THE REPORT

BREAST CANCER DETECTION AND PREDICTION USING MACHINE LEARNING

ABSTRACT

Machine learning is frequently used in medical applications such as detection of the type of cancerous cells. Breast cancer represents one of the diseases that causes a high number of deaths every year. It is the most common type of cancer and the main cause of women's deaths worldwide. The cancerous cells are classified as Benign (B) or Malignant (M). There are many algorithms for classification and prediction of breast cancer: Support Vector Machine (SVM), Decision Tree (CART), Naive Bayes (NB) and k Nearest Neighbours (kNN). In this project, Support Vector Machine (SVM) on the Wisconsin Breast Cancer dataset is used. The dataset is also trained with the other algorithms: KNN, Naives Bayes and CART and the accuracy of prediction for each algorithm is compared.

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 modeling.

DESCRIPTION OF THE DATASET

Data Set Characteristics:

1. Number of Instances: 569

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

Attribute Information:

3. radius (mean of distances from center to points on the perimeter)

4.texture (standard deviation of gray-scale values)

4. perimeter

5. area

6. smoothness (local variation in radius lengths)

7. compactness (perimeter^2 / area - 1.0)

8. concavity (severity of concave portions of the contour)

9. concave points (number of concave portions of the contour)

10. symmetry

11. 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.

PROPOSED SOLUTION

1. ID number 2) Diagnosis (M = malignant, B = benign) 3–32) [Ten real valued features are computed for each cell nucleus:]

2. radius (mean of distances from center to points on the perimeter)

3. texture (standard deviation of gray-scale values)

4. Perimeter

5. area

6. smoothness (local variation in radius lengths)

7. compactness (perimeter² / area — 1.0)

8. concavity (severity of concave portions of the contour)

9. concave points (number of concave portions of the contour)

10. Symmetry

11. 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

APPROACH

Our approach to the solution of this project is first we will be doing the visualization of the data evaluation matrices as well such as precision.

Understanding the given dataset and helps clean up the given dataset.It gives you a clear picture of the features and the relationships between them.Providing guidelines for essential variables and leaving behind/removing non-essential variables.Handling Missing values or human error.

Identifying outliers.EDA process would be maximizing insights of a dataset.

VISUALIZATION

IMPORTING LIBRARIES

```
[ ]  #import libraries
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
```

LOAD THE DATASET

```
[ ]  #Load the data
     from google.colab import files # Use to load data on Google Colab
     uploaded = files.upload() # Use to load data on Google Colab
     df = pd.read_csv('data.csv')
     df.head(7) #showing forst 7 data rows
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving data.csv to data.csv

	id	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	text
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	1.0950	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	0.5435	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	0.7456	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	0.4956	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	0.7572	
5	843786	M	12.45	15.70	82.57	477.1	0.12780	0.17000	0.1578	0.08089	0.2087	0.07613	0.3345	
6	844359	M	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.1127	0.07400	0.1794	0.05742	0.4467	

Get a count of all of the columns that contain empty (NaN, NAN, na) values

```
[ ]  #Count the empty (NaN, NAN, na) values in each column
     df.isna().sum()
```

```
id                           0
diagnosis                    0
radius_mean                  0
texture_mean                 0
perimeter_mean               0
area_mean                    0
smoothness_mean              0
compactness_mean             0
concavity_mean               0
concave points_mean          0
symmetry_mean                0
fractal_dimension_mean       0
radius_se                    0
texture_se                   0
perimeter_se                 0
area_se                      0
smoothness_se                0
compactness_se               0
concavity_se                 0
concave points_se            0
symmetry_se                  0
fractal_dimension_se         0
radius_worst                 0
texture_worst                0
perimeter_worst              0
area_worst                   0
smoothness_worst             0
compactness_worst            0
concavity_worst              0
concave points_worst         0
symmetry_worst               0
fractal_dimension_worst      0
Unnamed: 32                569
dtype: int64
```

Remove the column 'Unnamed: 32

