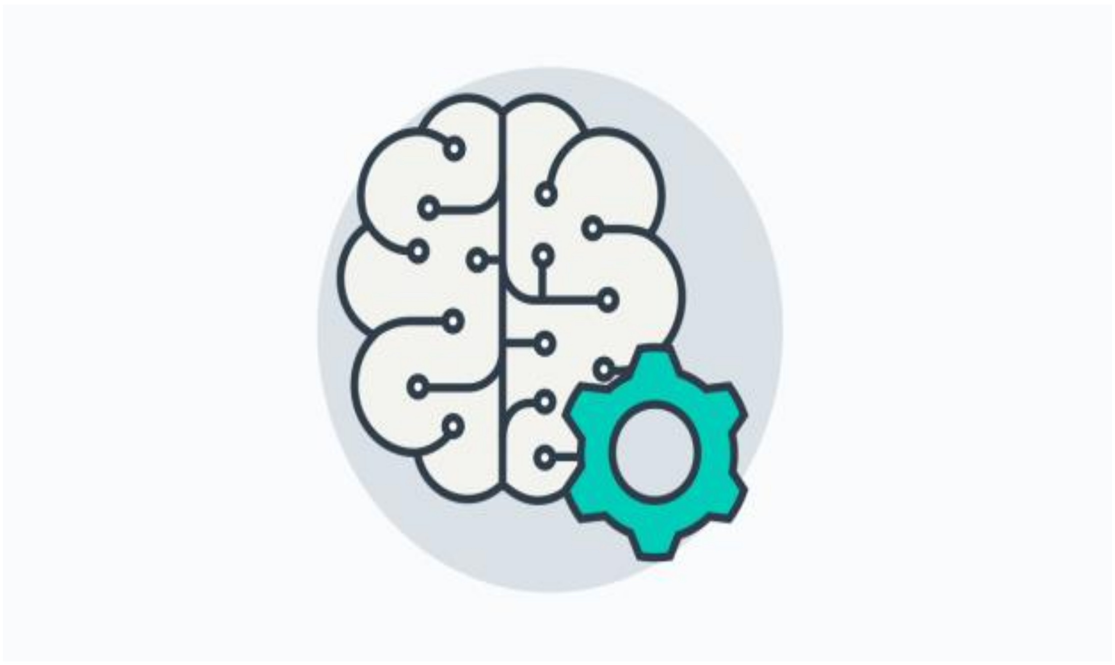
Prediction of Breast Cancer

Using

Machine Learning Techniques.



PROJECT REPORT

On

<u>Prediction of Breast Cancer using Machine Learning Techniques</u>

Submitted towards the partial fulfillment of the criteria for award of Data Science by TRENSDIVE

Submitted By: T.Dilip

Student Full Name: T.Dilip

Course and Batch: Batch Details – CDSM – B2

Abstract

Acknowledgements

We are using this opportunity to express my gratitude to everyone who supported us throughout the course of this group project. We are thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. I am sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

Further, we were fortunate to have **(Suresh Kadari)** as our mentor. He has readily shared his immense knowledge in data analytics and guide us in a manner that the out- come resulted in enhancing our data skills.

We also thank Pardhu.G for his constant support towards the project.

We wish to thank all the faculties and friends, as this project utilized knowledge gained from every course that formed the DS program.

We certify that the work done by us for conceptualizing and completing this project is original and authentic.

Date: 19-08-2020 Member 2 : T. DILIP

Place: Hyderabad

INDEX:-

CHAPTER-1

INTRODUCTION

- Breast cancer (BC) is one of the most common cancers among women worldwide, representing the majority of new cancer cases and cancer-related deaths according to global statistics, making it a significant public health problem in today's society.
- The early diagnosis of BC can improve the prognosis and chance of survival significantly, as it can promote timely clinical treatment to patients. Further accurate classification of benign tumors can prevent patients undergoing unnecessary treatments. Thus, the correct diagnosis of BC and classification of patients into malignant or benign groups is the subject of much research. Because of its unique advantages in critical features detection from complex BC datasets, machine learning (ML) is widely recognized as the methodology of choice in BC pattern classification and forecast modelling.
- Classification and data mining methods are an effective way to classify data. Especially in medical field, where those methods are widely used in diagnosis and analysis to make decisions.

Recommended Screening Guidelines:

- Mammography. The most important screening test for breast cancer is the mammogram. A mammogram is an X-ray of the breast. It can detect breast cancer up to two years before the tumor can be felt by you or your doctor.
- Women age 40–45 or older who are at average risk of breast cancer should have a mammogram once a year.
- Women at high risk should have yearly mammograms along with an MRI starting at age 30.

Some Risk Factors for Breast Cancer:-

The following are some of the known risk factors for breast cancer. However, most cases of breast cancer cannot be linked to a specific cause. Talk to your doctor about your specific risk.

- Age. The chance of getting breast cancer increases as women age. Nearly 80 percent of breast cancers are found in women over the age of 50.
- Personal history of breast cancer. A woman who has had breast cancer in one breast is at an increased risk of developing cancer in her other breast.
- Family history of breast cancer. A woman has a higher risk of breast cancer if her mother, sister or daughter had breast cancer, especially at a young age (before 40). Having other relatives with breast cancer may also raise the risk.

- Genetic factors. Women with certain genetic mutations, including changes to the BRCA1 and BRCA2 genes, are at higher risk of developing breast cancer during their lifetime. Other gene changes may raise breast cancer risk as well.
- **Childbearing and menstrual history**. The older a woman is when she has her first child, the greater her risk of breast cancer. Also at higher risk are:
    - Women who menstruate for the first time at an early age (before 12)
    - Women who go through menopause late (after age 55)
    - Women who’ve never had children

CHAPTER-2
DATA PREPARATION

The dataset used in this story is publicly available and was created by Dr. William H. Wolberg, physician at the University Of Wisconsin Hospital at Madison, Wisconsin, USA. To create the dataset Dr. Wolberg used fluid samples, taken from patients with solid breast masses and an easy-to-use graphical computer program called Xcyt, which is capable of perform the analysis of cytological features based on a digital scan. The program uses a curve-fitting algorithm, to compute ten features from each one of the cells in the sample, than it calculates the mean value, extreme value and standard error of each feature for the image, returning a 30 real-valuated vector

Attribute Information:

1. ID number 2) Diagnosis (M = malignant, B = benign) 3–32)

Ten real-valued features are computed for each cell nucleus:

- radius (mean of distances from center to points on the perimeter)
- 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 three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius

Objectives:-

- This analysis aims to observe which features are most helpful in predicting malignant or benign cancer and to see general trends that may aid us in model selection and hyper parameter selection.

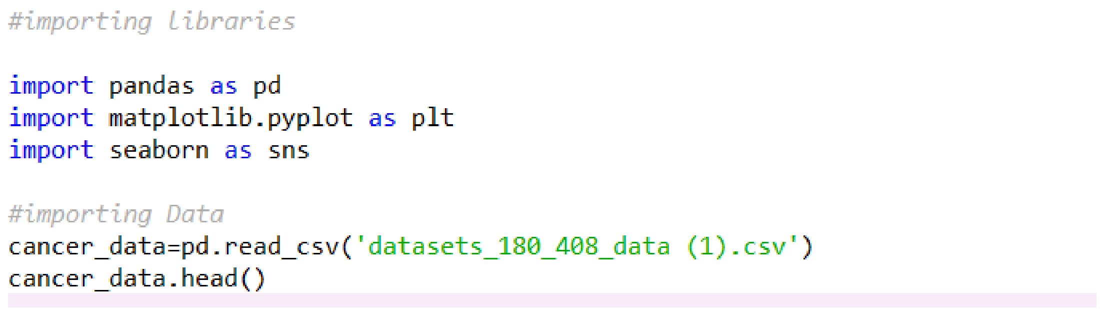
- The goal is to classify whether the breast cancer is benign or malignant.

- To achieve this i have used machine learning classification methods to fit a function that can predict the discrete class of new input.

