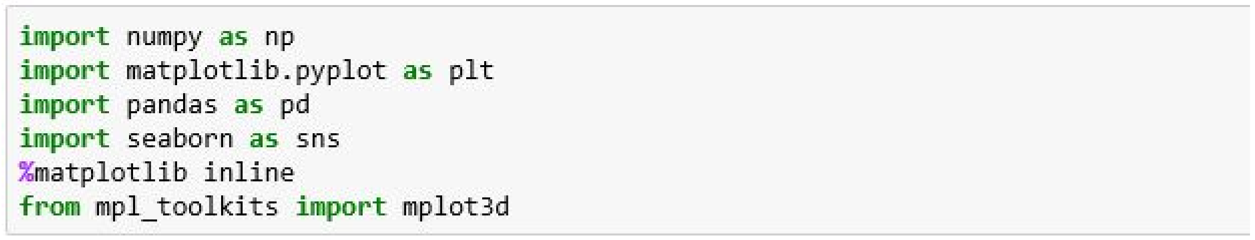
“Breast Cancer Prediction using Machine Learning”

These days Breast Cancer is a significant public health problem in society. It is almost one of the most common forms of cancer in women. This document is on early prediction of Breast cancer using machine learning technique as it can help in early diagnosis of Breast cancer, which can improve the prognosis and chances of survival significantly.

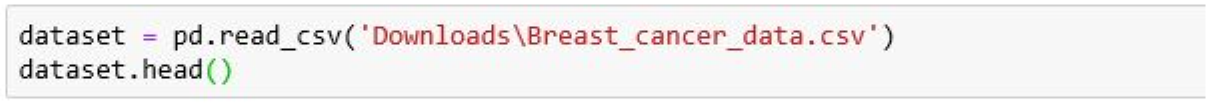
We will use Breast Cancer prediction dataset from Kaggle:
https://www.kaggle.com/merishnasuwal/breast-cancer-prediction-dataset

First of all import all required Libraries:



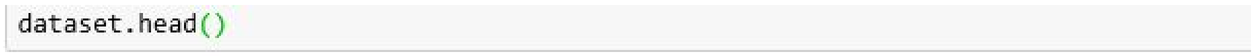
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
from mpl_toolkits import mplot3d
```

Import data:

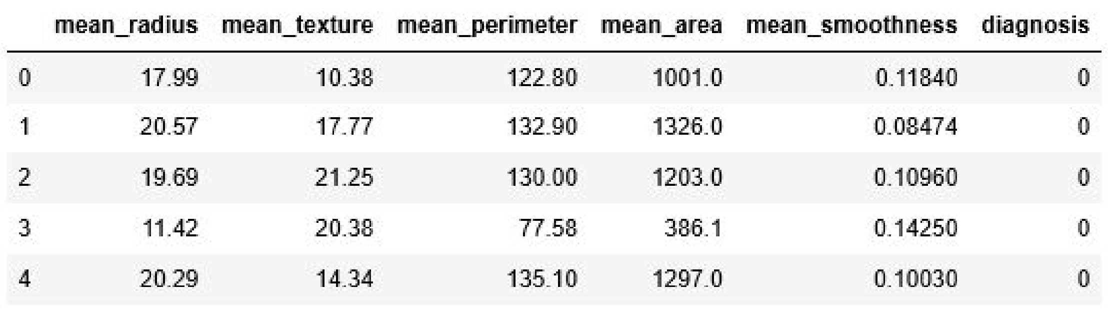


```
dataset = pd.read_csv('Downloads\Breast_cancer_data.csv')
dataset.head()
```

Dataset details:



```
dataset.head()
```



	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
0	17.99	10.38	122.80	1001.0	0.11840	0
1	20.57	17.77	132.90	1326.0	0.08474	0
2	19.69	21.25	130.00	1203.0	0.10960	0
3	11.42	20.38	77.58	386.1	0.14250	0
4	20.29	14.34	135.10	1297.0	0.10030	0

Dataset Summary using Describe:

```
dataset.describe()
```

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.627417
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.483918
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.000000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.000000
50%	13.370000	18.840000	86.240000	551.100000	0.095870	1.000000
75%	15.780000	21.800000	104.100000	782.700000	0.105300	1.000000
max	28.110000	39.280000	188.500000	2501.000000	0.163400	1.000000

Shape of the dataset

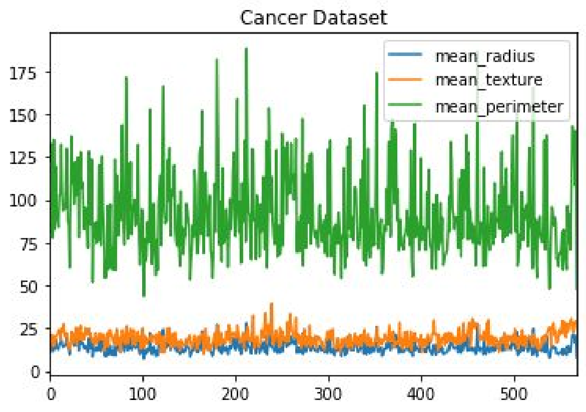
```
print("Cancer data set dimensions : {}".format(dataset.shape))
```

```
Cancer data set dimensions : (569, 6)
```

Mean radius, Mean texture and Mean Perimeter Comparison with line chart:

```
dataset.drop(['diagnosis','mean_area','mean_smoothness'], axis=1).plot.line(title='Cancer Dataset')
```

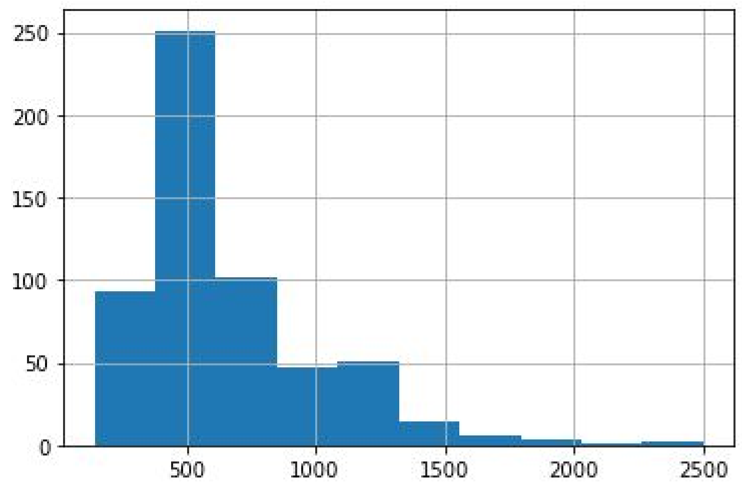
```
<matplotlib.axes._subplots.AxesSubplot at 0x2325a7be748>
```



Histogram and Boxplot of mean area column in dataset:

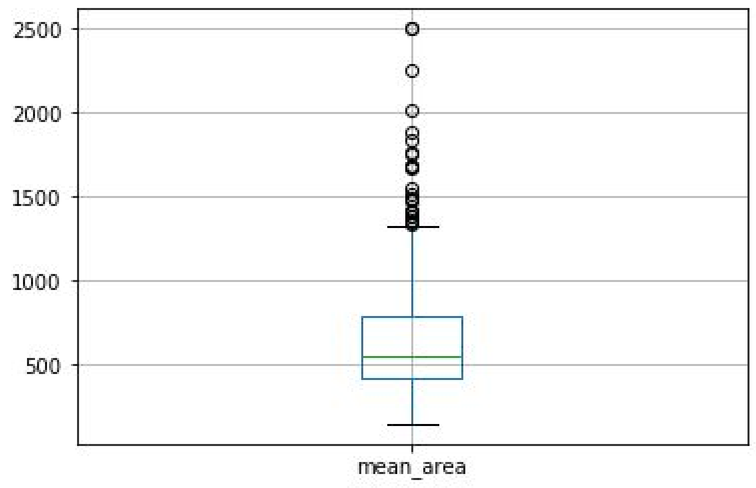
```
dataset['mean_area'].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2325a82f5f8>
```



