BREAST CANCER PREDICTION

Machine Learning in Python

Project Report
by
Sarma Bhamidipati

GitHub URL

https://github.com/sarmabhamidipati/UCDPA_SarmaBhamidipati

1.1 Abstract

Breast Cancer is one of the most common type of all cancers and also represents one of the main diseases that make for a significant number of deaths each year. The early diagnosis of Breast Cancer can improve prognosis and the chance of survival significantly as it can promote timely clinical treatment to patients. Also, correct diagnosis and classification of Breast Cancer is necessary to eradicate unnecessary treatment to patients with benign tumours.

Machine learning and data mining methods are an effective way to classify and forecast data, especially in the medical arena wherein such methods are helpful in correct decision making for diagnosis.

This report aims to observe which features are most helpful in the prediction of **malignant** or **benign** cancer. For this, I have used the 'supervised' machine learning methods to fit a function that can predict the discreet class of new input.

In this analysis, a performance comparison between different machine learning algorithms: **Logistic Regression, Decision Tree Classifier, Random Forest Classifier, K-Nearest Neighbour Classifier (KNN), and Support Vector Classification (SVC)** on the Wisconsin Breast Cancer(original) datasets is conducted. The main objective is to access the correctness in classifying data with respect to efficiency and effectiveness of each algorithm in terms of accuracy and precision. I have used ***PyCharm*** to work on this dataset. The experimental results show that **Random Forest Classifier** gives the highest accuracy ((97.9%).

Breast cancer prediction contains below features

1. Importing required libraries (Pandas and other machine learning related libraries)
2. Data loading using pandas
3. Understanding data types, size, rows, and columns of data frame in step2
4. Find the missing values.
5. Data cleaning – Drop the columns or features which are not required.
6. Observing categorical data and get the count of rows from each column.
7. Get a count of Malignant(M) vs Benign(B) cells data.
8. Visualize count plot of Malignant(M) vs Benign(B) using seaborn.
9. Transform categorical data (Diagnosis) to numerical format using Label Encoder.
10. Visualize pair plot, and heatmap using seaborn.
11. Create Training and Test Data.
12. Standardization and Normalization.
13. ML supervised learning models and fit the data.
14. Models used in this process are Logistic Regression Model, Decision Tree Classifier, Random Forest Classifier, k-Nearest Neighbours, and Support Vector Classification
15. Data Prediction and classification
16. Predict and get the accuracy score for all the models used in step 14.
17. Create a new data frame to print and compare accuracy score found in step 14. It is done using lists and dictionaries.
18. Use Hyper-Parameter Tuning model for with high score found in step 14 to see if we can boost its score. It is done using GridSearchCV

To demonstrate web scraping, regular expression, and merge data frame features, I have used the **weather forecast data** for better results.

Weather forecast prediction contains below features

1. Get the web page containing the forecast
2. Create soup class to parse
3. Get 14-day forecast
4. Create a list of day, Temp, Weather, Visibility and Temperature in Number
5. create a 14-day weather forecast data frame based on lists created
6. Use regular expression to extract only temperature and convert this to a number.
7. create another data frame with temperature in number.
8. Merge data frames created in step 5 and step 7 using pandas concat.
9. Display weather forecast data.

1.2 Introduction

Breast cancer is the most diagnosed cancer among women worldwide, accounting for 1 in 4 cases (GLOBOCAN 2020). It is the most frequent cancer amongst both sexes and is the leading cause of death from cancer in women. The estimated 2.3 million new cases indicate that one in every 8 cancers diagnosed in 2020 is breast cancer. In 2020, there were an estimated 684,996 deaths from breast cancer, with a disproportionate number of these deaths occurring in low-resource settings (UICC).

To tackle the growing breast cancer burden, it is critical that improvements are made in access to early detection, timely access to treatment and care, palliative and survivorship care and comprehensive data collection.

Information and Communications Control (ICT) can play potential roles in cancer care. Big data has become synonymous with data mining, business analytics and business intelligence and has made a big change in BI from reporting and decision making to prediction results. Data mining methods when used especially in the medical science arena become very useful and important in the prediction and diagnosis of serious illness thereby improving the healthcare quality in saving people’s lives.

There are many algorithms for classification and prediction of breast care outcomes. This report gives a comparison between the performance of five classifiers***: SVM, Logistic Regression, Decision Tree, k-Nearest Neighbour(k-NN) and Random Forest (RF) Classifier*** which are among the most influential data mining algorithms in the research community. My aim is to evaluate the efficiency of these algorithms in terms of their accuracy and precision for the prediction of malignant or benign cancer.

Also, to demonstrate web scraping, regular expression, and merge the data frame features, I have used the weather forecast data to obtain better results.

1.3 Dataset

The following are the datasets used.

Data Set	Data Source	Reason
Breast Cancer Wisconsin Data	https://www.kaggle.com/uciml/breast-cancer-wisconsin-data	Need to import data using Pandas so I have decided to use this source
NY Weather Forecast Data	https://www.timeanddate.com/weather/usa/new-york/ext	This is used to only demonstrate web scraping, regular expression, and merge data frame features

1.4 Implementation Process

My **main.py** python file contains code for both breast cancer prediction and weather forecast

Breast Cancer Prediction Implementation Process

This section outlines the implementation process for breast cancer prediction with brief description of each element needed in the prediction process.

1. Import required libraries for data analysis, cleaning, visualization, supervised machine learning and hyper tuning

2. **Data Processing**

   2.1 : **Data Loading**: - Load the data using pandas. Understand it's data, data type, shape, and columns

   2.2 **Data Cleaning**: - Check missing values and clean the data. Get the new count, shape, and data. Drop the features/columns which are not required for this prediction.

   2.3 Get the unique row count of its features and get a count of Malignant(M) vs Benign(B) cells data

   2.4 **Data Visualization**: - Visualize count plot of Malignant(M) vs Benign(B) using seaborn

   2.5 **Data Transformation**: - Transform categorical data (Diagnosis) to numerical format using Label Encoder

   2.6 **Data Visualization**: - Visualize pair plot, and heatmap using seaborn.

3. **Training and Test Data**: - Split the data set into independent/Features/Predictor Variables (X) and dependent data/Target/Response Variable(y)
   Split the data into 75% training and 25% test data

4. **Standardization and Normalization**: - Scale and Transform independent data.

5. Used **Supervised Machine Learning models** to fit the training data and make predictions on the test set. Models used in this process are Logistic Regression Model, Decision Tree Classifier, Random Forest Classifier, k-Nearest Neighbours, and Support Vector Classification

6. Get the accuracy score on the test set for the supervised ML models used. Get the score in highest to least order based on the models used in step 5.

7. Using **Hyper tuning model using GridSearchCV** to boost the score for the highest accuracy score found from the models used.

Weather Forecast Implementation Process

1. Import required libraries for web scraping, parsing, regular expression, and data frame.

2. **Data Processing**

   2.1 : **Scrape**: - Get the web scrape page containing the forecast data using requests.get

   2.2 **Parse**: - Create soup class to parse the page.

   2.3 **Data Transformation**: - Create a weather list consists of weather, visibility, temperature, and temperature in number.
   2.4 **Create Data Frame**: - create a 14-day weather forecast data frame based on lists created

   2.5 Use regular expression to extract only temperature and convert this to a number.

   2.6 create another data frame with temperature in number.

   2.7 Merge data frames created in step 4 and step 6 using pandas concat

   2.8 Display final weather forecast data which now contains temperature in characters as well as in number format.

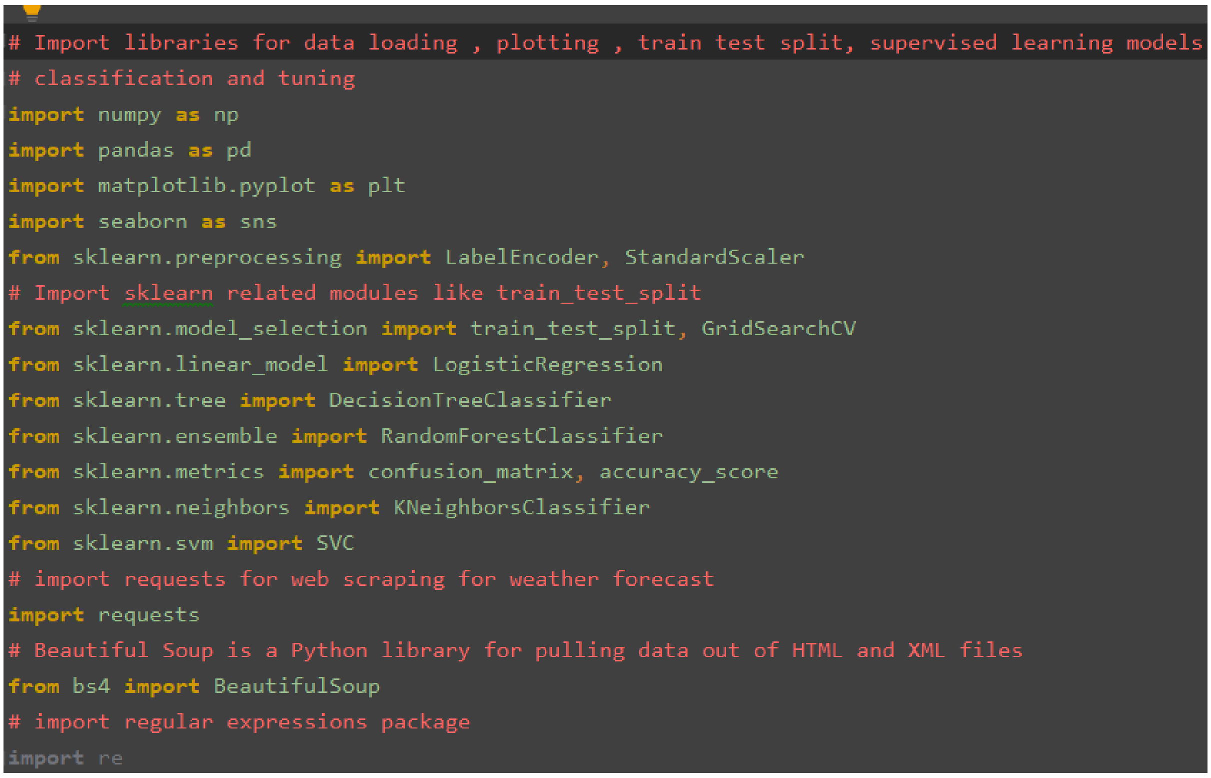
1.5 Results

This section includes the results of the breast cancer prediction implementation process in detail including the import libraries, prediction scores and visualisation charts.