```
[ ]  #Drop the column with all missing values (na, NAN, NaN)
     #NOTE: This drops the column Unnamed
     df = df.dropna(axis=1)
```

Get a count of the number of patients with Malignant (M) cancerous and Benign (B) non-cancerous cells

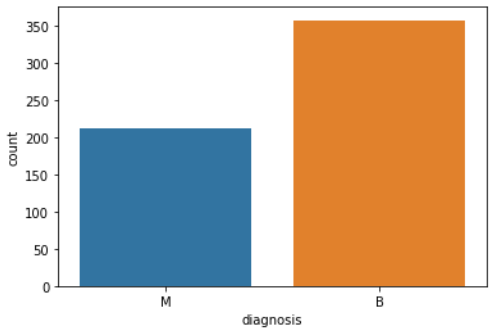
```
[ ]  #Get a count of the number of 'M' & 'B' cells
     df['diagnosis'].value_counts()
```

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

Visualize the counts

```
[ ]  #Visualize this count
     sns.countplot(df['diagnosis'],label="Count")
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fd8cabef690>
```



Encode the categorical data. Change the values in the column ‘diagnosis’ from M and B to 1 and 0 respectively, then print the results

```
[ ]  #Encoding categorical data values (
     from sklearn.preprocessing import LabelEncoder
     labelencoder_Y = LabelEncoder()
     df.iloc[:,1]= labelencoder_Y.fit_transform(df.iloc[:,1].values)
     print(labelencoder_Y.fit_transform(df.iloc[:,1].values))#Encoding categorical data values (
     from sklearn.preprocessing import LabelEncoder
     labelencoder_Y = LabelEncoder()
     df.iloc[:,1]= labelencoder_Y.fit_transform(df.iloc[:,1].values)
     print(labelencoder_Y.fit_transform(df.iloc[:,1].values))
```

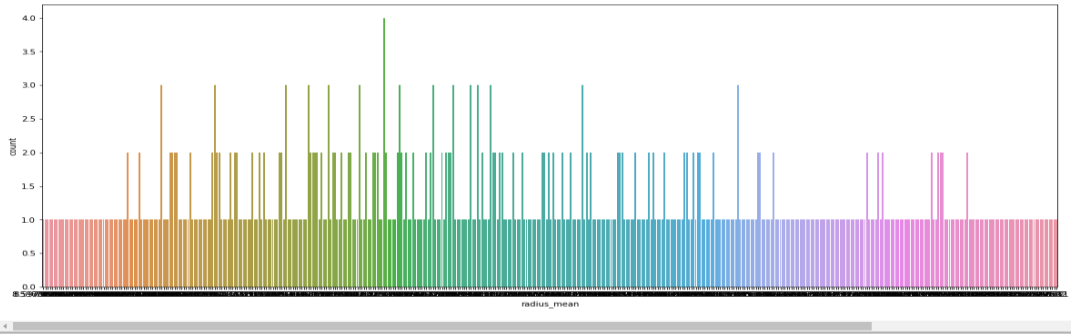
```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 1
 0 1 0 1 1 0 0 0 1 1 0 1 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 0