```
dataset.boxplot(column = 'mean_area')
```

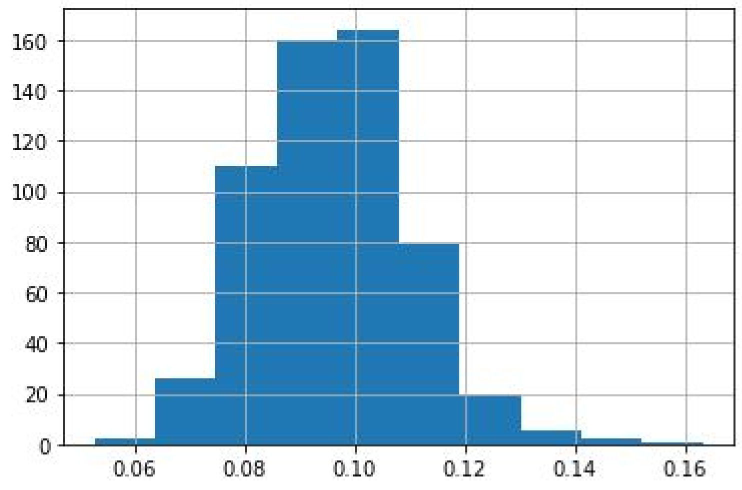
```
<matplotlib.axes._subplots.AxesSubplot at 0x2325a8b2438>
```



Histogram and Boxplot of mean smoothness column in dataset:

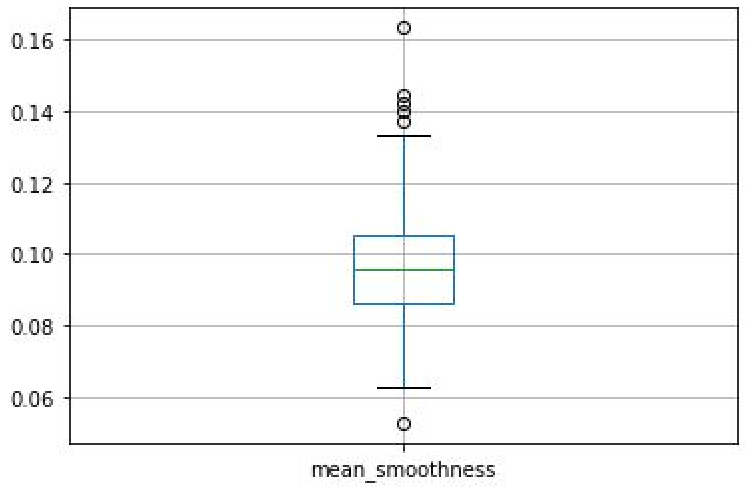
```
dataset['mean_smoothness'].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2325a8f8e80>
```



```
dataset.boxplot(column = 'mean_smoothness')
```

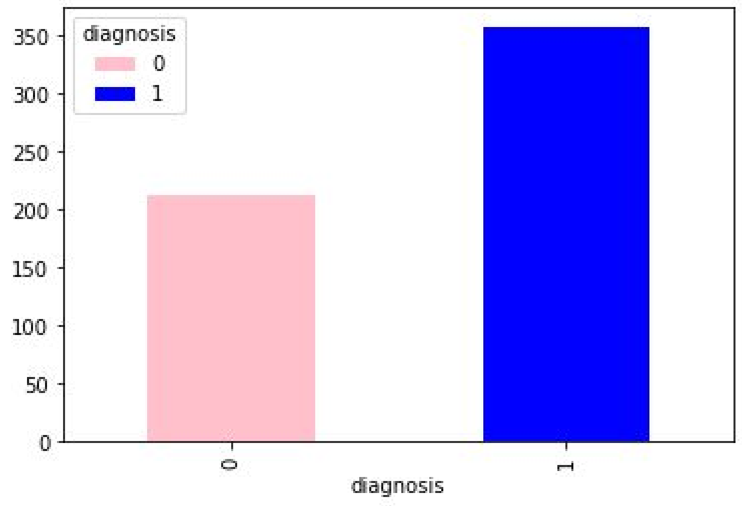
```
<matplotlib.axes._subplots.AxesSubplot at 0x2325a88f908>
```



Diagnosis Samples in dataset:

```
# number of samples ! for infected and 0 for non-infected
temp3 = pd.crosstab(dataset['diagnosis'], dataset['diagnosis'])
temp3.plot(kind='bar', stacked=True, color=['pink','blue'], grid=False)
```

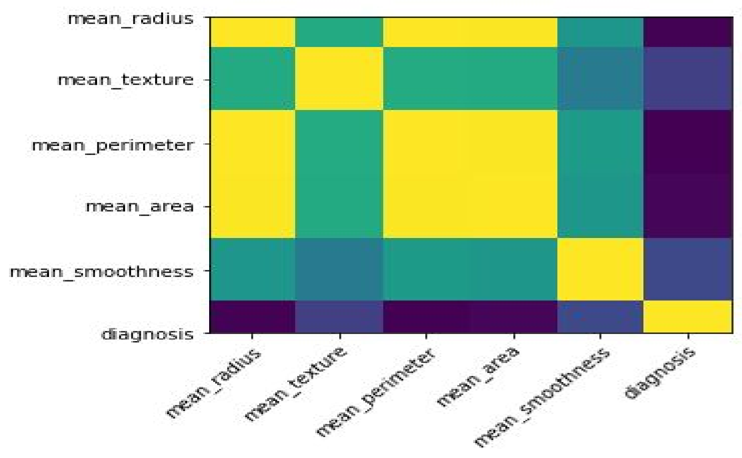
```
<matplotlib.axes._subplots.AxesSubplot at 0x2325a9db198>
```



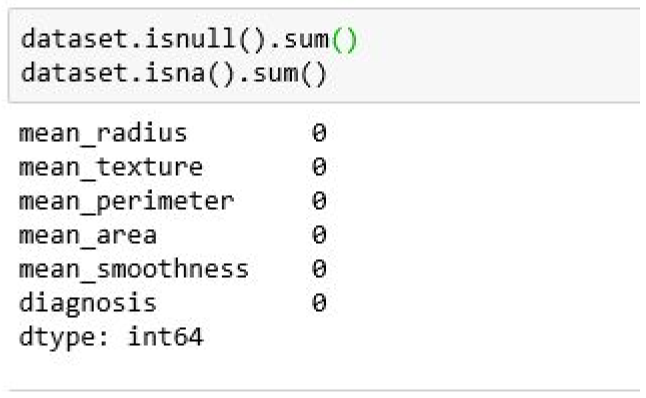
Correlation heat map of dataset:

```
#HeatMap of dataset corrilation
corr = dataset.corr()
fig, ax = plt.subplots()
# create heatmap
im = ax.imshow(corr.values)

# set labels
ax.set_xticks(np.arange(len(corr.columns)))
ax.set_yticks(np.arange(len(corr.columns)))
ax.set_xticklabels(corr.columns)
ax.set_yticklabels(corr.columns)

# Rotate the tick labels and set their alignment.
plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
         rotation_mode="anchor")
```

```
[None, None, None, None, None, None, None, None, None, None, None, None]
```



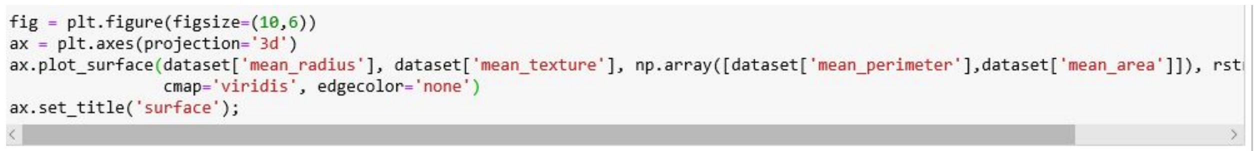
Missing values analysis in dataset:



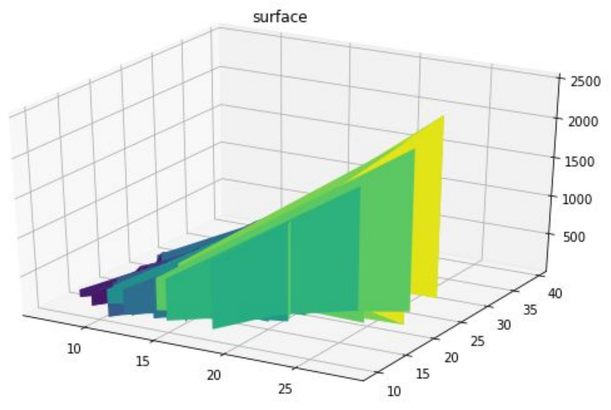
```
dataset.isnull().sum()
dataset.isna().sum()
```