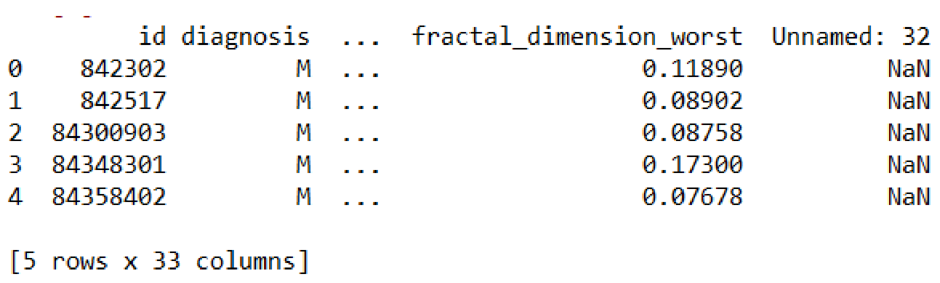
CHAPTER-3

DATA EXPLORATION

We will be using Spyder to work on this dataset. We will first go with importing the necessary libraries and import our dataset to Spyder :-

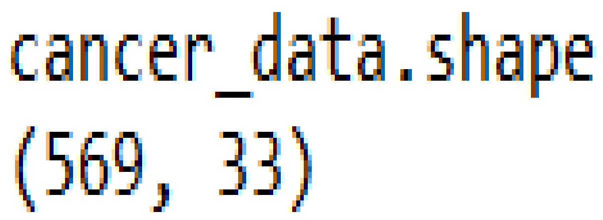


```
#importing libraries

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#importing Data
cancer_data=pd.read_csv('datasets_180_408_data (1).csv')
cancer_data.head()
```



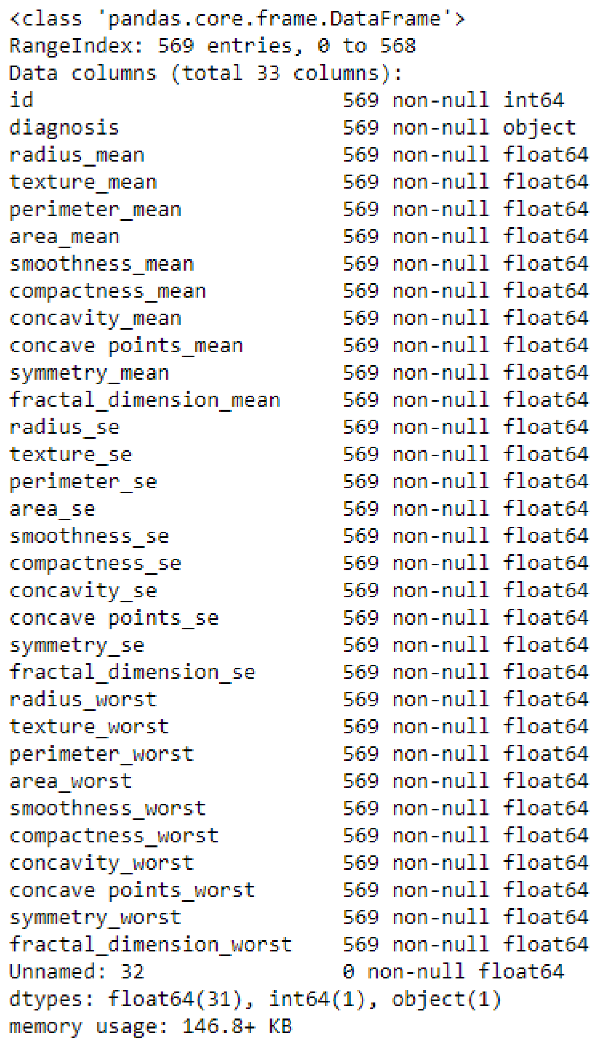
```
         id diagnosis  ...  fractal_dimension_worst  Unnamed: 32
0    842302         M  ...                  0.11890          NaN
1    842517         M  ...                  0.08902          NaN
2  84300903         M  ...                  0.08758          NaN
3  84348301         M  ...                  0.17300          NaN
4  84358402         M  ...                  0.07678          NaN

[5 rows x 33 columns]
```

We can find the dimensions of the data set using the panda dataset ‘shape’ attribute.



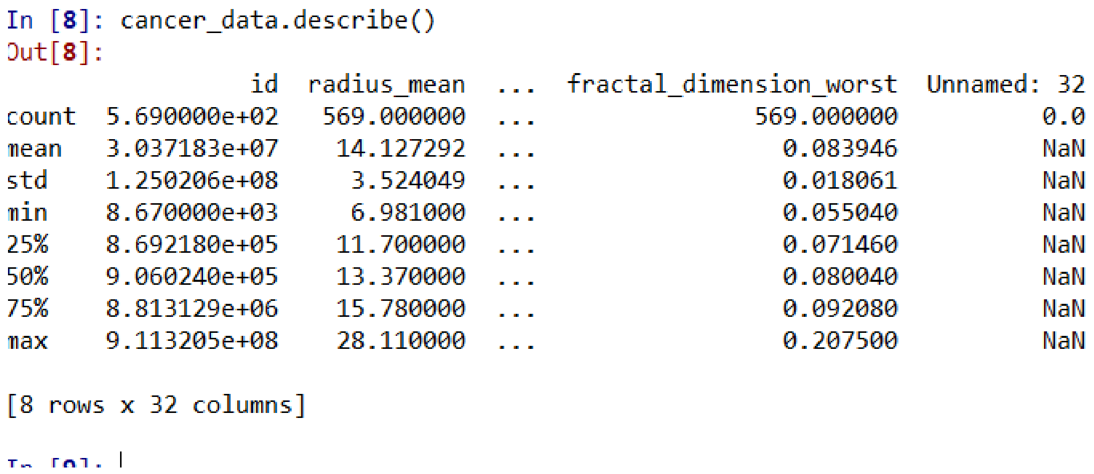
```
cancer_data.shape
(569, 33)
```

Info about the model(gives null value and count the non float values)



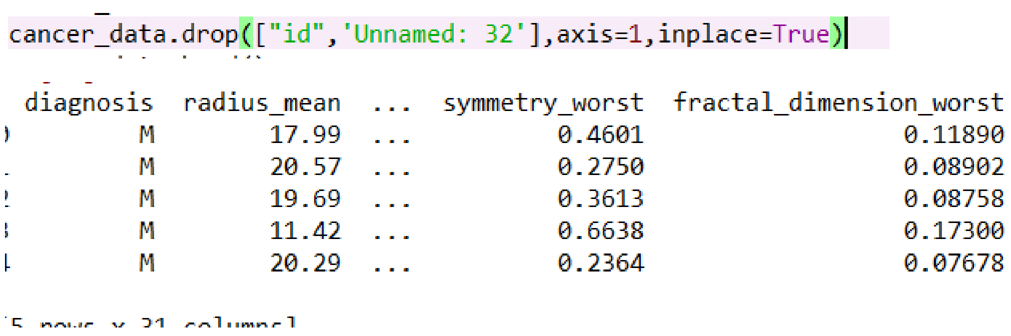
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
id                         569 non-null int64
diagnosis                  569 non-null object
radius_mean                569 non-null float64
texture_mean               569 non-null float64
perimeter_mean             569 non-null float64
area_mean                  569 non-null float64
smoothness_mean            569 non-null float64
compactness_mean           569 non-null float64
concavity_mean             569 non-null float64
concave points_mean        569 non-null float64
symmetry_mean              569 non-null float64
fractal_dimension_mean     569 non-null float64
radius_se                  569 non-null float64
texture_se                 569 non-null float64
perimeter_se               569 non-null float64
area_se                    569 non-null float64
smoothness_se              569 non-null float64
compactness_se             569 non-null float64
concavity_se               569 non-null float64
concave points_se          569 non-null float64
symmetry_se                569 non-null float64
fractal_dimension_se       569 non-null float64
radius_worst               569 non-null float64
texture_worst              569 non-null float64
perimeter_worst            569 non-null float64
area_worst                 569 non-null float64
smoothness_worst           569 non-null float64
compactness_worst          569 non-null float64
concavity_worst            569 non-null float64
concave points_worst       569 non-null float64
symmetry_worst             569 non-null float64
fractal_dimension_worst    569 non-null float64
Unnamed: 32                0 non-null float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