The entire code is written in ***main.py*** which loads all the required libraries for both the projects.

Breast Cancer Prediction Implementation Process Results

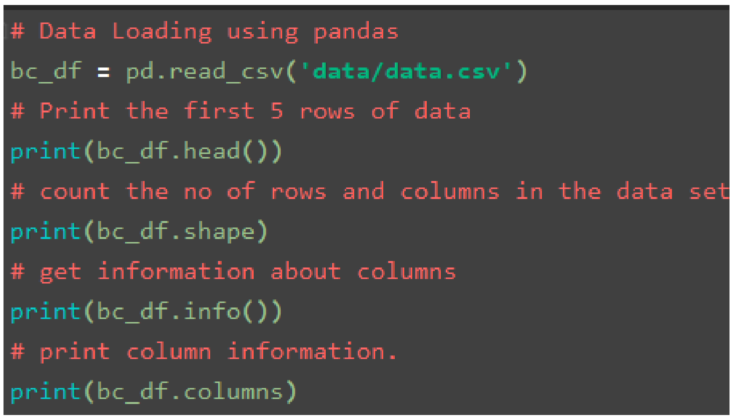
1. Import the libraries
   It imports libraries related for both the prediction projects.



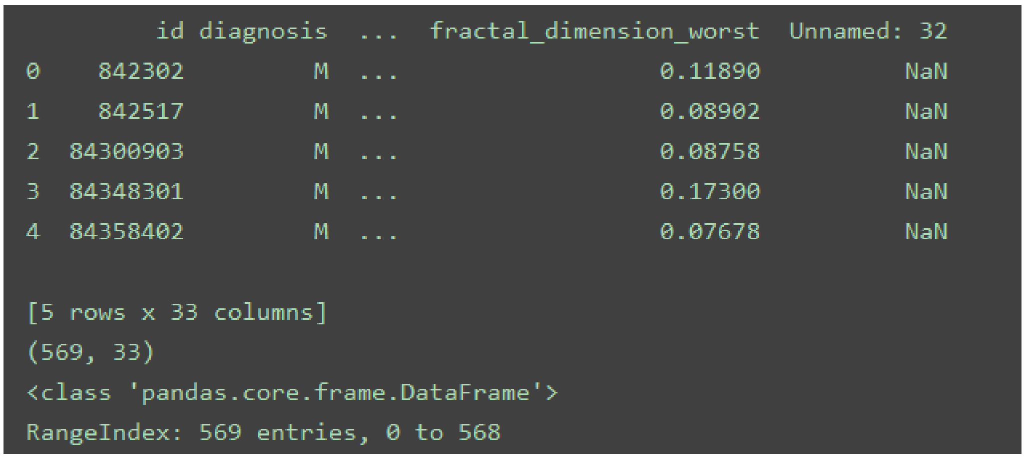
```
# Import libraries for data loading , plotting , train test split, supervised learning models
# classification and tuning
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
# Import sklearn related modules like train_test_split
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
# import requests for web scraping for weather forecast
import requests
# Beautiful Soup is a Python library for pulling data out of HTML and XML files
from bs4 import BeautifulSoup
# import regular expressions package
import re
```

2. Data Loading using Pandas

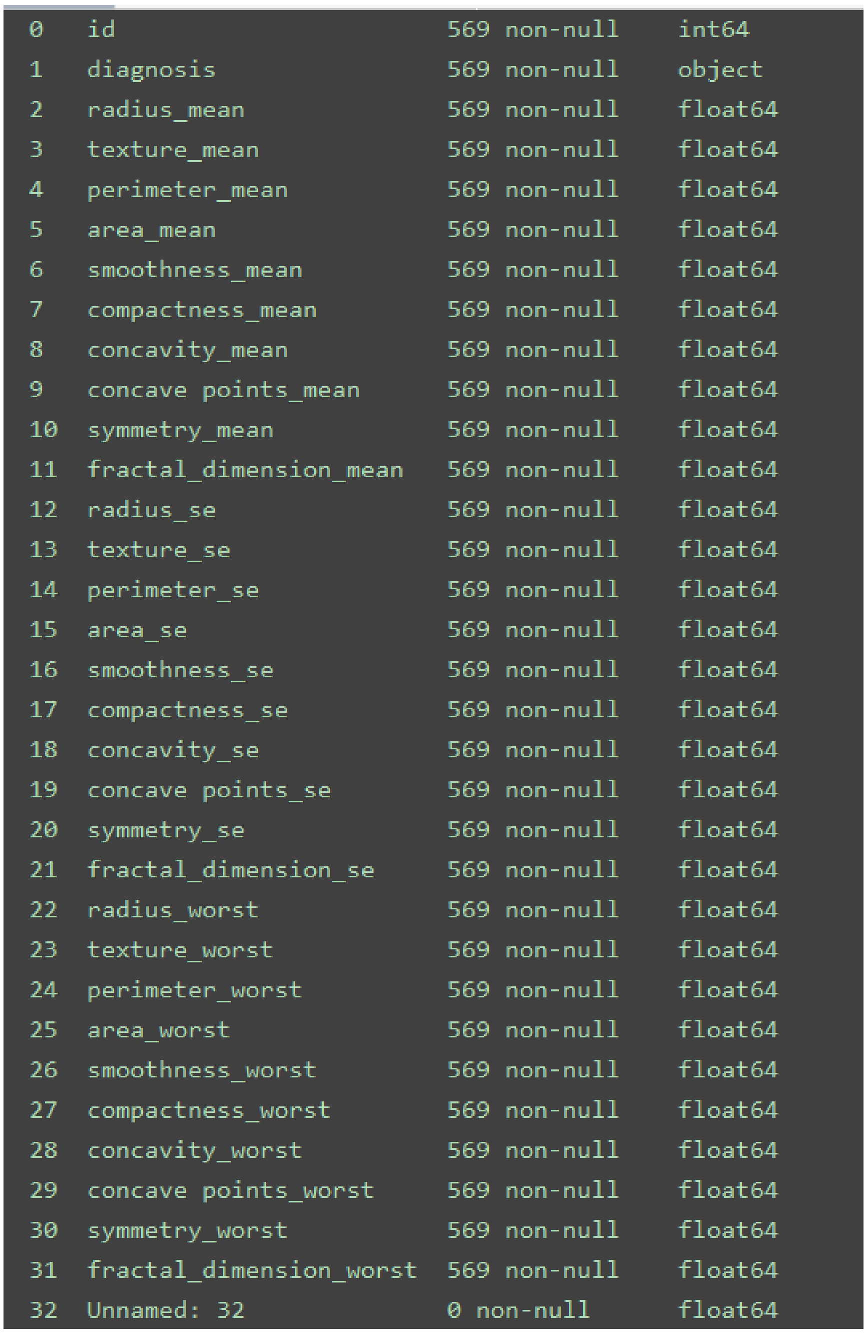
   Read the data which is data.csv using pd. read_csv. Display first 5 rows of data , get the count of rows and columns, information about columns



```
# Data Loading using pandas
bc_df = pd.read_csv('data/data.csv')
# Print the first 5 rows of data
print(bc_df.head())
# count the no of rows and columns in the data set
print(bc_df.shape)
# get information about columns
print(bc_df.info())
# print column information.
print(bc_df.columns)
```



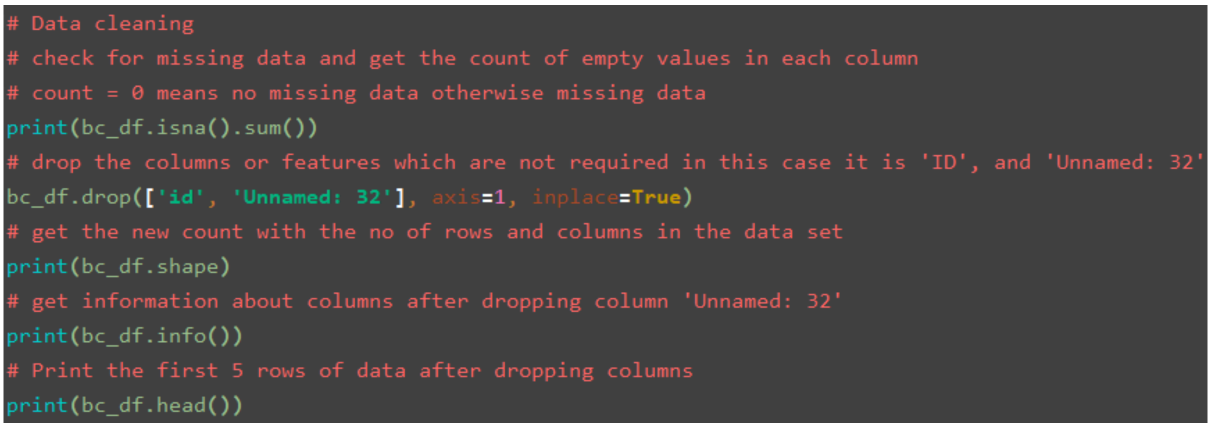
```
         id diagnosis  ...  fractal_dimension_worst  Unnamed: 32
0    842302         M  ...                  0.11890          NaN
1    842517         M  ...                  0.08902          NaN
2  84300903         M  ...                  0.08758          NaN
3  84348301         M  ...          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

[5 rows x 33 columns]
(569, 33)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
```