```
mean_radius        0
mean_texture       0
mean_perimeter     0
mean_area          0
mean_smoothness    0
diagnosis          0
dtype: int64
```

Dataset 3D Projection

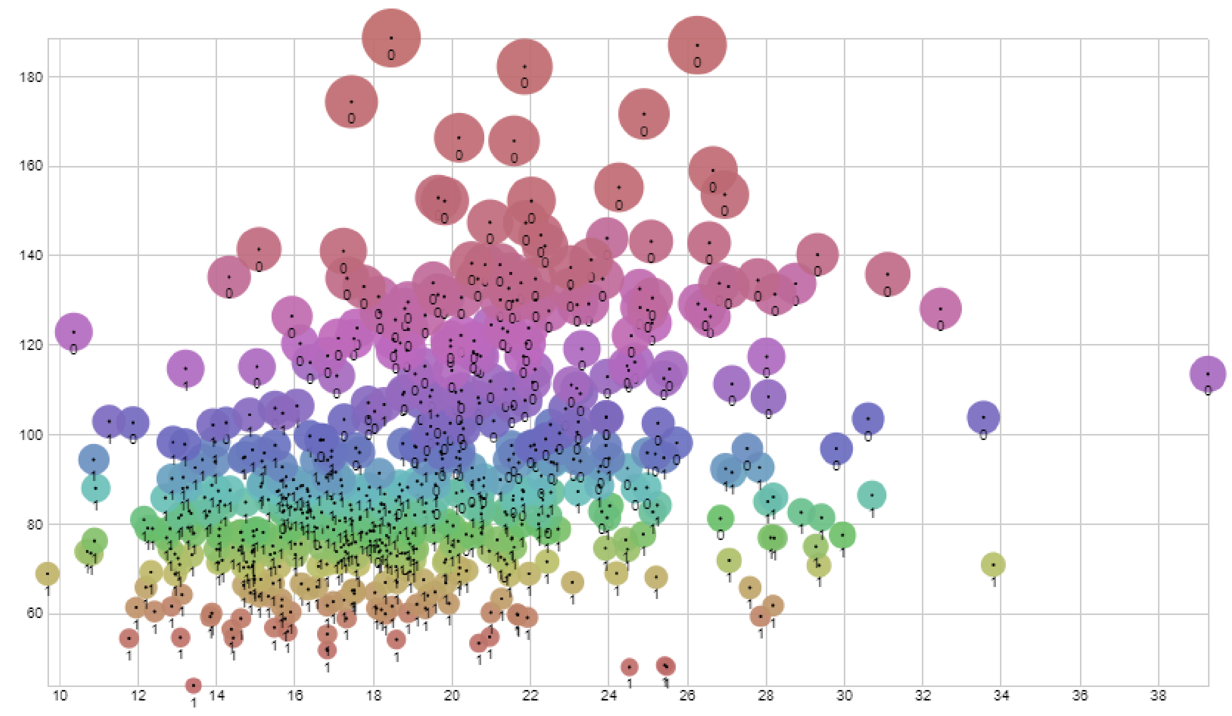