 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 0 1 0
 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 1
 0 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0 1 0 1 0 0 1 0 1 1 1 1 0 0 1 1 0 0
 0 1 0 0 0 0 0 1 1 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0
 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1
 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0
 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 0
 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 1
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 1
 0 1 0 1 1 0 0 0 1 1 0 1 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 0
 0 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 0 1 0
 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 1
 0 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0 1 0 1 0 0 1 0 1 1 1 1 0 0 1 1 0 0
 0 1 0 0 0 0 0 1 1 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0
 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1
 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0
 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 0
 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 1
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0]
```

Create a pair plot

counter plot of feature mean radius

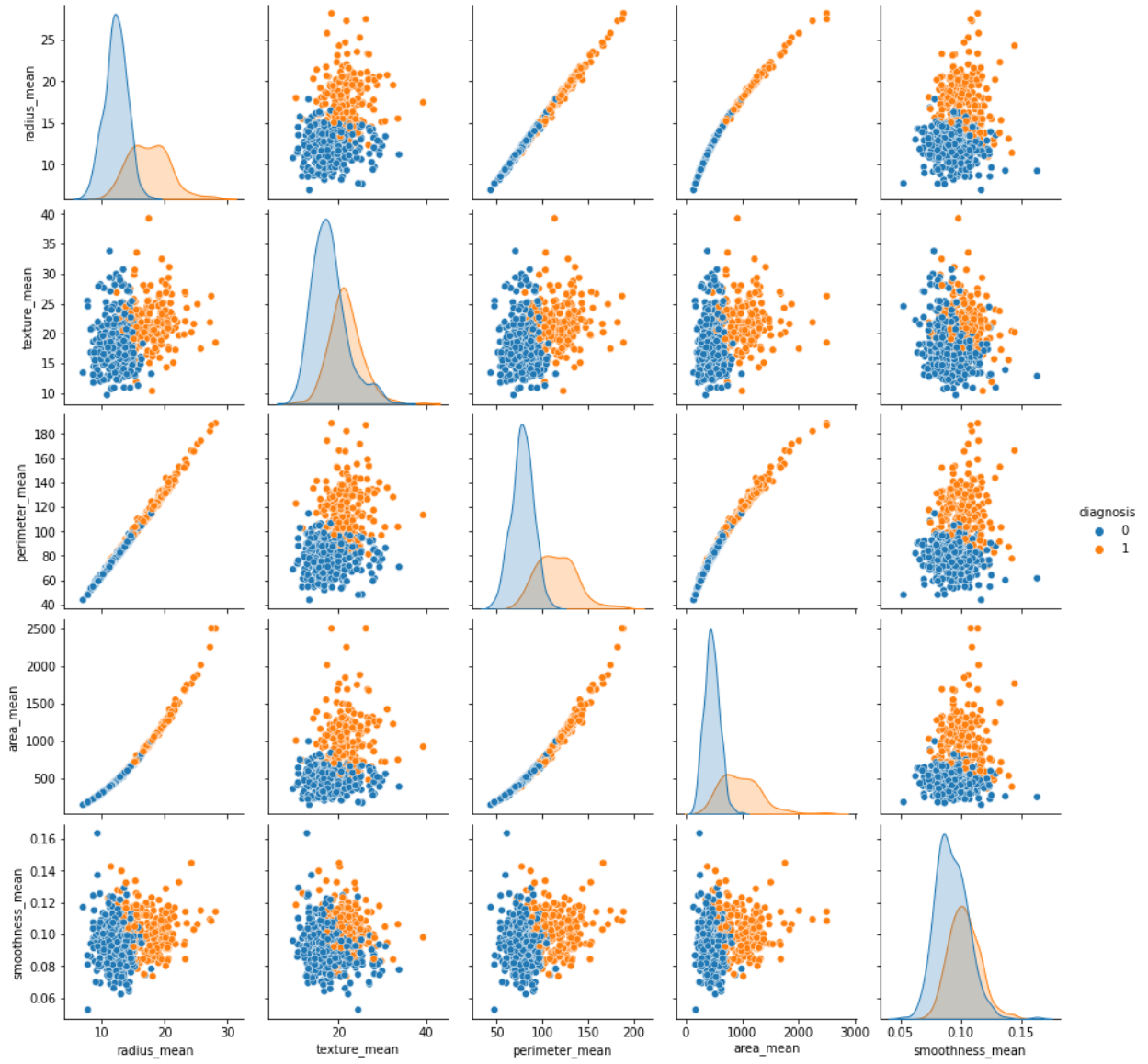
```
# counter plot of feature mean radius
plt.figure(figsize = (20,8))
sns.countplot(df['radius_mean'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fd8b99dc490>