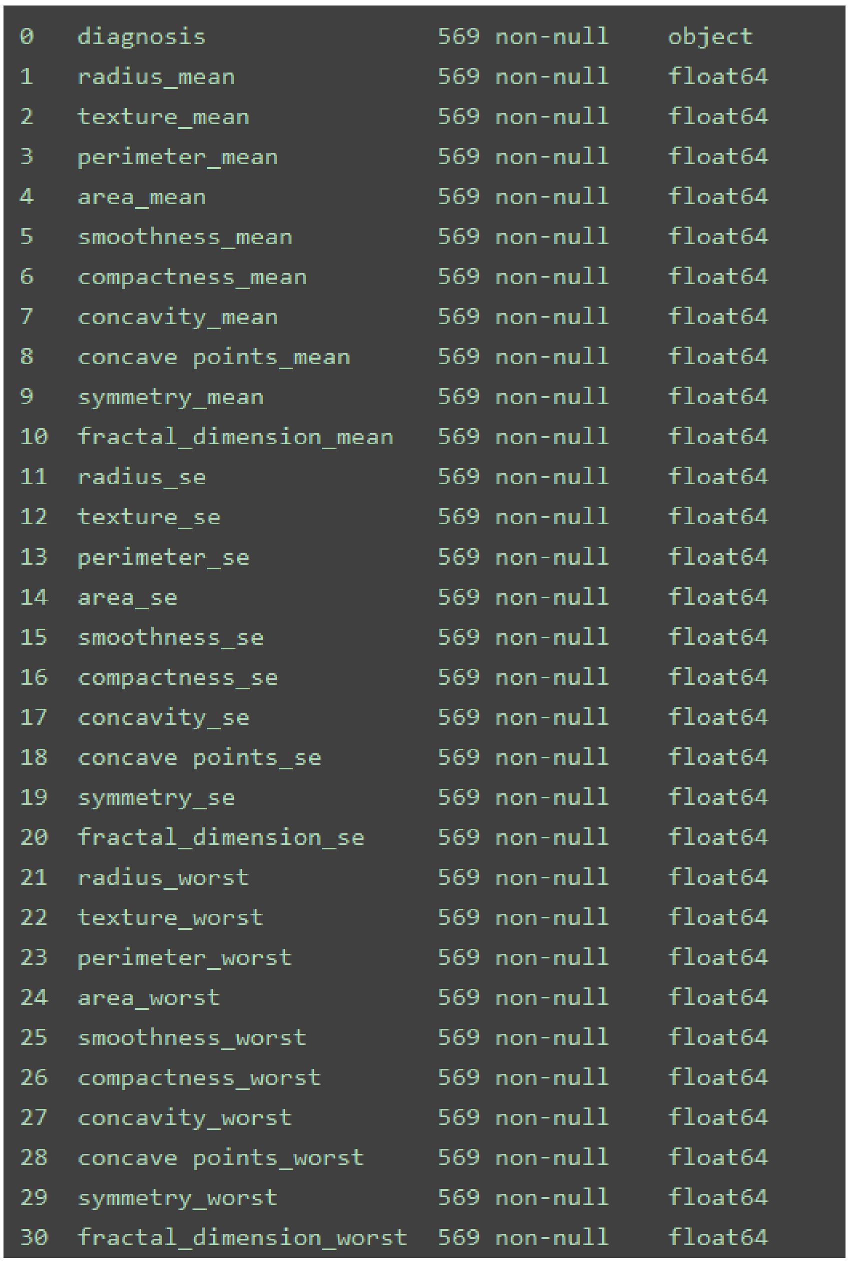
```
0   id                       569 non-null    int64
1   diagnosis                569 non-null    object
2   radius_mean              569 non-null    float64
3   texture_mean             569 non-null    float64
4   perimeter_mean           569 non-null    float64
5   area_mean                569 non-null    float64
6   smoothness_mean          569 non-null    float64
7   compactness_mean         569 non-null    float64
8   concavity_mean           569 non-null    float64
9   concave points_mean      569 non-null    float64
10  symmetry_mean            569 non-null    float64
11  fractal_dimension_mean   569 non-null    float64
12  radius_se                569 non-null    float64
13  texture_se               569 non-null    float64
14  perimeter_se             569 non-null    float64
15  area_se                  569 non-null    float64
16  smoothness_se            569 non-null    float64
17  compactness_se           569 non-null    float64
18  concavity_se             569 non-null    float64
19  concave points_se        569 non-null    float64
20  symmetry_se              569 non-null    float64
21  fractal_dimension_se     569 non-null    float64
22  radius_worst             569 non-null    float64
23  texture_worst            569 non-null    float64
24  perimeter_worst          569 non-null    float64
25  area_worst               569 non-null    float64
26  smoothness_worst         569 non-null    float64
27  compactness_worst        569 non-null    float64
28  concavity_worst          569 non-null    float64
29  concave points_worst     569 non-null    float64
30  symmetry_worst           569 non-null    float64
31  fractal_dimension_worst  569 non-null    float64
32  Unnamed: 32              0 non-null      float64
```

3. Data cleaning

   Check for missing data and drop the columns or features which are not required for this prediction. For this I have used insa() function.
   Upon on investigation, it was found that column ‘Unnamed: 32’ does not have any values. Also found that column ‘ID’ does not server any purpose except for the identification of the row. So, I have decided to drop these two columns from the data set.

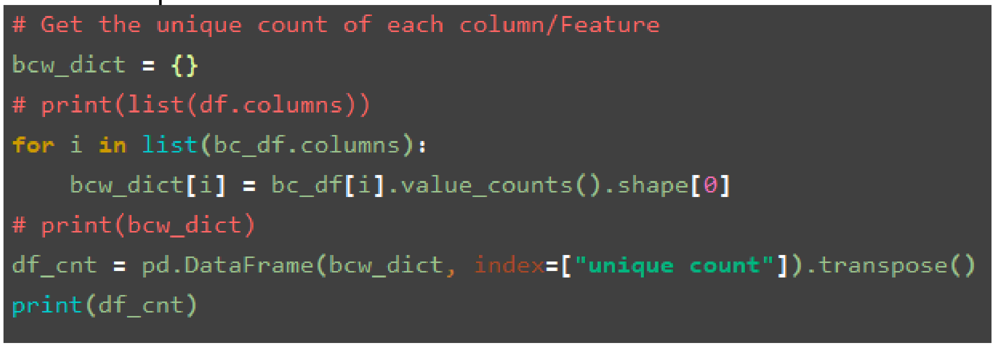


```
# Data cleaning
# check for missing data and get the count of empty values in each column
# count = 0 means no missing data otherwise missing data
print(bc_df.isna().sum())
# drop the columns or features which are not required in this case it is 'ID', and 'Unnamed: 32'
bc_df.drop(['id', 'Unnamed: 32'], axis=1, inplace=True)
# get the new count with the no of rows and columns in the data set
print(bc_df.shape)
# get information about columns after dropping column 'Unnamed: 32'
print(bc_df.info())
# Print the first 5 rows of data after dropping columns
print(bc_df.head())
```

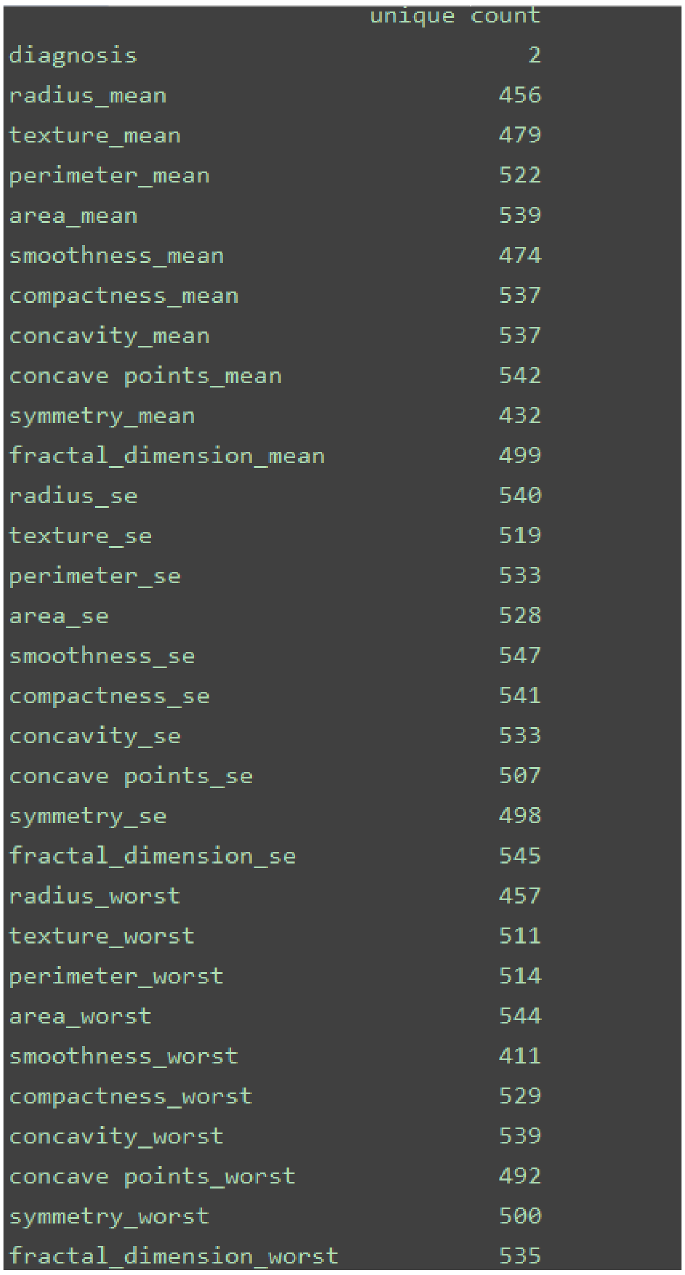


```
 0   diagnosis                569 non-null    object
 1   radius_mean              569 non-null    float64
 2   texture_mean             569 non-null    float64
 3   perimeter_mean           569 non-null    float64
 4   area_mean                569 non-null    float64
 5   smoothness_mean          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
```