```
fig = plt.figure(figsize=(10,6))
ax = plt.axes(projection='3d')
ax.plot_surface(dataset['mean_radius'], dataset['mean_texture'], np.array([dataset['mean_perimeter'],dataset['mean_area']]), rst
                cmap='viridis', edgecolor='none')
ax.set_title('surface');
```



Shape of the data: X-axis = Mean radius, Y-axis = Mean texture, Z = [Mean perimeter, Mean Area]

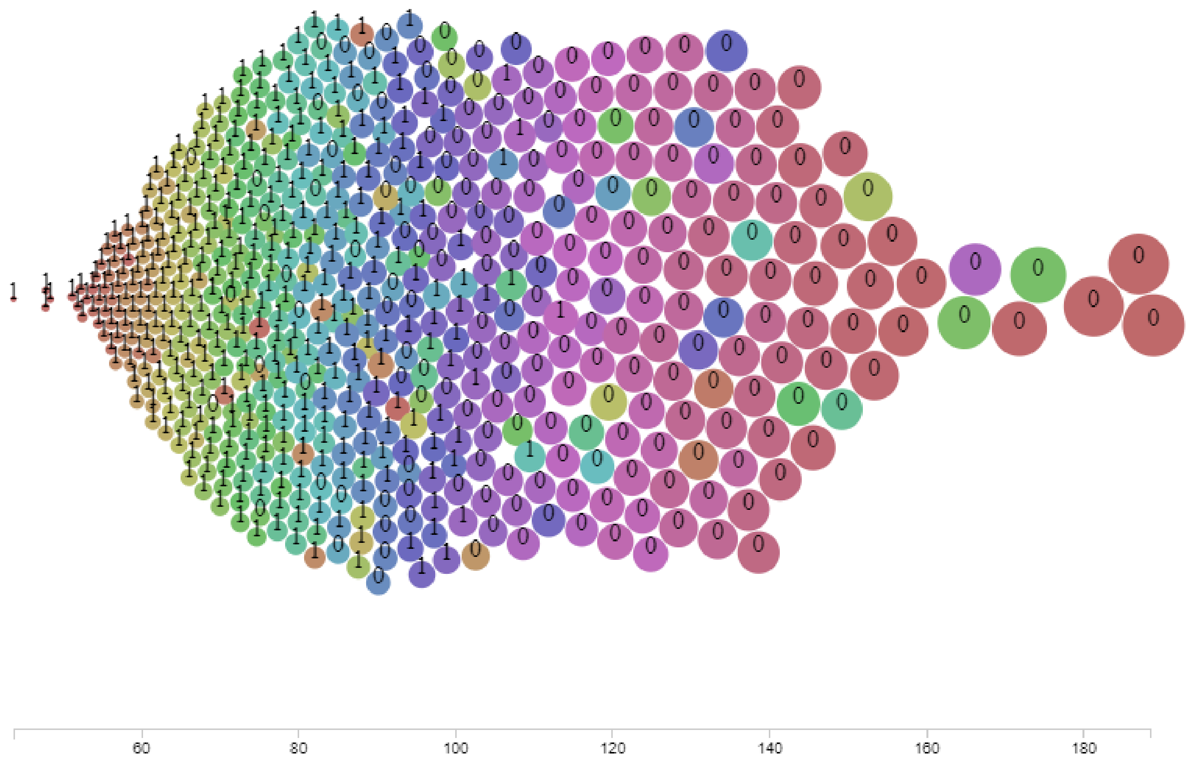
Scatter plot



X-axis = Mean texture, y-axis = Mean Perimeter, Color =Mean radius, Size = Mean area, 1 = Infected, 0 = Not infected

As the plot show, samples with mean perimeter b/w 0-100 are the mostly infected and the samples above 100 are mostly not infected. As the size shows, samples with less than or equal to median of Mean Area are the infected ones.

Bee swarm plot



X-axis = Mean perimeter, Size = Mean Radius, Color = Mean texture.

As the Bee swarm plot shows, samples with greater radius then Mean of Mean Radius are not infected samples but still it is very clear the samples with Mean perimeter b/w 0-100 are the infected samples.

Separating labels from dataset for training:

```
X = dataset.iloc[:, 0:5].values
Y = dataset.iloc[:,5].values
```

Splitting the data into training and testing data:

```
# Spliting data into trainingset(75%) and testset(25%)

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, random_state = 0)
```

Scaling the feature before training the model:

```
# Scaling the features

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Training the model using Support Vector Machines :

```
#Using SVC method of svm class to use Kernel SVM Algorithm

# import warnings filter
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)

from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, Y_train)
```

Prediction of labels using test data:

```
Y_pred = classifier.predict(X_test)
```

Confusion Matrix of our model:

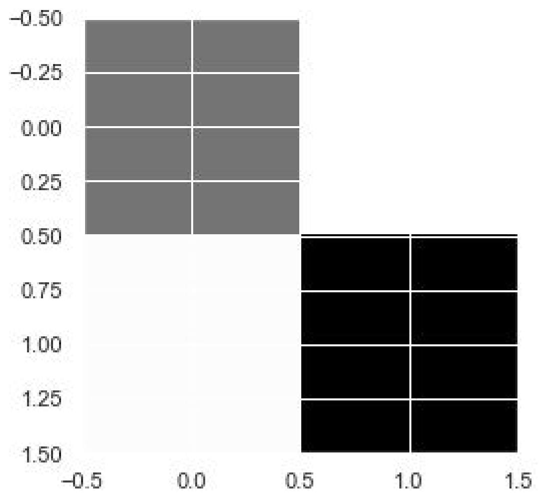
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, Y_pred)
```

```
print(cm)
```

```
[[48  5]
 [ 6 84]]
```

```
plt.imshow(cm, cmap='binary')
```

```
<matplotlib.image.AxesImage at 0x2325c981a58>
```



Accuracy Of the Model:

```
def accuracy(confusion_matrix):
    diagonal_sum = confusion_matrix.trace()
    sum_of_all_elements = confusion_matrix.sum()
    return diagonal_sum / sum_of_all_elements
```

```
accur = accuracy(cm)
print("Model Accuracy: ", accur)
```

```
Model Accuracy:  0.9230769230769231
```