Numerical description about the data (mean,median,25%,interquantile range and many other value of each feature.



```
In [8]: cancer_data.describe()
Out[8]:
                 id  radius_mean  ...  fractal_dimension_worst  Unnamed: 32
count  5.690000e+02   569.000000  ...               569.000000          0.0
mean   3.037183e+07    14.127292  ...                 0.083946          NaN
std    1.250206e+08     3.524049  ...                 0.018061          NaN
min    8.670000e+03     6.981000  ...                 0.055040          NaN
25%    8.692180e+05    11.700000  ...                 0.071460          NaN
50%    9.060240e+05    13.370000  ...                 0.080040          NaN
75%    8.813129e+06    15.780000  ...                 0.092080          NaN
max    9.113205e+08    28.110000  ...                 0.207500          NaN

[8 rows x 32 columns]

In [9]:
```

Dropping the column that contain Nan Values:-

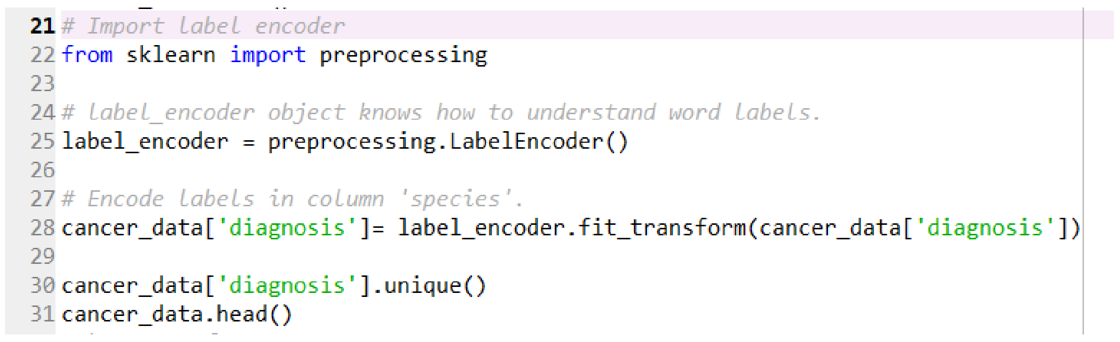


```
cancer_data.drop(["id",'Unnamed: 32'],axis=1,inplace=True)
```

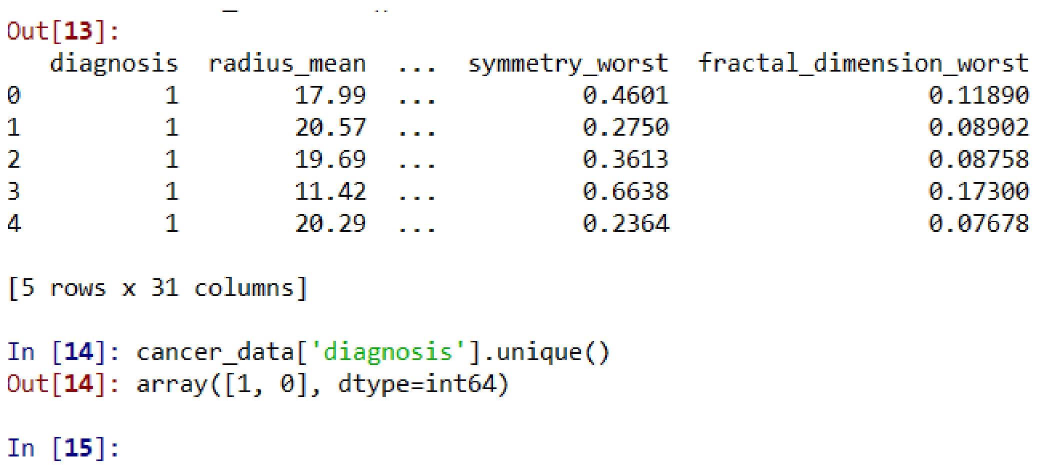
```
  diagnosis  radius_mean  ...  symmetry_worst  fractal_dimension_worst
          M        17.99  ...          0.4601                  0.11890
          M        20.57  ...          0.2750                  0.08902
          M        19.69  ...          0.3613                  0.08758
          M        11.42  ...          0.6638                  0.17300
          M        20.29  ...          0.2364                  0.07678

[5 rows x 31 columns]
```

Label Encoding:-

We will use Label Encoder to label the categorical data. Label Encoder is the part of SciKit Learn library in Python and used to convert categorical data, or text data, into numbers, which our predictive models can better understand.

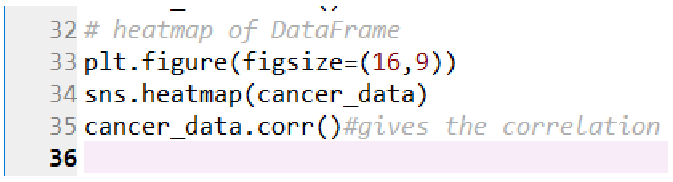


```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'species'.
cancer_data['diagnosis']= label_encoder.fit_transform(cancer_data['diagnosis'])

cancer_data['diagnosis'].unique()
cancer_data.head()
```

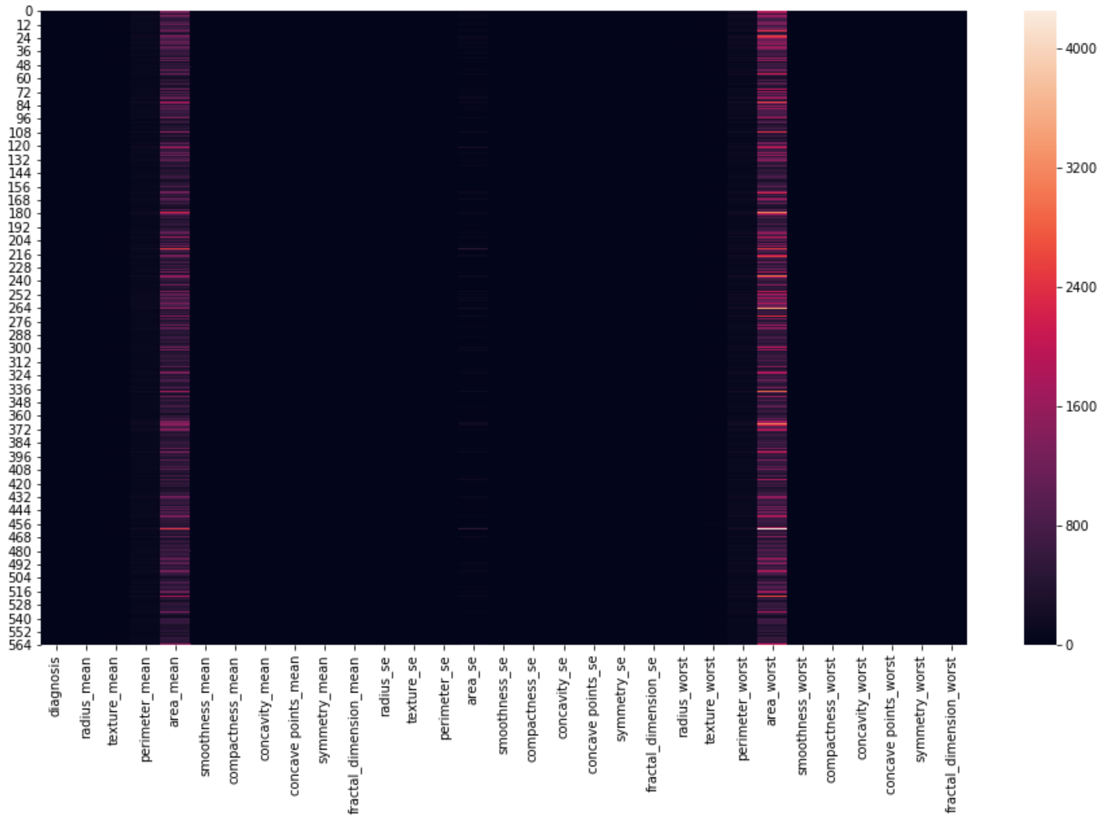


```
Out[13]:
   diagnosis  radius_mean  ...  symmetry_worst  fractal_dimension_worst
0          1        17.99  ...          0.4601                  0.11890
1          1        20.57  ...          0.2750                  0.08902
2          1        19.69  ...          0.3613                  0.08758
3          1        11.42  ...          0.6638                  0.17300
4          1        20.29  ...          0.2364                  0.07678

[5 rows x 31 columns]

In [14]: cancer_data['diagnosis'].unique()
Out[14]: array([1, 0], dtype=int64)

In [15]:
```