4. Get the unique values of each column/feature

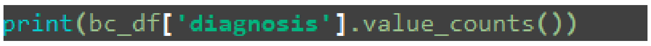


```python
# Get the unique count of each column/Feature
bcw_dict = {}
# print(list(df.columns))
for i in list(bc_df.columns):
    bcw_dict[i] = bc_df[i].value_counts().shape[0]
# print(bcw_dict)
df_cnt = pd.DataFrame(bcw_dict, index=["unique count"]).transpose()
print(df_cnt)
```

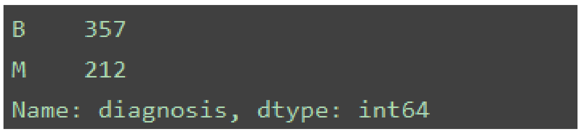


```
                         unique count
diagnosis                           2
radius_mean                       456
texture_mean                      479
perimeter_mean                    522
area_mean                         539
smoothness_mean                   474
compactness_mean                  537
concavity_mean                    537
concave points_mean               542
symmetry_mean                     432
fractal_dimension_mean            499
radius_se                         540
texture_se                        519
perimeter_se                      533
area_se                           528
smoothness_se                     547
compactness_se                    541
concavity_se                      533
concave points_se                 507
symmetry_se                       498
fractal_dimension_se              545
radius_worst                      457
texture_worst                     511
perimeter_worst                   514
area_worst                        544
smoothness_worst                  411
compactness_worst                 529
concavity_worst                   539
concave points_worst              492
symmetry_worst                    500
fractal_dimension_worst           535
```

5. Get a count of the number of Malignant(M) or Benign(B) cells. Benign means no cancer.

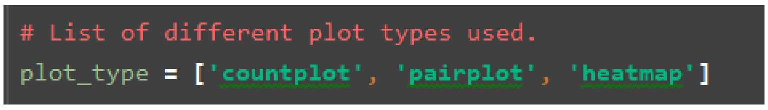


```
print(bc_df['diagnosis'].value_counts())
```



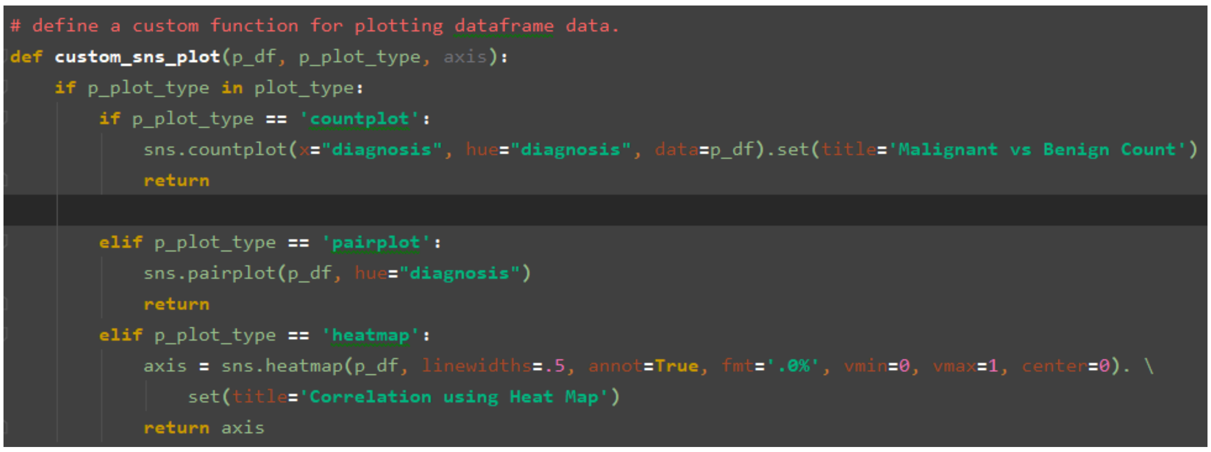
```
B    357
M    212
Name: diagnosis, dtype: int64
```

6. Created a list for different plot types used.



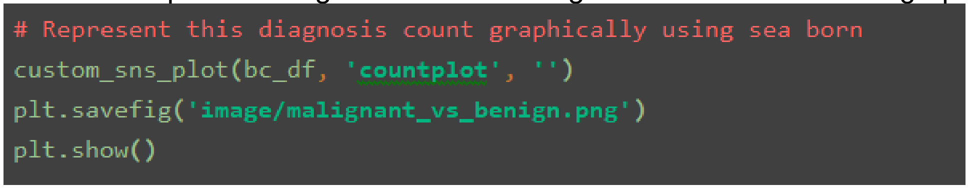
```
# List of different plot types used.
plot_type = ['countplot', 'pairplot', 'heatmap']
```

7. Created a custom function for plotting data frame data.

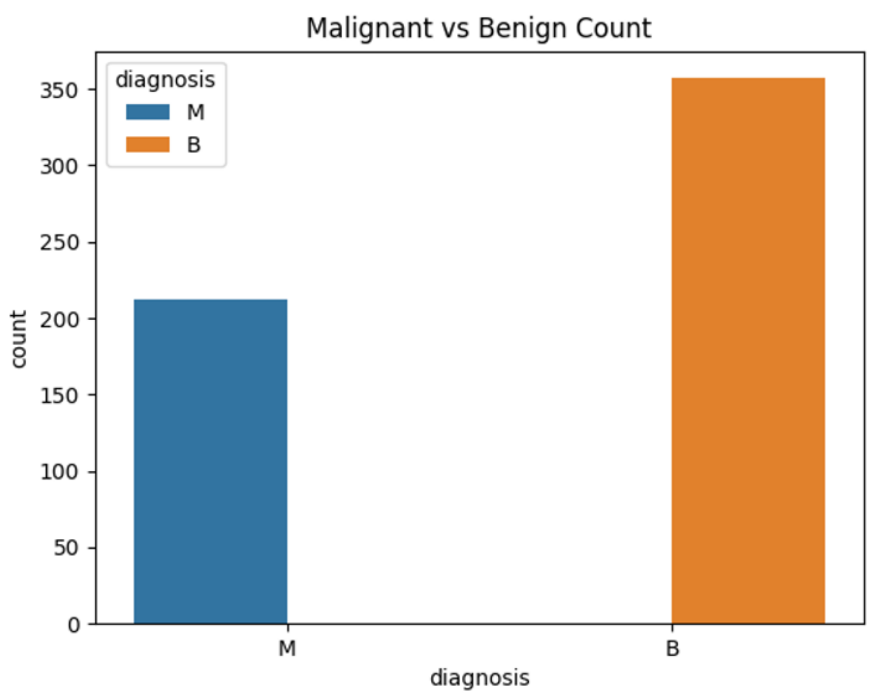


```
# define a custom function for plotting dataframe data.
def custom_sns_plot(p_df, p_plot_type, axis):
    if p_plot_type in plot_type:
        if p_plot_type == 'countplot':
            sns.countplot(x="diagnosis", hue="diagnosis", data=p_df).set(title='Malignant vs Benign Count')
            return

        elif p_plot_type == 'pairplot':
            sns.pairplot(p_df, hue="diagnosis")
            return
        elif p_plot_type == 'heatmap':
            axis = sns.heatmap(p_df, linewidths=.5, annot=True, fmt='.0%', vmin=0, vmax=1, center=0). \
                set(title='Correlation using Heat Map')
            return axis
```

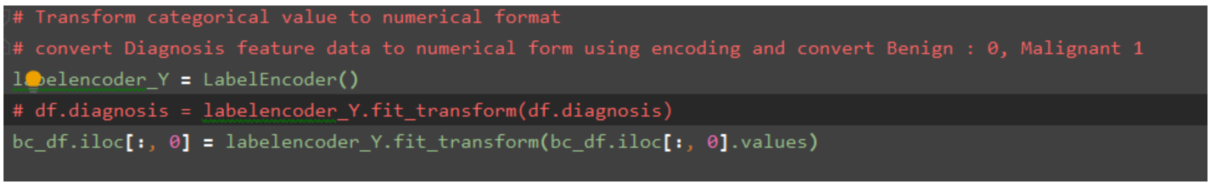
8. Plot a count plot for ‘Diagnosis’ feature using sea born and save that graph.



```
# Represent this diagnosis count graphically using sea born
custom_sns_plot(bc_df, 'countplot', '')
plt.savefig('image/malignant_vs_benign.png')
plt.show()
```