```



Pair plot of sample feature

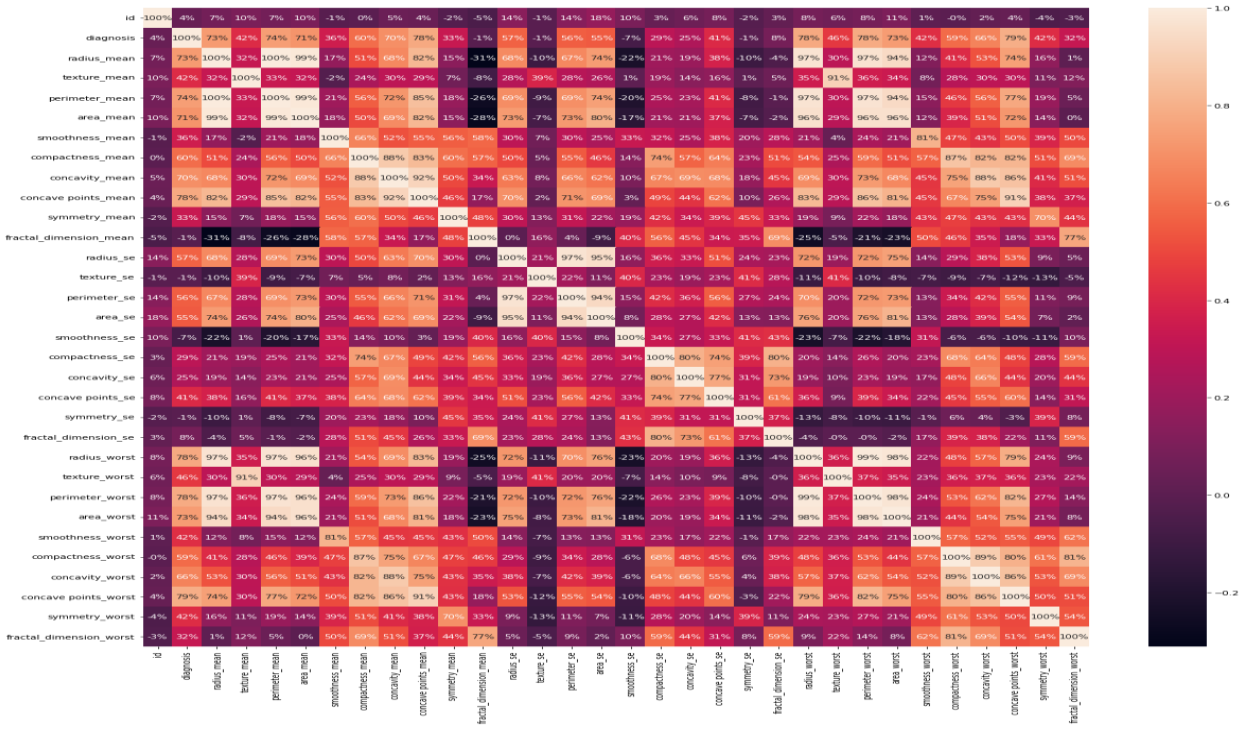
```
# pair plot of sample feature
sns.pairplot(df, hue = 'diagnosis',
             vars = ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness_mean'] )
```



Get the correlation of the columns

	id	diagnosis	radius_mean	texture_mean	perimeter_mean
id	1.000000	0.039769	0.074626	0.099770	0.073159
diagnosis	0.039769	1.000000	0.730029	0.415185	0.742636
radius_mean	0.074626	0.730029	1.000000	0.323782	0.997855
texture_mean	0.099770	0.415185	0.323782	1.000000	0.329533
perimeter_mean	0.073159	0.742636	0.997855	0.329533	1.000000
area_mean	0.096893	0.708984	0.987357	0.321086	0.986507
smoothness_mean	-0.012968	0.358560	0.170581	-0.023389	0.207278
compactness_mean	0.000096	0.596534	0.506124	0.236702	0.556936
concavity_mean	0.050080	0.696360	0.676764	0.302418	0.716136
concave points_mean	0.044158	0.776614	0.822529	0.293464	0.850977
symmetry_mean	-0.022114	0.330499	0.147741	0.071401	0.183027
fractal_dimension_mean	-0.052511	-0.012838	-0.311631	-0.076437	-0.261477
radius_se	0.143048	0.567134	0.679090	0.275869	0.691765
texture_se	-0.007526	-0.008303	-0.097317	0.386358	-0.086761
perimeter_se	0.137331	0.556141	0.674172	0.281673	0.693135
area_se	0.177742	0.548236	0.735864	0.259845	0.744983
smoothness_se	0.096781	-0.067016	-0.222600	0.006614	-0.202694

Visualize the correlation by creating a heat map



Splitting the data set