Heat Map:-

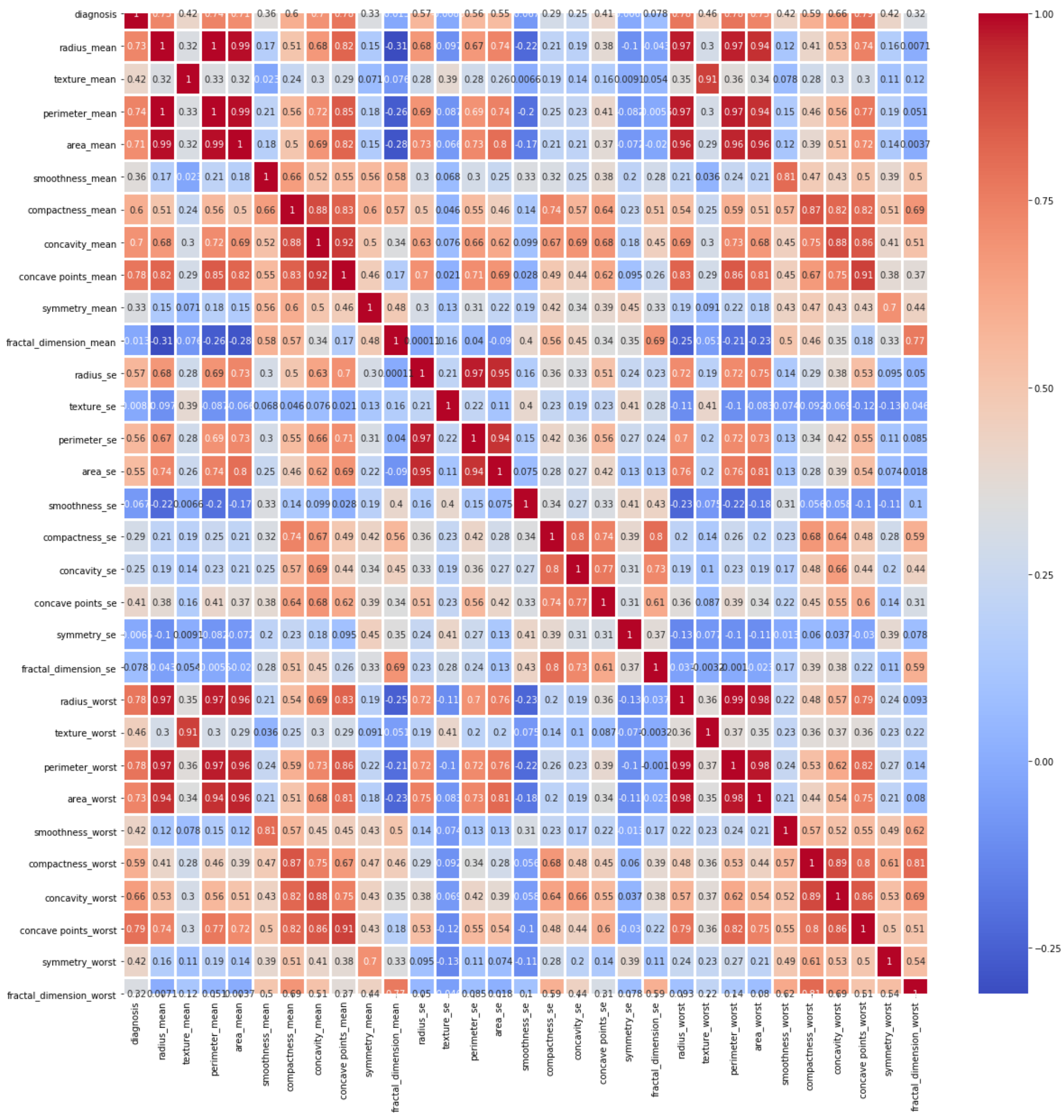


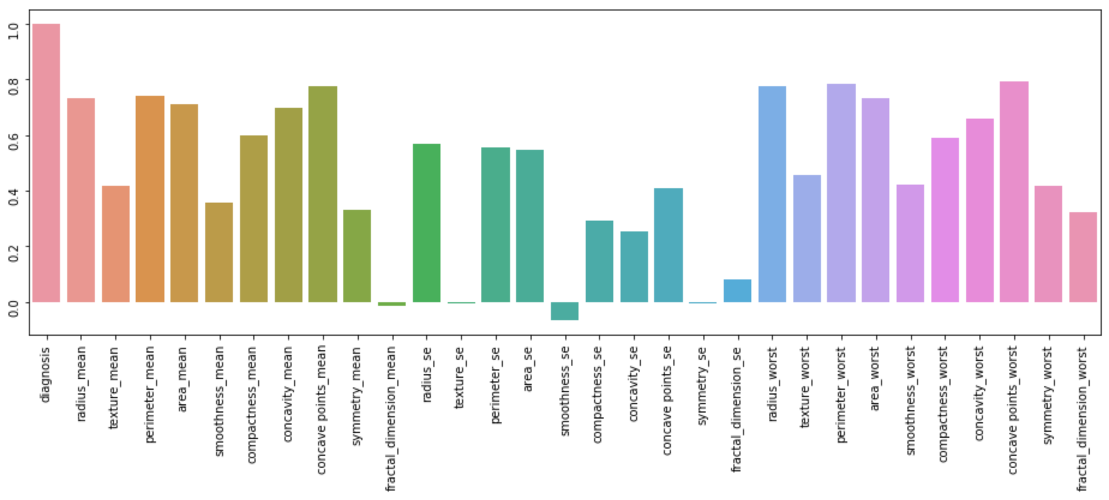
```
# heatmap of DataFrame
plt.figure(figsize=(16,9))
sns.heatmap(cancer_data)
cancer_data.corr()#gives the correlation
```



Heatmap of a correlation matrix:-

```
# Heatmap of Correlation matrix of breast cancer DataFrame
plt.figure(figsize=(20,20))
sns.heatmap(cancer_data.corr(), annot = True, cmap ='coolwarm', linewidths=2)

plt.figure(figsize=(16,9))
sns.heatmap(cancer_data)
cancer_data.corrwith(cancer_data.diagnosis) # visualize correlation barplot
plt.figure(figsize = (16,5))
ax = sns.barplot(cancer_data.corrwith(cancer_data.diagnosis).index, cancer_data.corrwith(cancer_data.diagnosis))
ax.tick_params(labelrotation = 90)
```





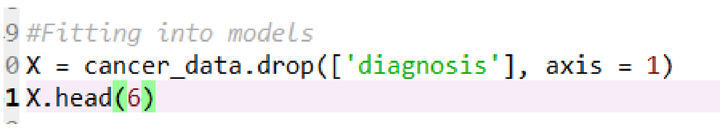
1.0
0.8
0.6
0.4
0.2
0.0
diagnosis
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave points_se
symmetry_se
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

CHAPTER-4

FITTING THE DATA INTO THE MODEL

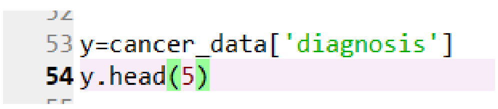
Split DataFrame in Train and Test

Input variable



```
#Fitting into models
X = cancer_data.drop(['diagnosis'], axis = 1)
X.head(6)
```

Output variable

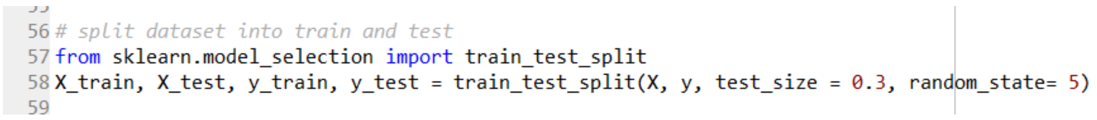


```
y=cancer_data['diagnosis']
y.head(5)
```

Split dataset for training and testing:-

The data we use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. We have the test dataset (or subset) in order to test our model's prediction on this subset.

We will do this using SciKit-Learn library in Python using the train_test_split method.