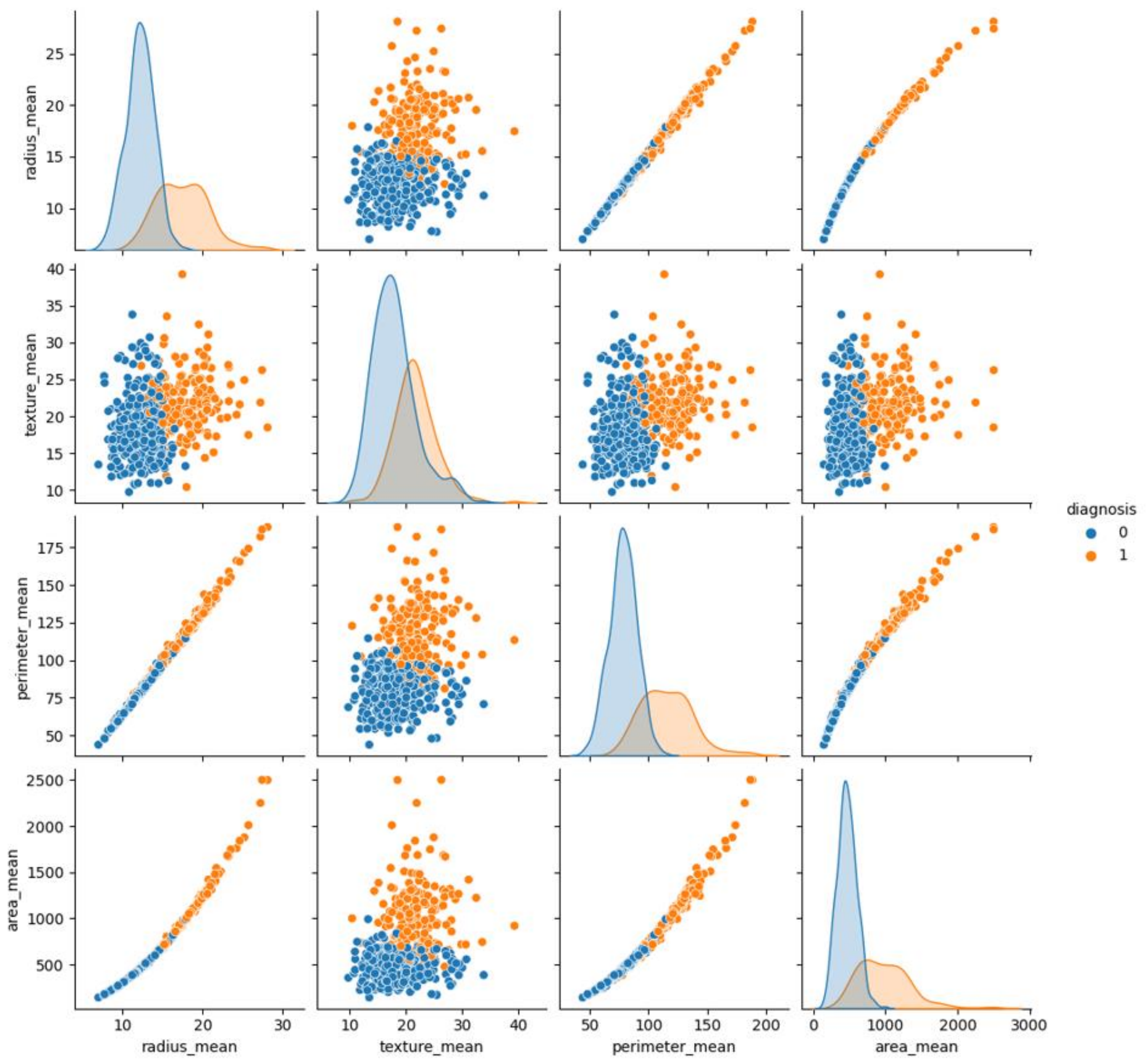


9. Transform the categorical values to numerical format using LabelEncoder() function. Here I have converted diagnosis feature.

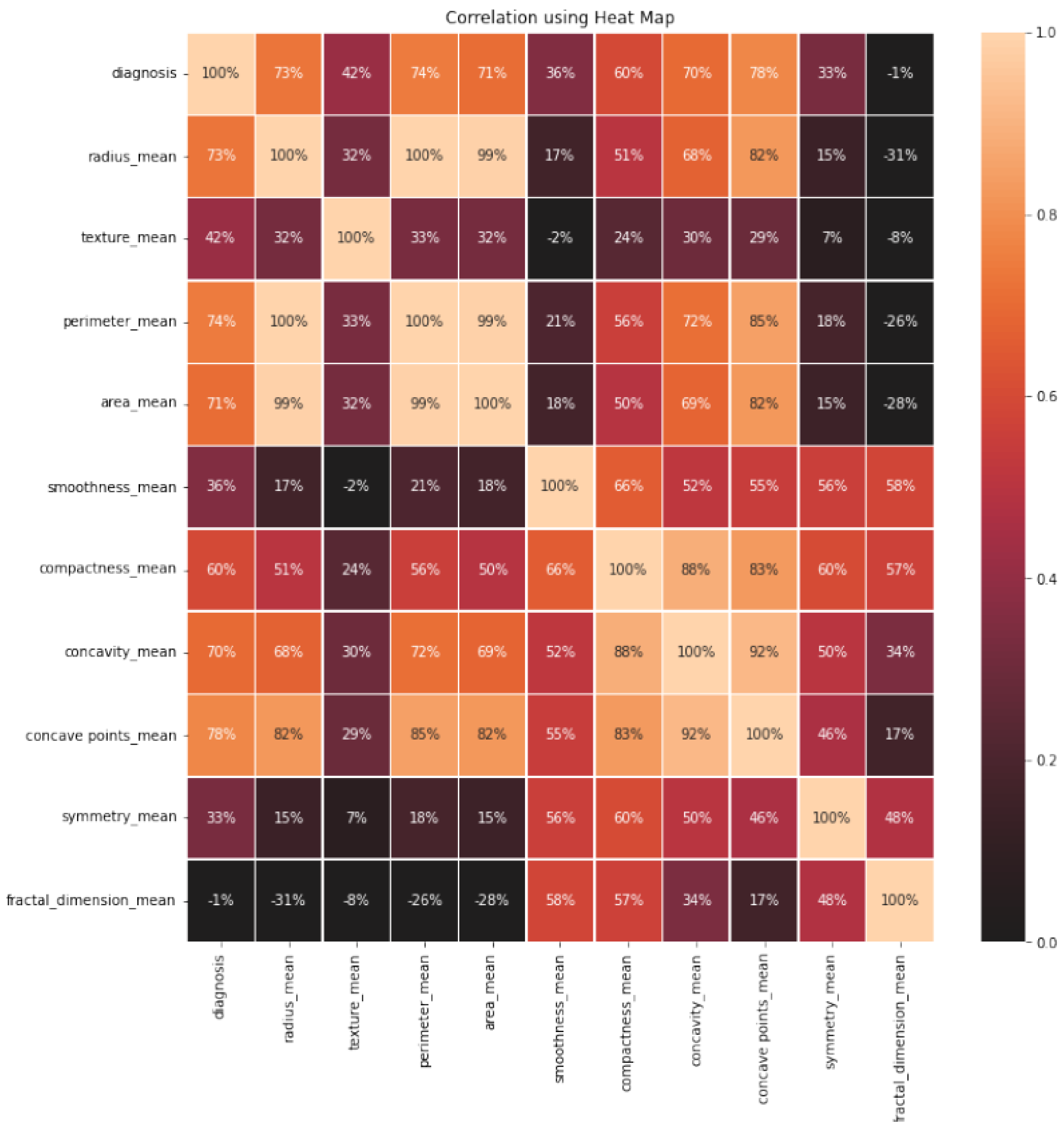


```
# Transform categorical value to numerical format
# convert Diagnosis feature data to numerical form using encoding and convert Benign : 0, Malignant 1
labelencoder_Y = LabelEncoder()
# df.diagnosis = labelencoder_Y.fit_transform(df.diagnosis)
bc_df.iloc[:, 0] = labelencoder_Y.fit_transform(bc_df.iloc[:, 0].values)
```

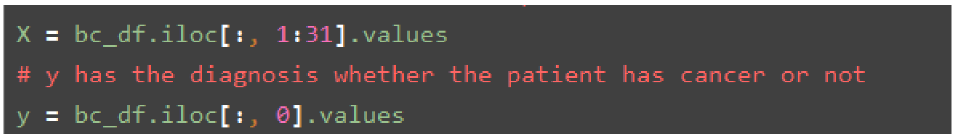
10. Created a pair plot using sea born for features
    diagnosis, radius_mean, texture_mean, perimeter_mean, and area_mean



11. Get the correlation of columns for plotting heat map using sea born. Here I have considered only first 11 columns (diagnosis to fractal_dimension_mean)

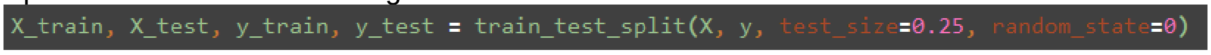


12. Split the data set into independent/Features/Predictor Variables (X) and dependent data/Target/Response Variable(y)
    X has the features of the cancer patients
    y has the diagnosis whether the patient has cancer or not