```
[ ]  X = df.iloc[:, 2:31].values
     Y = df.iloc[:, 1].values
```

Split the data again, but this time into 75% training and 25% testing data sets

```
[ ]  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)
```

Scale the data to bring all features to the same level of magnitude, which means the feature / independent data will be within a specific range for example 0–100 or 0–1

```
[ ]  #Feature Scaling
     from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     X_train = sc.fit_transform(X_train)
     X_test = sc.transform(X_test)
```

Create a function to hold many different models (e.g. Logistic Regression, Decision Tree Classifier, Random Forest Classifier) to make the classification. These are the models that will detect if a patient has cancer or not. Within this function I will also print the accuracy of each model on the training data

```
def models(X_train,Y_train):

    #Using Logistic Regression
    from sklearn.linear_model import LogisticRegression
    log = LogisticRegression(random_state = 0)
    log.fit(X_train, Y_train)

    #Using KNeighborsClassifier
    from sklearn.neighbors import KNeighborsClassifier
    knn = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
    knn.fit(X_train, Y_train)

    #Using SVC linear
    from sklearn.svm import SVC
    svc_lin = SVC(kernel = 'linear', random_state = 0)
    svc_lin.fit(X_train, Y_train)

    #Using SVC rbf
    from sklearn.svm import SVC
    svc_rbf = SVC(kernel = 'rbf', random_state = 0)
    svc_rbf.fit(X_train, Y_train)

    #Using GaussianNB
    from sklearn.naive_bayes import GaussianNB
    gauss = GaussianNB()
    gauss.fit(X_train, Y_train)

    #Using DecisionTreeClassifier
    from sklearn.tree import DecisionTreeClassifier
    tree = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
    tree.fit(X_train, Y_train)

    #Using RandomForestClassifier method of ensemble class to use Random Forest Classificati
    from sklearn.ensemble import RandomForestClassifier
    forest = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state =
    forest.fit(X_train, Y_train)

    #print model accuracy on the training data.
    print('[0]Logistic Regression Training Accuracy:', log.score(X_train, Y_train))
    print('[1]K Nearest Neighbor Training Accuracy:', knn.score(X_train, Y_train))
    print('[2]Support Vector Machine (Linear Classifier) Training Accuracy:', svc_lin.score(
    print('[3]Support Vector Machine (RBF Classifier) Training Accuracy:', svc_rbf.score(X_t
    print('[4]Gaussian Naive Bayes Training Accuracy:', gauss.score(X_train, Y_train))
    print('[5]Decision Tree Classifier Training Accuracy:', tree.score(X_train, Y_train))
    print('[6]Random Forest Classifier Training Accuracy:', forest.score(X_train, Y_train))

    return log, knn, svc_lin, svc_rbf, gauss, tree, forest
```

Create the model that contains all of the models, and look at the accuracy score on the training data for each model to classify if a patient has cancer or not

```
[ ]  model = models(X_train,Y_train)
```

```
[0]Logistic Regression Training Accuracy: 0.9906103286384976
[1]K Nearest Neighbor Training Accuracy: 0.9765258215962441
[2]Support Vector Machine (Linear Classifier) Training Accuracy: 0.9882629107981221
[3]Support Vector Machine (RBF Classifier) Training Accuracy: 0.9835680751173709
[4]Gaussian Naive Bayes Training Accuracy: 0.9507042253521126
[5]Decision Tree Classifier Training Accuracy: 1.0
[6]Random Forest Classifier Training Accuracy: 0.9953051643192489
```

Show the confusion matrix and the accuracy of the models on the test data. The confusion matrix tells us how many patients each model misdiagnosed (number of patients with cancer that were misdiagnosed as not having cancer a.k.a false negative, and the number of patients who did not have cancer that were misdiagnosed with having cancer a.k.a false positive) and the number of correct diagnosis, the true positives and true negatives

False Positive (FP) = A test result which incorrectly indicates that a particular condition or attribute is present

True Positive (TP) = Sensitivity (also called the true positive rate, or probability of detection in some fields) measures the proportion of actual positives that are correctly identified as such

True Negative (TN) = Specificity (also called the true negative rate) measures the proportion of actual negatives that are correctly identified as such

False Negative (FN) = A test result that indicates that a condition does not hold, while in fact it does. For example a test result that indicates a person does not have cancer when the person actually does have it

```
[ ]  from sklearn.metrics import confusion_matrix
     for i in range(len(model)):
       cm = confusion_matrix(Y_test, model[i].predict(X_test))

       TN = cm[0][0]
       TP = cm[1][1]
       FN = cm[1][0]
       FP = cm[0][1]

       print(cm)
       print('Model[{}] Testing Accuracy = "{}!"'.format(i,  (TP + TN) / (TP + TN + FN + FP)))
       print()# Print a new line
```

```
[[86  4]
 [ 4 49]]
Model[0] Testing Accuracy = "0.9440559440559441!"