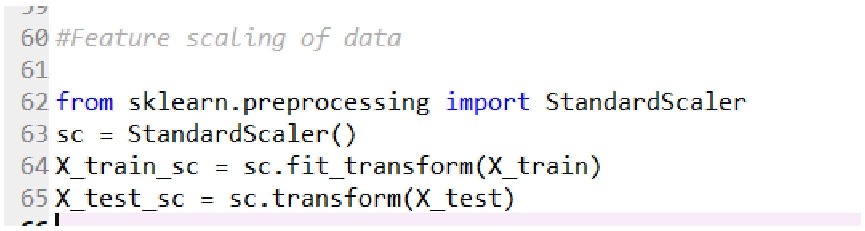


```
# split dataset into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state= 5)
```

Feature scaling of data:-

Most of the times, your dataset will contain features highly varying in magnitudes, units and range. But since, most of the machine learning algorithms use Eucledian distance between two data points in their computations. We need to bring all features to the same level of magnitudes. This can be achieved by scaling. This means that you're transforming your data so that it fits within a specific scale, like 0–100 or 0–1.

We will use StandardScaler method from SciKit-Learn library.



```
#Feature scaling of data

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

CHAPTER-5
Machine Learning Model Building

In our dataset we have the outcome variable or Dependent variable i.e Y having only two set of values, either M (Malign) or B(Benign). So we will use Classification algorithm of supervised learning.

We have different types of classification algorithms in Machine Learning :-

1.Support Vector Classifier

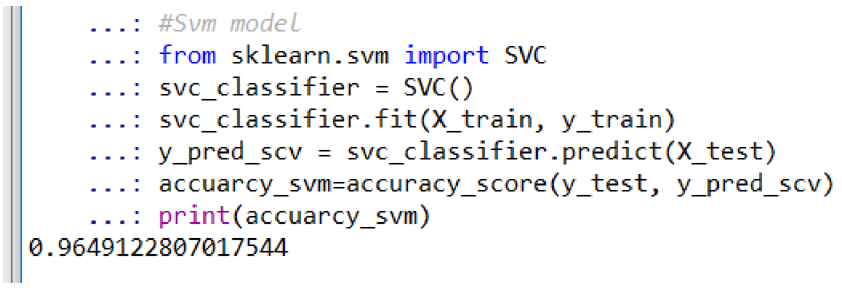
2.Logistic Regression

3.Decesion Trees

Lets start applying the algorithms :

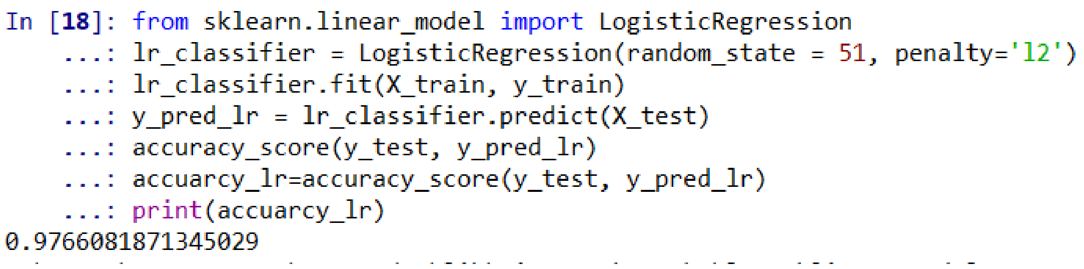
We will use sklearn library to import all the methods of classification algorithms.

1.Support Vector Classifier:-



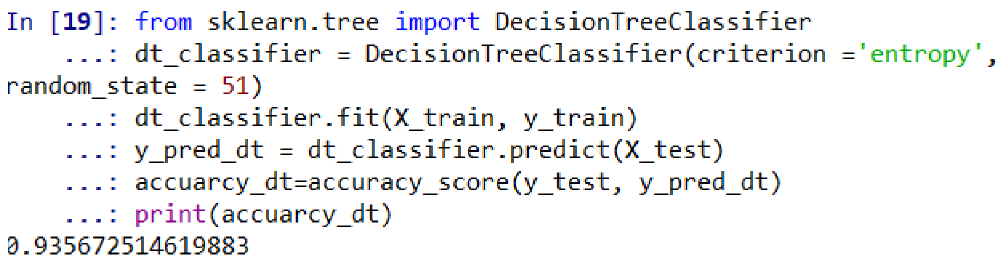
```
    ...: #Svm model
    ...: from sklearn.svm import SVC
    ...: svc_classifier = SVC()
    ...: svc_classifier.fit(X_train, y_train)
    ...: y_pred_scv = svc_classifier.predict(X_test)
    ...: accuarcy_svm=accuracy_score(y_test, y_pred_scv)
    ...: print(accuarcy_svm)
0.9649122807017544
```

2.Logistic Regression:-



```
In [18]: 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)
    ...: accuarcy_lr=accuracy_score(y_test, y_pred_lr)
    ...: print(accuarcy_lr)
0.9766081871345029
```

3.Decision Trees:-



```
In [19]: from sklearn.tree import DecisionTreeClassifier
    ...: dt_classifier = DecisionTreeClassifier(criterion ='entropy',
random_state = 51)
    ...: dt_classifier.fit(X_train, y_train)
    ...: y_pred_dt = dt_classifier.predict(X_test)
    ...: accuarcy_dt=accuracy_score(y_test, y_pred_dt)
    ...: print(accuarcy_dt)
0.935672514619883
```

Similarly, we have to do for test data and implement them on we can see that their will be no overfitting and underfitting the test data. their should low bias and low variance.

As,we can conclude the test data is performing nearly good result in Logistic classifier with low bias and variance.

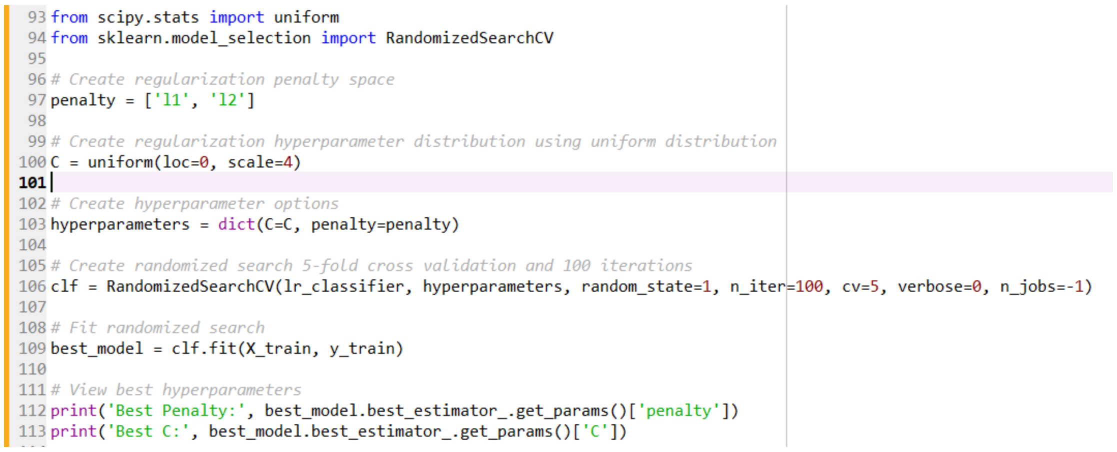
For further improving we should go for the tuning method such as randomised search on Logistic Regression Model because we want our accuracy to be more optimal and fix all contraints like precision ,recall ,beta value and support which are import to satisfy to overcome the Type I and Type II Error.

CHAPTER-6

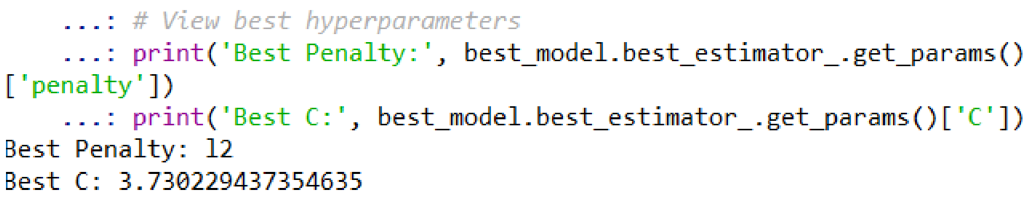
HYPERTUNING OF MODEL

Randomized search :-

Applying randomized search on the model which works on sample of data and it works more faster than any search tuning method

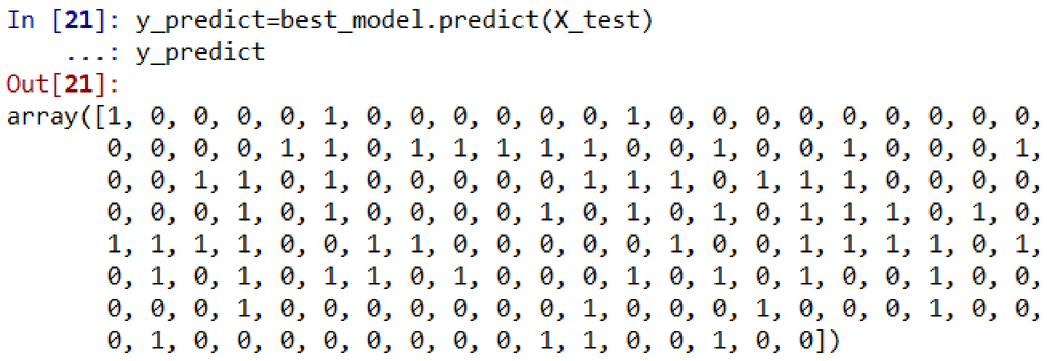


