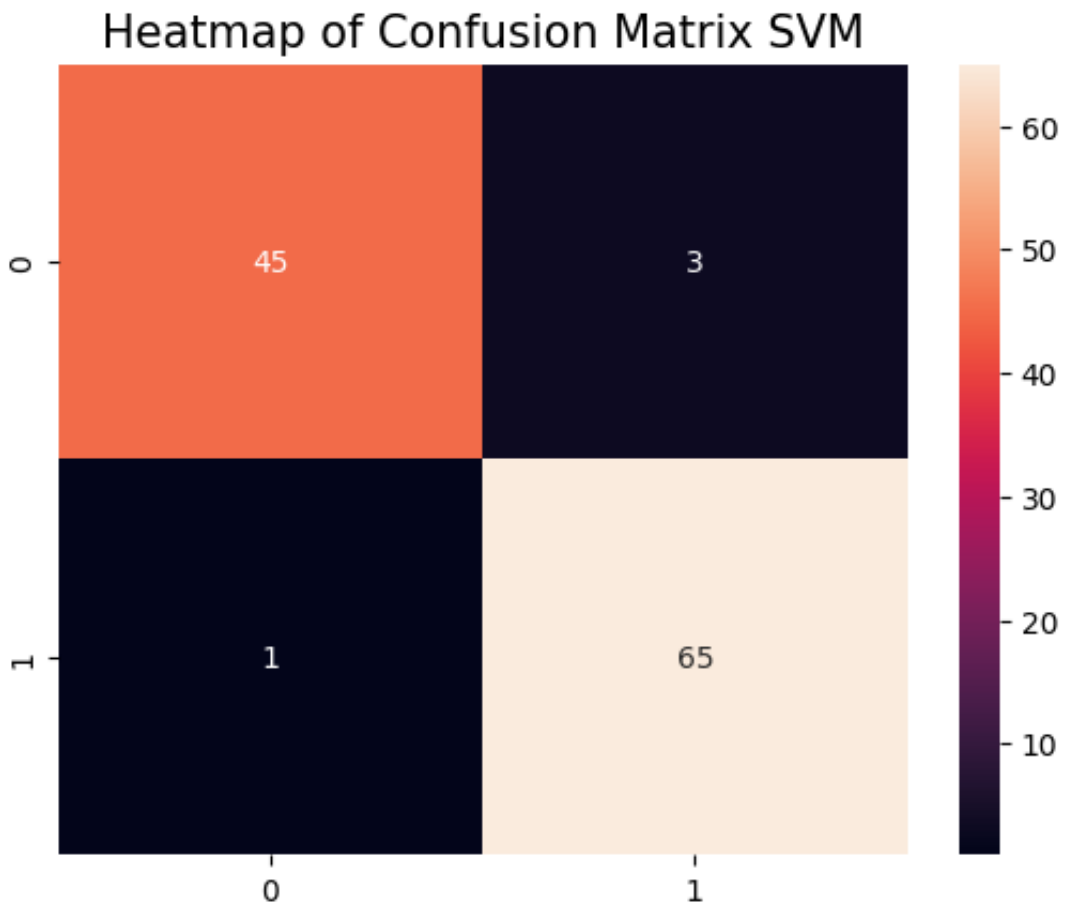
```
==============================
Accuracy Score: 0.7544
==============================
```

```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-pac
kages/sklearn/base.py:465: UserWarning: X does not have valid feature name
s, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```



Confusion Matrix

In [ ]:
```
cm = confusion_matrix(y_test, y_pred_svc_sc)
plt.title('Heatmap of Confusion Matrix SVM', fontsize = 15)
sns.heatmap(cm, annot = True)
plt.show()
```



Classification Report of Model

In [ ]:
```
accuracy=accuracy_score(y_test, y_pred_svc_sc)
print("="*30)
print(f"Accuracy Score: {accuracy:.4f}")
print("="*30)
print(classification_report(y_test, y_pred_svc_sc))
```

```
==============================
Accuracy Score: 0.9649
==============================
              precision    recall  f1-score   support

         0.0       0.98      0.94      0.96        48
         1.0       0.96      0.98      0.97        66

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114
```