[[89  1]
 [ 5 48]]
Model[1] Testing Accuracy = "0.958041958041958!"

[[87  3]
 [ 2 51]]
Model[2] Testing Accuracy = "0.965034965034965!"

[[88  2]
 [ 3 50]]
Model[3] Testing Accuracy = "0.965034965034965!"

[[85  5]
 [ 6 47]]
Model[4] Testing Accuracy = "0.9230769230769231!"

[[84  6]
 [ 1 52]]
Model[5] Testing Accuracy = "0.951048951048951!"

[[87  3]
 [ 2 51]]
Model[6] Testing Accuracy = "0.965034965034965!"
```

Other ways to show accuracy

```
[ ]  #Show other ways to get the classification accuracy & other metrics

     from sklearn.metrics import classification_report
     from sklearn.metrics import accuracy_score

     for i in range(len(model)):
       print('Model ',i)
       #Check precision, recall, f1-score
       print( classification_report(Y_test, model[i].predict(X_test)) )
       #Another way to get the models accuracy on the test data
       print( accuracy_score(Y_test, model[i].predict(X_test)))
       print()#Print a new line
```

```
Model  0
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        90
           1       0.92      0.92      0.92        53

    accuracy                           0.94       143
   macro avg       0.94      0.94      0.94       143
weighted avg       0.94      0.94      0.94       143

0.9440559440559441

Model  1
              precision    recall  f1-score   support

           0       0.95      0.99      0.97        90
           1       0.98      0.91      0.94        53

    accuracy                           0.96       143
   macro avg       0.96      0.95      0.95       143
weighted avg       0.96      0.96      0.96       143

0.958041958041958
```

```
Model  2
              precision    recall  f1-score   support

           0       0.98      0.97      0.97        90
           1       0.94      0.96      0.95        53

    accuracy                           0.97       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.97      0.97      0.97       143

0.965034965034965

Model  3
              precision    recall  f1-score   support

           0       0.97      0.98      0.97        90
           1       0.96      0.94      0.95        53

    accuracy                           0.97       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.96      0.97      0.96       143

0.965034965034965

Model  4
              precision    recall  f1-score   support

           0       0.93      0.94      0.94        90
           1       0.90      0.89      0.90        53

    accuracy                           0.92       143
   macro avg       0.92      0.92      0.92       143
weighted avg       0.92      0.92      0.92       143

0.9230769230769231

Model  5
              precision    recall  f1-score   support

           0       0.99      0.93      0.96        90
           1       0.90      0.98      0.94        53

    accuracy                           0.95       143
   macro avg       0.94      0.96      0.95       143
weighted avg       0.95      0.95      0.95       143

0.951048951048951
```

Prediction of Random Forest Classifier model

```
#Print Prediction of Random Forest Classifier model
pred = model[6].predict(X_test)
print(pred)

#Print a space
print()

#Print the actual values
print(Y_test)
```

```
[1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 0 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 1 0 0 1 0 0
 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0
 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1]

[1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 1 0 0 1 0 1
 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0
 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1]
```

CONCLUSION & FUTURE SCOPE

In this project in python, we learned to build a breast cancer tumor predictor on the Wisconsin dataset and created graphs and results for the same. It has been observed that a good dataset provides better accuracy. Selection of appropriate algorithms with a good home dataset will lead to the development of prediction systems. These systems can assist in proper treatment methods for a patient diagnosed with breast cancer. There are many treatments for a patient based on breast cancer stage; data mining and machine learning can be a very good help in deciding the line of treatment to be followed by extracting knowledge from such suitable databases.

The references

Dataset:https://docs.google.com/spreadsheets/d/1Lb-fXd0VUMekf_w-mW6bFlbIB8Q_qqxdGsFTw9gYQ0M/edit?usp=sharing

This project is done by:

1. Zameeel Ali Mohammad

2.Paras Kushuwa

3.Sridhar Mallarapu

4.Sulagna Maji