```
from scipy.stats import uniform
from sklearn.model_selection import RandomizedSearchCV

# Create regularization penalty space
penalty = ['l1', 'l2']

# Create regularization hyperparameter distribution using uniform distribution
C = uniform(loc=0, scale=4)

# Create hyperparameter options
hyperparameters = dict(C=C, penalty=penalty)

# Create randomized search 5-fold cross validation and 100 iterations
clf = RandomizedSearchCV(lr_classifier, hyperparameters, random_state=1, n_iter=100, cv=5, verbose=0, n_jobs=-1)

# Fit randomized search
best_model = clf.fit(X_train, y_train)

# View best hyperparameters
print('Best Penalty:', best_model.best_estimator_.get_params()['penalty'])
print('Best C:', best_model.best_estimator_.get_params()['C'])
```



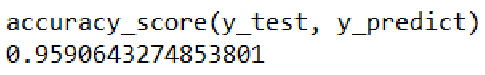
```
    ...: # View best hyperparameters
    ...: print('Best Penalty:', best_model.best_estimator_.get_params()
['penalty'])
    ...: print('Best C:', best_model.best_estimator_.get_params()['C'])
Best Penalty: l2
Best C: 3.730229437354635
```

We will now predict the test set results and check the accuracy with each of our model:



```
In [21]: y_predict=best_model.predict(X_test)
    ...: y_predict
Out[21]:
array([1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0,
       1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1,
       0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0])
```

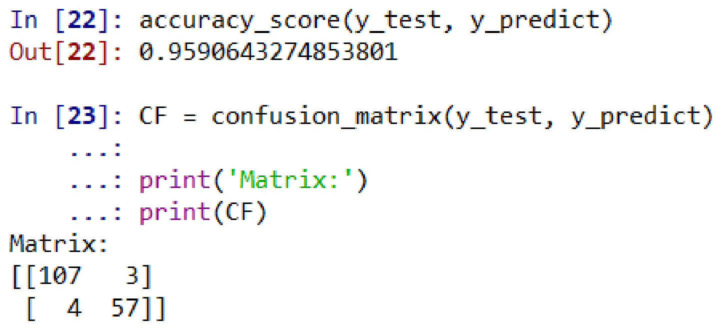
Accuracy of model after optimization:-



```
accuracy_score(y_test, y_predict)
0.9590643274853801
```

Confusion Matrix:-

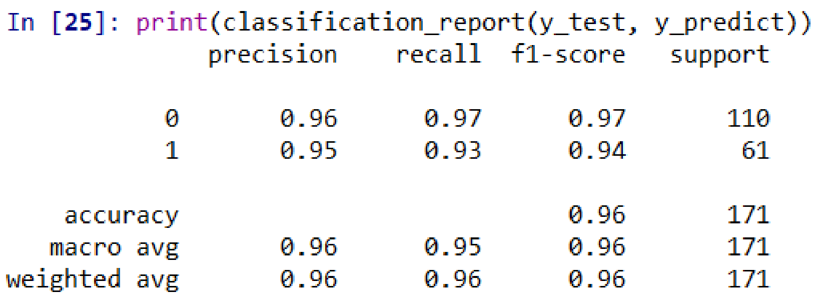
To check the accuracy we need to import confusion_matrix method of metrics class. The confusion matrix is a way of tabulating the number of mis-classifications, i.e., the number of predicted classes which ended up in a wrong classification bin based on the true classes.



```
In [22]: accuracy_score(y_test, y_predict)
Out[22]: 0.9590643274853801

In [23]: CF = confusion_matrix(y_test, y_predict)
    ...:
    ...: print('Matrix:')
    ...: print(CF)
Matrix:
[[107   3]
 [  4  57]]
```

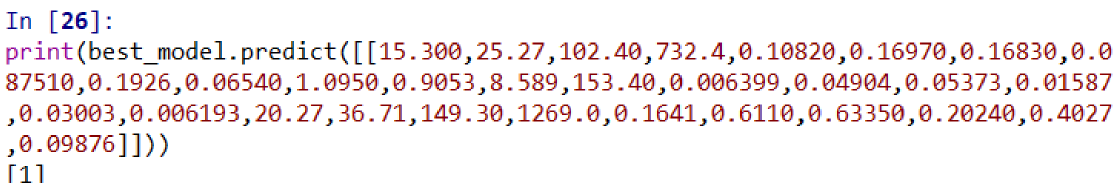
The model is giving 4 type II errors and it is best and for model is giving 3/164 near 0.0182 error. it means we have very less chance for the wrong prediction around zero.

Classification of the model:-



```
In [25]: print(classification_report(y_test, y_predict))
              precision    recall  f1-score   support

           0       0.96      0.97      0.97       110
           1       0.95      0.93      0.94        61

    accuracy                           0.96       171
   macro avg       0.96      0.95      0.96       171
weighted avg       0.96      0.96      0.96       171
```

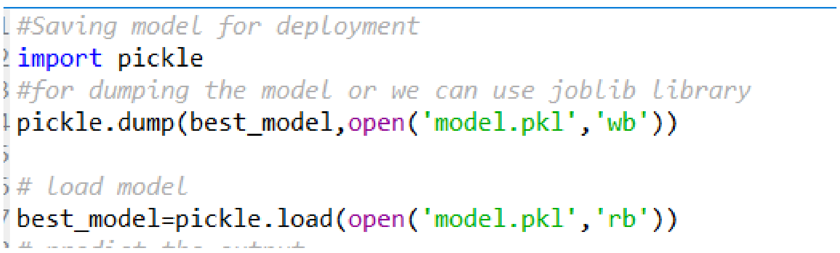
Saving model for deployment:-