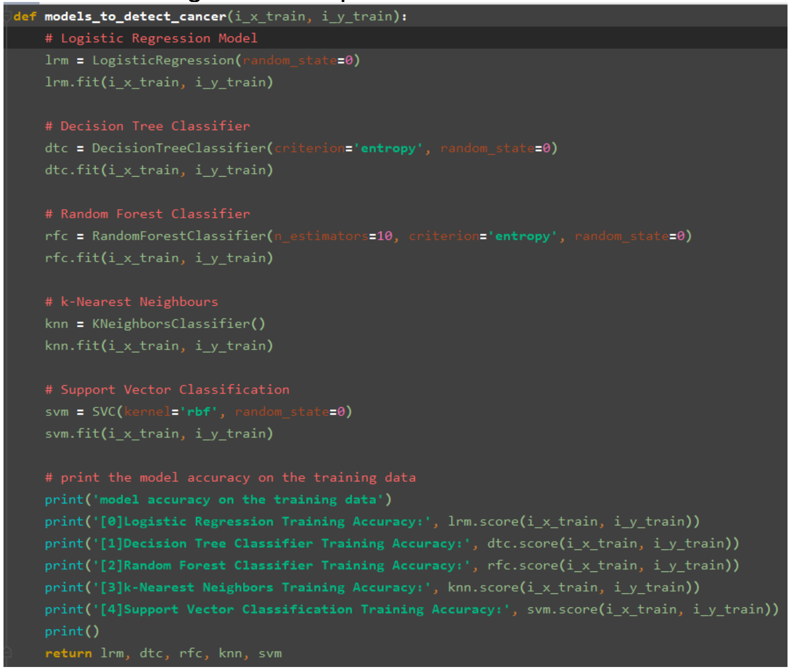
```
X = bc_df.iloc[:, 1:31].values
# y has the diagnosis whether the patient has cancer or not
y = bc_df.iloc[:, 0].values
```

13. Training And Test Data
    Split the data into 75% training and 25% test data



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

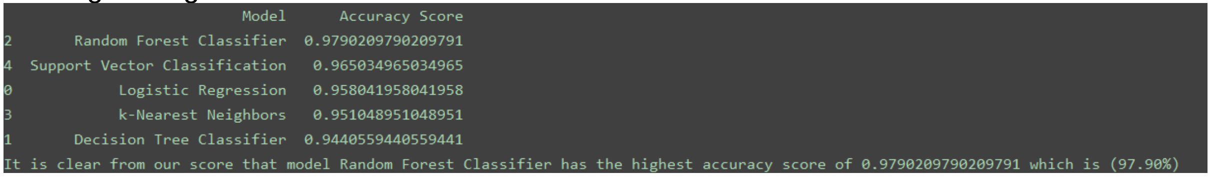
14. Created a custom function for supervised learning models to fit the data. Models which I have used for this prediction are Logistic Regression Model, Decision Tree Classifier, Random Forest Classifier, k-Nearest Neighbours, and Support Vector Classification.
    It takes training data as a parameter to fit the model.



```
def models_to_detect_cancer(i_x_train, i_y_train):
    # Logistic Regression Model
    lrm = LogisticRegression(random_state=0)
    lrm.fit(i_x_train, i_y_train)

    # Decision Tree Classifier
    dtc = DecisionTreeClassifier(criterion='entropy', random_state=0)
    dtc.fit(i_x_train, i_y_train)

    # Random Forest Classifier
    rfc = RandomForestClassifier(n_estimators=10, criterion='entropy', random_state=0)
    rfc.fit(i_x_train, i_y_train)

    # k-Nearest Neighbours
    knn = KNeighborsClassifier()
    knn.fit(i_x_train, i_y_train)

    # Support Vector Classification
    svm = SVC(kernel='rbf', random_state=0)
    svm.fit(i_x_train, i_y_train)

    # print the model accuracy on the training data
    print('model accuracy on the training data')
    print('[0]Logistic Regression Training Accuracy:', lrm.score(i_x_train, i_y_train))
    print('[1]Decision Tree Classifier Training Accuracy:', dtc.score(i_x_train, i_y_train))
    print('[2]Random Forest Classifier Training Accuracy:', rfc.score(i_x_train, i_y_train))
    print('[3]k-Nearest Neighbors Training Accuracy:', knn.score(i_x_train, i_y_train))
    print('[4]Support Vector Classification Training Accuracy:', svm.score(i_x_train, i_y_train))
    print()
    return lrm, dtc, rfc, knn, svm
```

15. Data Prediction and classification

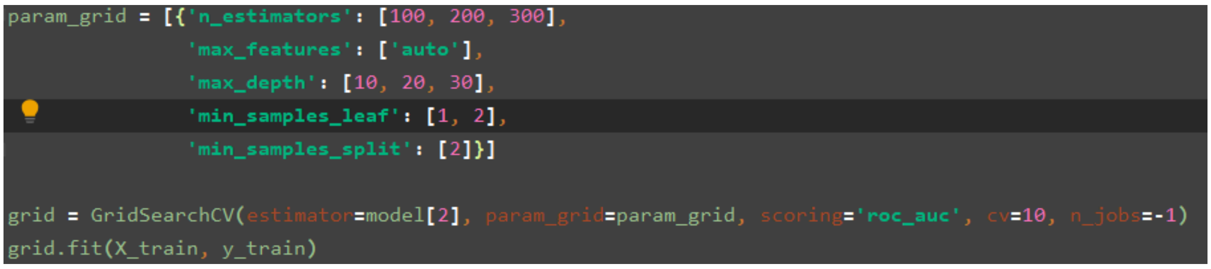
    Calculating the model accuracy for each of the models to see which one returns the highest score.
    Printing the highest score model to the least score.



```
                          Model      Accuracy Score
2       Random Forest Classifier  0.9790209790209791
4  Support Vector Classification   0.965034965034965
0            Logistic Regression   0.958041958041958
3            k-Nearest Neighbors   0.951048951048951
1       Decision Tree Classifier  0.9440559440559441
It is clear from our score that model Random Forest Classifier has the highest accuracy score of 0.9790209790209791 which is (97.90%)
```

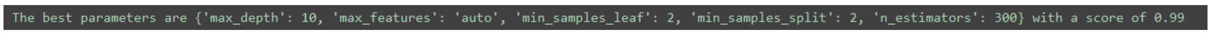
Random forest classifier has got a score of 97.9%

16. Next, I have used Hyper-Parameter Tuning model for Random Forest Classifier to see if we can increase its score



```
param_grid = [{'n_estimators': [100, 200, 300],
               'max_features': ['auto'],
               'max_depth': [10, 20, 30],
               'min_samples_leaf': [1, 2],
               'min_samples_split': [2]}]

grid = GridSearchCV(estimator=model[2], param_grid=param_grid, scoring='roc_auc', cv=10, n_jobs=-1)
grid.fit(X_train, y_train)
```

Output from this show accuracy score increased it to 0.99



```
The best parameters are {'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 300} with a score of 0.99
```

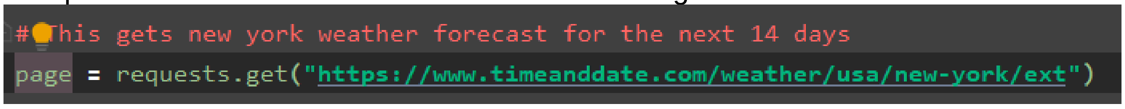
It is now evident from the accuracy score that Random Forest Classifier has got a highest score both before and after hyper tuning. I conclude that it is best suited model for this dataset.

Weather Forecast Implementation Process Results

This is to demonstrate web scraping, regular expression, and merge data frame features

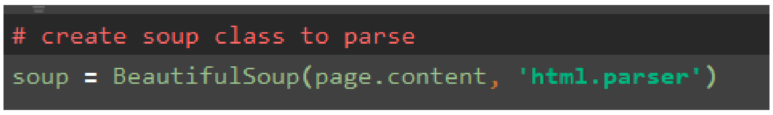
Steps

1. Scrape the NY forecast data from forecast.weather.gov



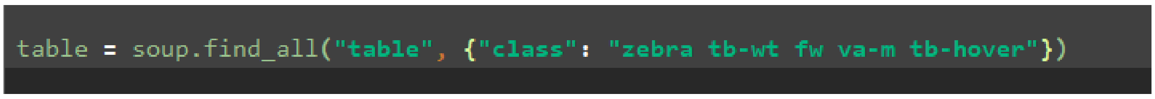
```
#This gets new york weather forecast for the next 14 days
page = requests.get("https://www.timeanddate.com/weather/usa/new-york/ext")
```

2. Created a soup class to parse html data using BeautifulSoup



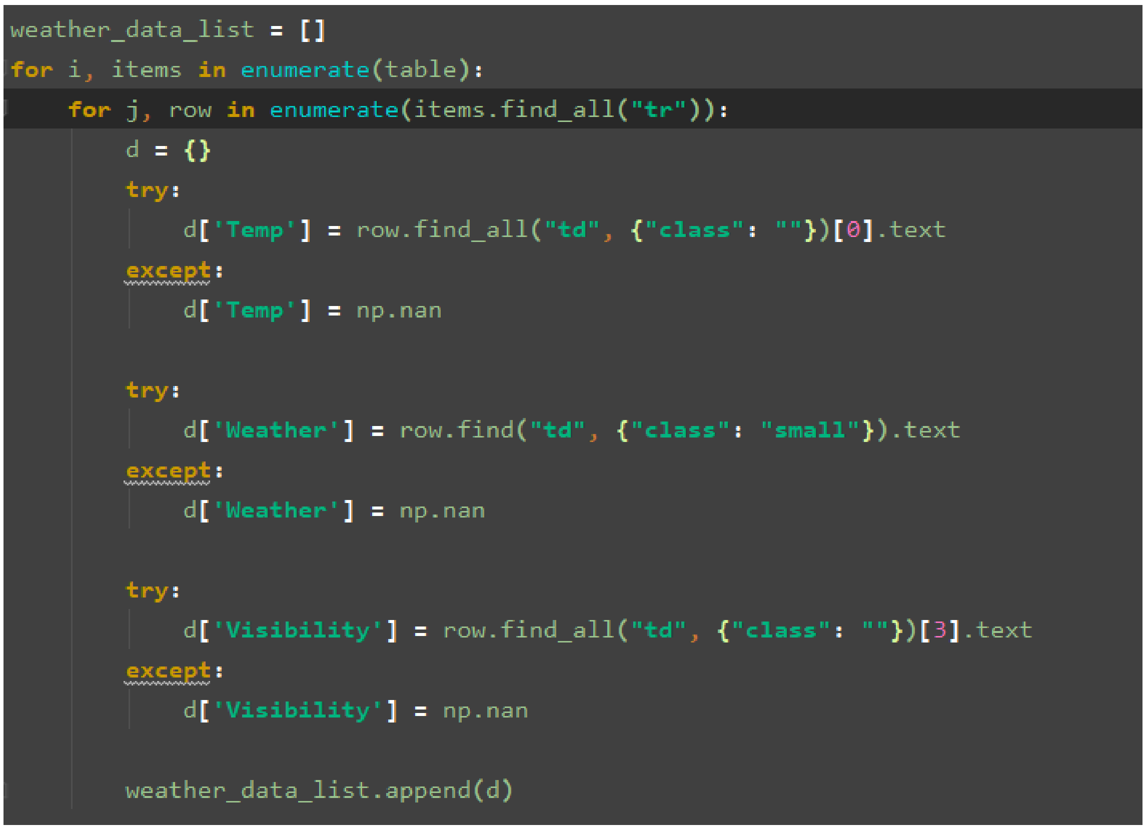
```
# create soup class to parse
soup = BeautifulSoup(page.content, 'html.parser')
```