```
In [26]:
print(best_model.predict([[15.300,25.27,102.40,732.4,0.10820,0.16970,0.16830,0.0
87510,0.1926,0.06540,1.0950,0.9053,8.589,153.40,0.006399,0.04904,0.05373,0.01587
,0.03003,0.006193,20.27,36.71,149.30,1269.0,0.1641,0.6110,0.63350,0.20240,0.4027
,0.09876]]))
[1]
```

CHAPTER-7

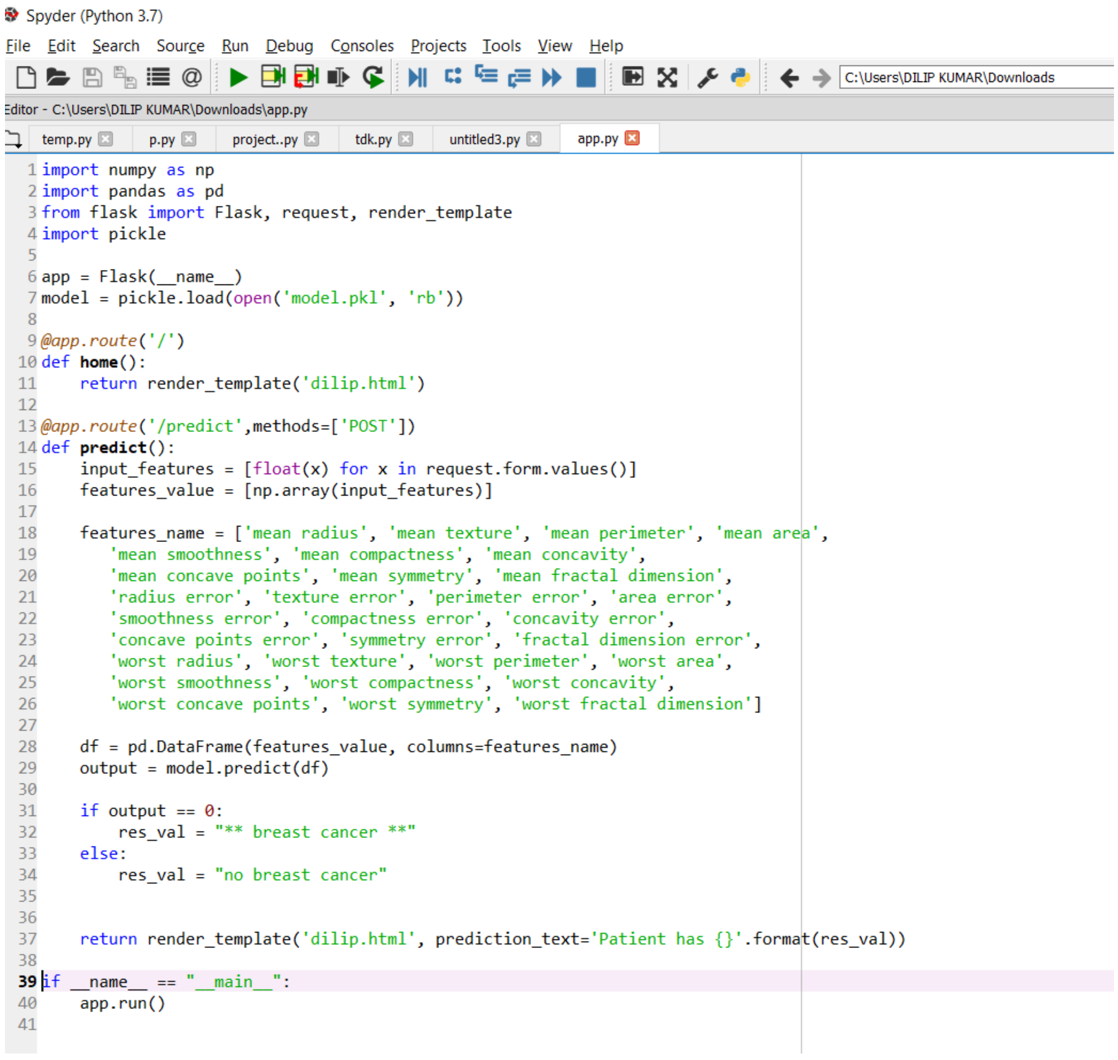
MODEL DEPLOYMENT



```python
#Saving model for deployment
import pickle
#for dumping the model or we can use joblib library
pickle.dump(best_model,open('model.pkl','wb'))

# load model
best_model=pickle.load(open('model.pkl','rb'))
```

Now are model is dump into pickle file.now its time for the flask for the model to deploy.

Now we have to switch towards sublime text editor for the deployment.the main aim is to used html,css with flask in it.

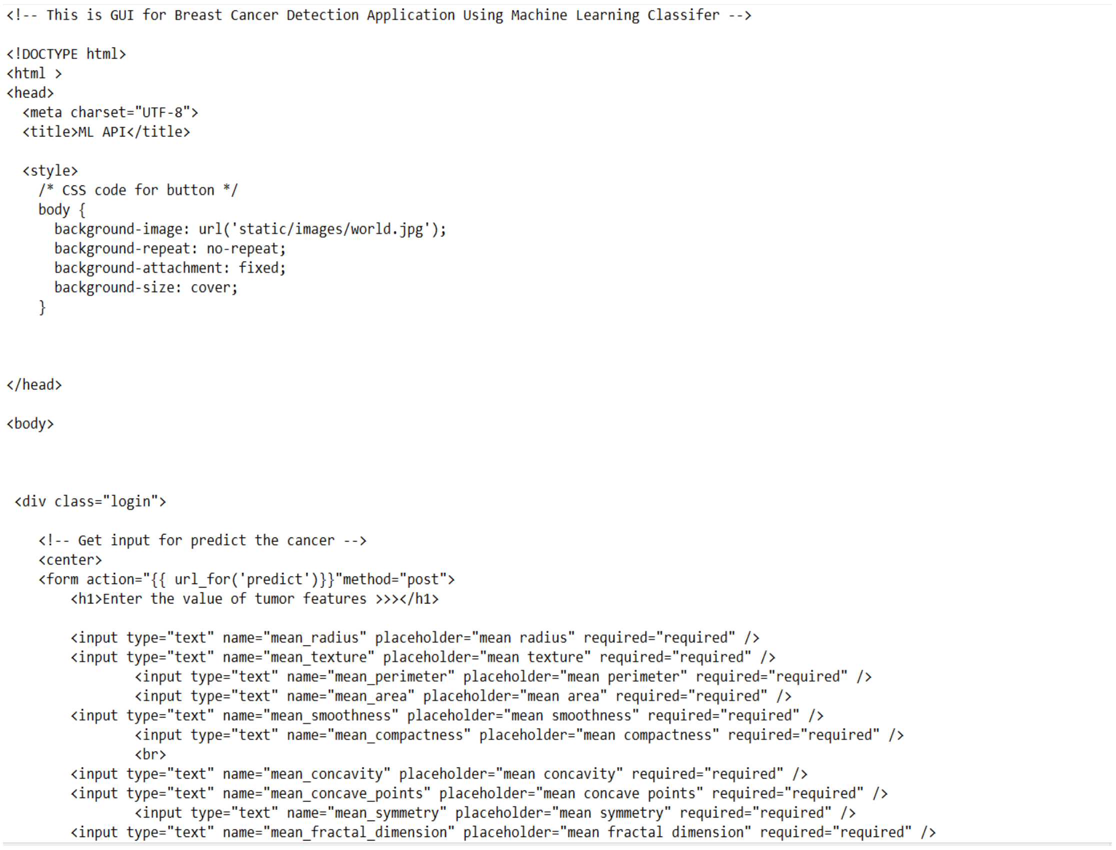


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\DILIP KUMAR\Downloads
Editor - C:\Users\DILIP KUMAR\Downloads\app.py
temp.py
p.py
project..py
tdk.py
untitled3.py
app.py

```python
import numpy as np
import pandas as pd
from flask import Flask, request, render_template
import pickle

app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('dilip.html')

@app.route('/predict',methods=['POST'])
def predict():
    input_features = [float(x) for x in request.form.values()]
    features_value = [np.array(input_features)]

    features_name = ['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']

    df = pd.DataFrame(features_value, columns=features_name)
    output = model.predict(df)

    if output == 0:
        res_val = "** breast cancer **"
    else:
        res_val = "no breast cancer"


    return render_template('dilip.html', prediction_text='Patient has {}'.format(res_val))

if __name__ == "__main__":
    app.run()
```

The code here depicts the loading of dumb model and then we are accessing the index.html file for the home page which we can discuss further.

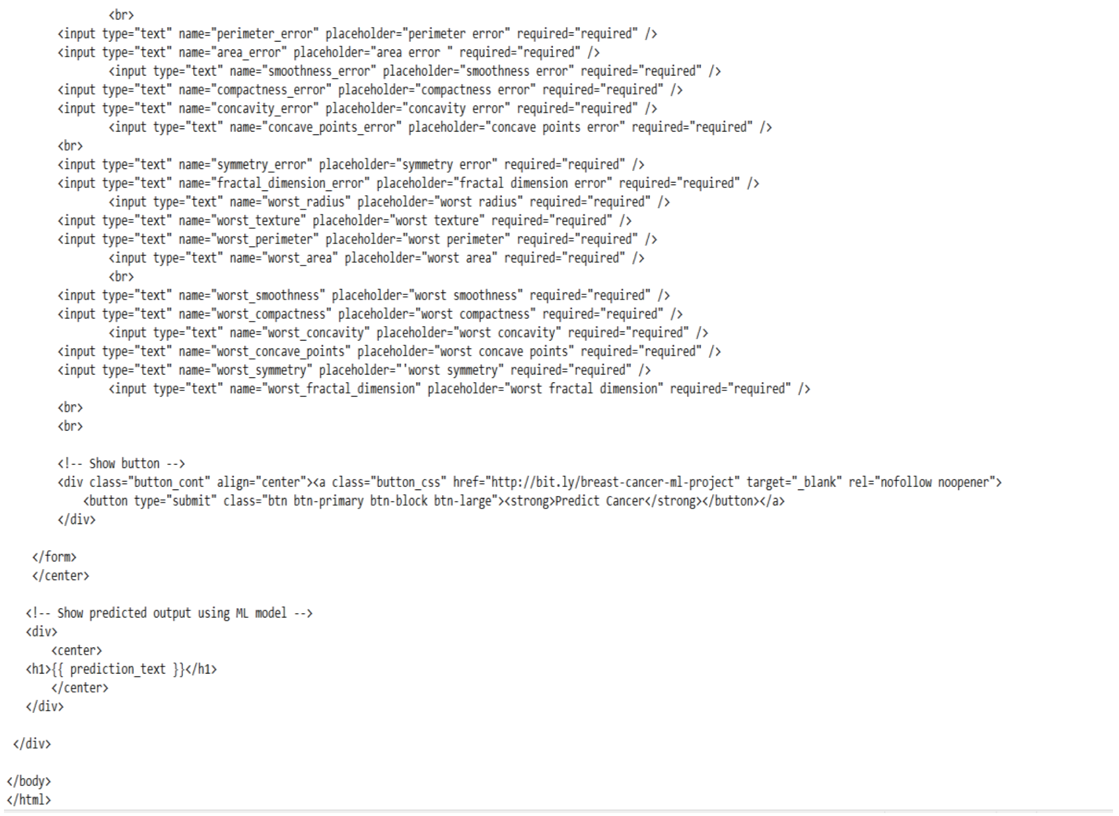
then we have the predict function which will be implemented when we will enter the input 30 constraints as an input in the box and then array of size 30 goes to the data frame for the prediction and return will make the after.html file which tell about the output as an tumor malignant and benign according to the value input by user and new webpage open with this classification.



```
<!-- This is GUI for Breast Cancer Detection Application Using Machine Learning Classifer -->

<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>ML API</title>

  <style>
    /* CSS code for button */
    body {
      background-image: url('static/images/world.jpg');
      background-repeat: no-repeat;
      background-attachment: fixed;
      background-size: cover;
    }


</head>

<body>



 <div class="login">

    <!-- Get input for predict the cancer -->
    <center>
    <form action="{{ url_for('predict')}}"method="post">
        <h1>Enter the value of tumor features >>></h1>

        <input type="text" name="mean_radius" placeholder="mean radius" required="required" />
        <input type="text" name="mean_texture" placeholder="mean texture" required="required" />
                <input type="text" name="mean_perimeter" placeholder="mean perimeter" required="required" />
                <input type="text" name="mean_area" placeholder="mean area" required="required" />
        <input type="text" name="mean_smoothness" placeholder="mean smoothness" required="required" />
                <input type="text" name="mean_compactness" placeholder="mean compactness" required="required" />
                <br>
        <input type="text" name="mean_concavity" placeholder="mean concavity" required="required" />
        <input type="text" name="mean_concave_points" placeholder="mean concave points" required="required" />
                <input type="text" name="mean_symmetry" placeholder="mean symmetry" required="required" />
        <input type="text" name="mean_fractal_dimension" placeholder="mean fractal dimension" required="required" />
```

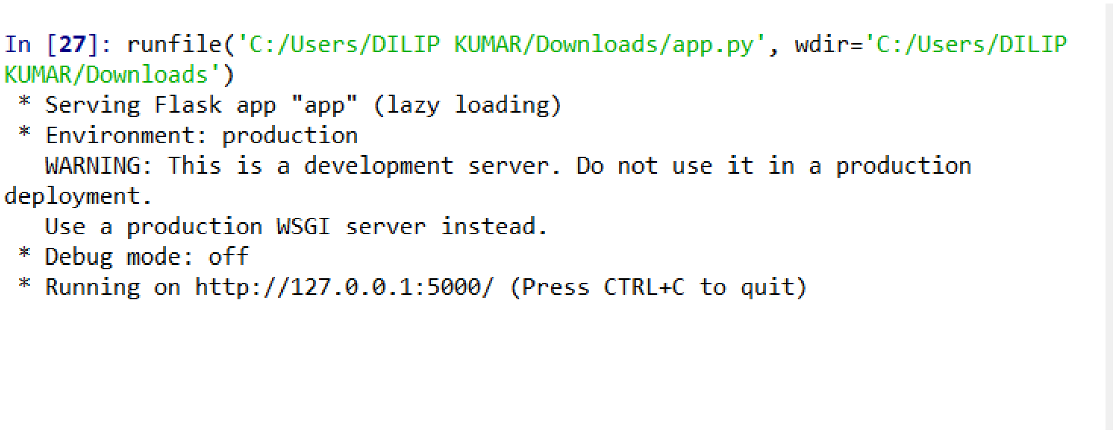
Then , we make a placeholder which will help us to store the input value and display placeholder name on webpage. Then,we will make the **“Click here to predict ”**button for the prediction. and then header for the name. the code next to it depicts about the displaying the icons of the social media with the font size and the specific color. Each specific icon is hyperlinked to my social media handle and then end of the body.

Now comes the file which will come and depicts output of the inbuild functions.



```
                <br>
        <input type="text" name="perimeter_error" placeholder="perimeter error" required="required" />
        <input type="text" name="area_error" placeholder="area error " required="required" />
                <input type="text" name="smoothness_error" placeholder="smoothness error" required="required" />
        <input type="text" name="compactness_error" placeholder="compactness error" required="required" />
        <input type="text" name="concavity_error" placeholder="concavity error" required="required" />
                <input type="text" name="concave_points_error" placeholder="concave points error" required="required" />
        <br>
        <input type="text" name="symmetry_error" placeholder="symmetry error" required="required" />
        <input type="text" name="fractal_dimension_error" placeholder="fractal dimension error" required="required" />
                <input type="text" name="worst_radius" placeholder="worst radius" required="required" />
        <input type="text" name="worst_texture" placeholder="worst texture" required="required" />
        <input type="text" name="worst_perimeter" placeholder="worst perimeter" required="required" />
                <input type="text" name="worst_area" placeholder="worst area" required="required" />
                <br>
        <input type="text" name="worst_smoothness" placeholder="worst smoothness" required="required" />
        <input type="text" name="worst_compactness" placeholder="worst compactness" required="required" />
                <input type="text" name="worst_concavity" placeholder="worst concavity" required="required" />
        <input type="text" name="worst_concave_points" placeholder="worst concave points" required="required" />
        <input type="text" name="worst_symmetry" placeholder="'worst symmetry" required="required" />
                <input type="text" name="worst_fractal_dimension" placeholder="worst fractal dimension" required="required" />
        <br>
        <br>

        <!-- Show button -->
        <div class="button_cont" align="center"><a class="button_css" href="http://bit.ly/breast-cancer-ml-project" target="_blank" rel="nofollow noopener">
            <button type="submit" class="btn btn-primary btn-block btn-large"><strong>Predict Cancer</strong></button></a>
        </div>

    </form>
    </center>

   <!-- Show predicted output using ML model -->
   <div>
       <center>
   <h1>{{ prediction_text }}</h1>
       </center>
   </div>

 </div>

</body>
</html>
```

The after.html file have background to the webapge as image background and then it will have the **PREDICTION will be 0 or 1** according to the data entered and image**.**



```
In [27]: runfile('C:/Users/DILIP KUMAR/Downloads/app.py', wdir='C:/Users/DILIP
KUMAR/Downloads')
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Now our model is ready to run command first change directory to the folder where we have the run **python app.py**

Then ,it will be return on local system http://127.0.0.1:5000/

CHAPTER 8: REFERENCES

- https://towardsdatascience.com/building-a-simple-machine-learning-model-on-cancer-data-eca4b3b99fa3https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html
- https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score
- https://scikit-learn.org/stable/model_selection.html#model-selection
- https://machinelearningmastery.com/classification-for-machine-learning