3. Create a result set class called table of weather forecast



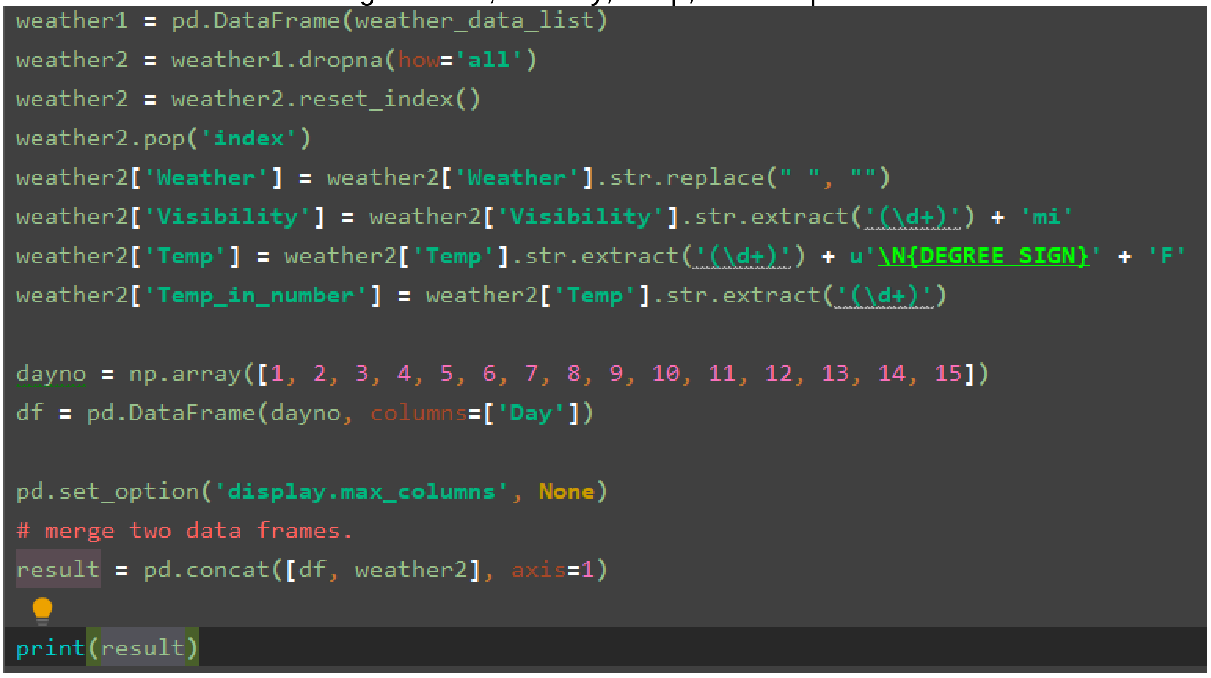
```
table = soup.find_all("table", {"class": "zebra tb-wt fw va-m tb-hover"})
```

4. Get all items from class ResultSet and append the data to weather data list



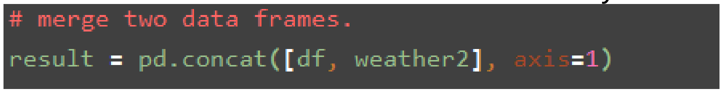
```python
weather_data_list = []
for i, items in enumerate(table):
    for j, row in enumerate(items.find_all("tr")):
        d = {}
        try:
            d['Temp'] = row.find_all("td", {"class": ""})[0].text
        except:
            d['Temp'] = np.nan

        try:
            d['Weather'] = row.find("td", {"class": "small"}).text
        except:
            d['Weather'] = np.nan

        try:
            d['Visibility'] = row.find_all("td", {"class": ""})[3].text
        except:
            d['Visibility'] = np.nan

        weather_data_list.append(d)
```

5. create data frame containing weather, visibility, temp, and temperature in number.

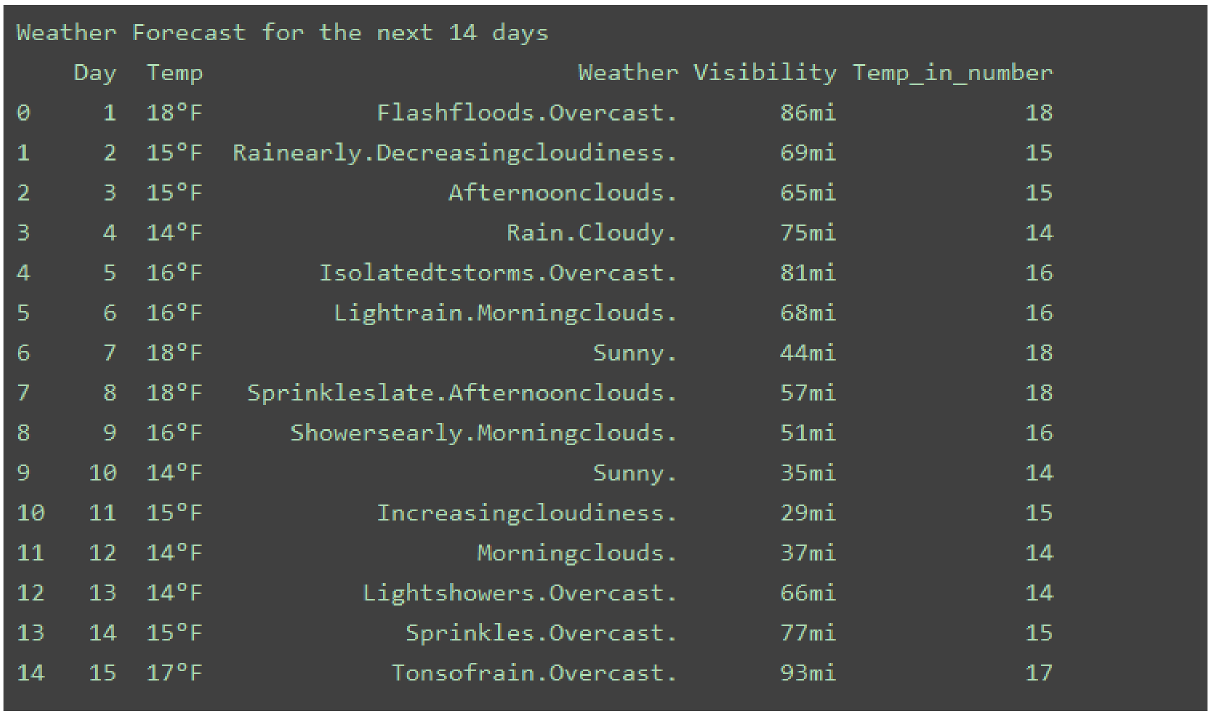


```python
weather1 = pd.DataFrame(weather_data_list)
weather2 = weather1.dropna(how='all')
weather2 = weather2.reset_index()
weather2.pop('index')
weather2['Weather'] = weather2['Weather'].str.replace(" ", "")
weather2['Visibility'] = weather2['Visibility'].str.extract('(\d+)') + 'mi'
weather2['Temp'] = weather2['Temp'].str.extract('(\d+)') + u'\N{DEGREE SIGN}' + 'F'
weather2['Temp_in_number'] = weather2['Temp'].str.extract('(\d+)')

dayno = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])
df = pd.DataFrame(dayno, columns=['Day'])

pd.set_option('display.max_columns', None)
# merge two data frames.
result = pd.concat([df, weather2], axis=1)

print(result)
```

6. merge data frames weather2 and df as weather using function pd.contact(). Display the overall weather forecast for the next 14 days.



```
# merge two data frames.
result = pd.concat([df, weather2], axis=1)
```

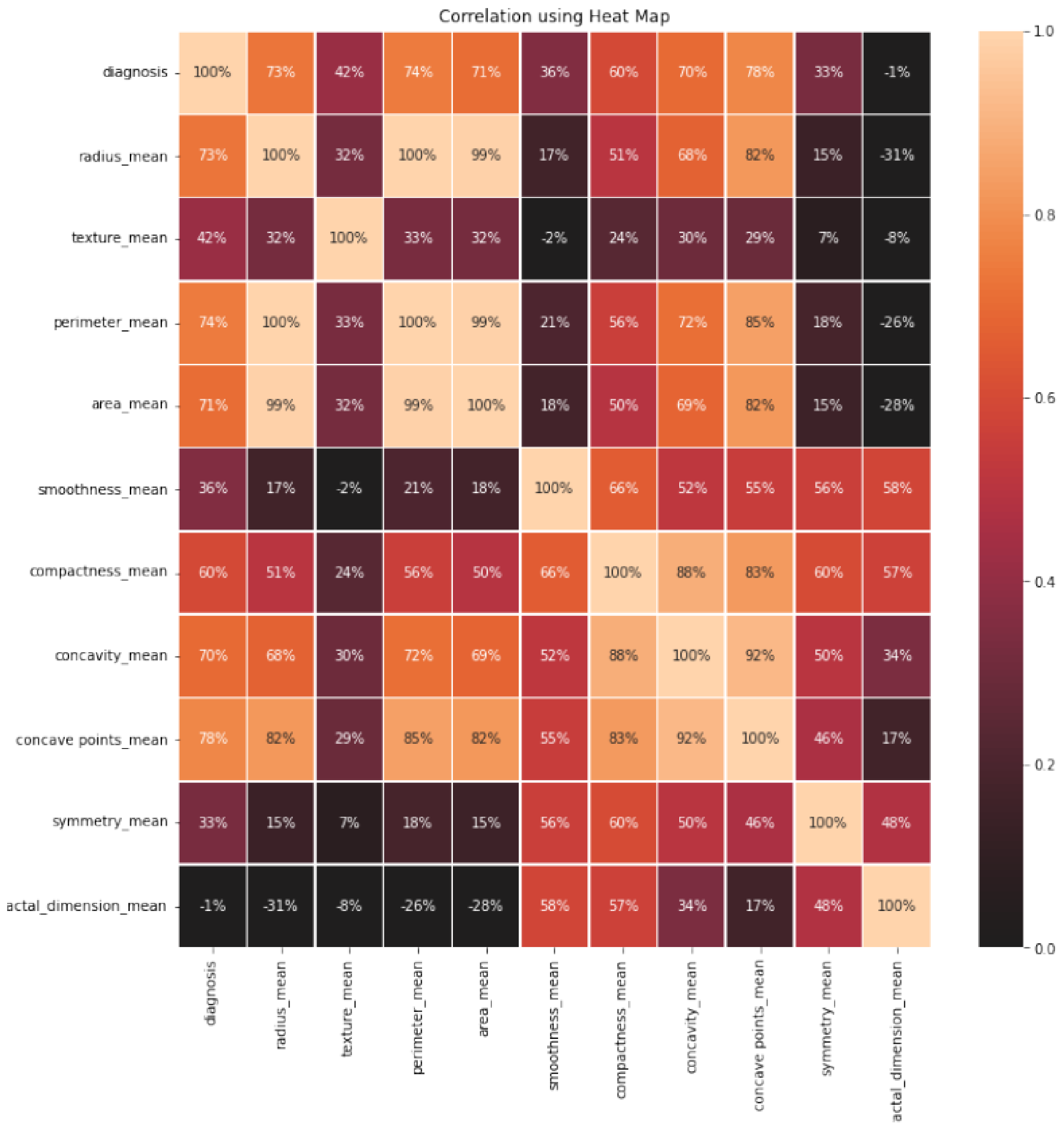


```
Weather Forecast for the next 14 days
    Day  Temp                            Weather Visibility Temp_in_number
0     1  18°F             Flashfloods.Overcast.       86mi             18
1     2  15°F  Rainearly.Decreasingcloudiness.       69mi             15
2     3  15°F                  Afternoonclouds.       65mi             15
3     4  14°F                      Rain.Cloudy.       75mi             14
4     5  16°F          Isolatedtstorms.Overcast.       81mi             16
5     6  16°F           Lightrain.Morningclouds.       68mi             16
6     7  18°F                            Sunny.       44mi             18
7     8  18°F    Sprinkleslate.Afternoonclouds.       57mi             18
8     9  16°F       Showersearly.Morningclouds.       51mi             16
9    10  14°F                            Sunny.       35mi             14
10   11  15°F             Increasingcloudiness.       29mi             15
11   12  14°F                    Morningclouds.       37mi             14
12   13  14°F            Lightshowers.Overcast.       66mi             14
13   14  15°F               Sprinkles.Overcast.       77mi             15
14   15  17°F              Tonsofrain.Overcast.       93mi             17
```

1.6 <u>Insights</u>

1. From heat map we can correlate few things

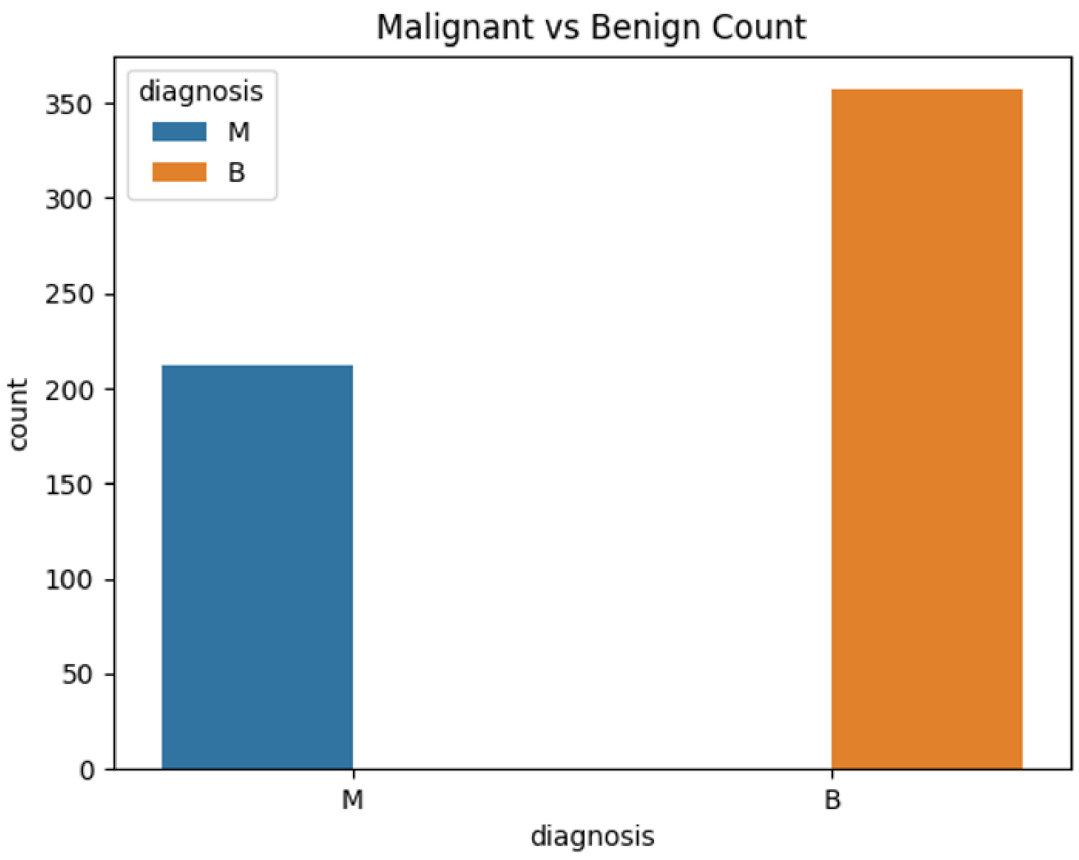
   ,



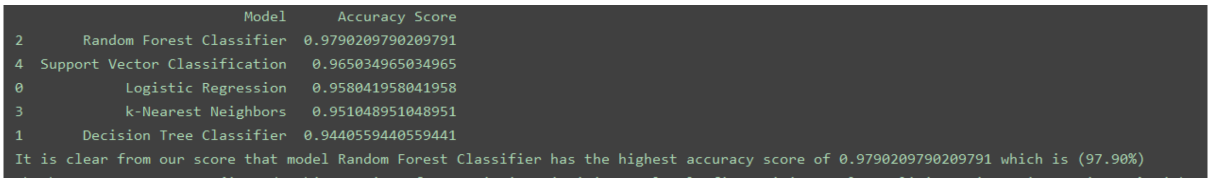
   The _mean which has over 0.7 are strongly correlated with feature diagnosis which are radius_mean, perimeter_mean, area_mean, concavity_mean, concave points_mean.

2. We can say features _mean which are less than 0.7 are weakly correlated with diagnosis.

3. From the count plot we can see there are more Benign than Malignant

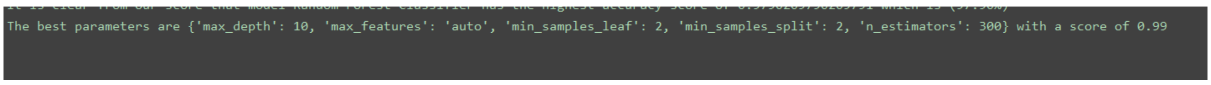


4. From data prediction and classification, we can confer random forest classifier model got the highest accuracy score before going for hyper tuning.



```
                             Model      Accuracy Score
2       Random Forest Classifier  0.9790209790209791
4  Support Vector Classification   0.965034965034965
0            Logistic Regression   0.958041958041958
3            k-Nearest Neighbors   0.951048951048951
1       Decision Tree Classifier  0.9440559440559441
It is clear from our score that model Random Forest Classifier has the highest accuracy score of 0.9790209790209791 which is (97.90%)
```

5. It is also confirmed after applying hyper tuning it has boosted score for random forest classifier model



```
The best parameters are {'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 300} with a score of 0.99
```

1.7 References

- Data Source for breast cancer-based Wisconsin:

  https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

- Data Source NY Weather Forecast Data: https://www.timeanddate.com/weather/usa/new-york/ext

- Plots: https://seaborn.pydata.org/

- Pandas: https://pandas.pydata.org/

- Breast Cancer Data Information Source: https://www.uicc.org/news/globocan-2020-new-global-cancer-data