Breast Cancer Data Analysis with Python

Prepared by Antika Das[redacted] MSc Computer Science for Autonomous System
Eötvös Loránd University, H-1053 Budapest, Egyetem tér 1-3

Email-id:antika.das95@gmail.com

Abstract:

This is an analysis of the *Breast Cancer Wisconsin (Diagnostic) DataSet,* obtained from **https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29.** The dataset used is publicly available and was created by Dr. William H. Wolberg, physician at the University of Wisconsin Hospital at Madison, Wisconsin, USA. After exploring and processing the raw data, the classification task was executed using several ML models like SVC, Logistic Regression, Random Forest, Decision Tree, KNN algorithm etc. This report presents better options for the choice of classifier.

Key words: *Breast Cancer,* Malignant, Benign, Data Mining, Classification algorithms, Cross Validation, Accuracy.

Introduction:

Breast cancer is the most common malignancy among women, accounting for nearly 1 in 3 cancers diagnosed among women in the United States and all over the worlds, and it is the second leading cause of cancer death among women. Breast Cancer occurs as a results of abnormal growth of cells in the breast tissue, commonly referred to as a Tumor. A tumor does not mean cancer - tumors can be benign (not cancerous), pre-malignant (pre-cancerous), or malignant (cancerous). Tests such as MRI, mammogram, ultrasound and biopsy are commonly used to diagnose breast cancer performed .The early diagnosis of breast cancer 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 breast cancer and classification of patients into malignant or benign groups is the subject of much research. 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.

- **Some Risk Factors for Breast Cancer**

  Age. The chance of getting breast cancer increases as women age. **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.

Genetic 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.

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 (after 12)
- Women who go through menopause late (after age 55)
- Women who've never had children

Data Preparation:

The dataset has 11 variables with 699 observations, first variable is the identifier and has been excluded in the analysis. Thus, there are **9 predictors** and **a response** variable (class). The response variable denotes "Malignant" or "Benign" cases.

Predictor variables:

- Clump Thickness
- Uniformity of Cell Size
- Single Epithelial Cell Size
- Bare Nuclei
- Uniformity of Cell Shape
- Bland Chromatin
- Mitoses
- Marginal Adhesion

Response variable:

- Class - (2 for benign, 4 for malignant)

** **There are 16 observations where data is incomplete. In further analysis, these cases are replaced by the most frequent value of the attribute. In total, there are 241 cases of malignancy (4), whereas benign cases are 458(2).**

Data loading (.csv):

In [2]:
```
#PRE DATA PROCESSING

#read data
columns = [ "id_number","Clump_Thickness","Uniformity_of_Cell_Size","Uniformity_of_Cell_Shape","Marginal_Adhesion",
"Single_Epithelial_Cell_Size","Bare_Nuclei","Bland_Chromatin","Normal_Nucleoli","Mitose","Class"]

df = pd.read_csv('C:\\Users\\ASUS\\Downloads\\breast-cancer-wisconsin.data', sep=",",names=columns)
#df.to_csv (r'C:\\Users\\ASUS\\Downloads\\Breast-Cancer-Wisconsin.csv', index=None

df.head()

```

Out[2]:

	id_number	Clump_Thickness	Uniformity_of_Cell_Size	Uniformity_of_Cell_Shape	Marginal_Adhesion	Single_Epithelial_Cell_Size	Bare_Nuclei	Bland_Chromati
0	1000025	5	1	1	1	2	1	
1	1002945	5	4	4	5	7	10	
2	1015425	3	1	1	1	2	2	
3	1016277	6	8	8	1	3	4	
4	1017023	4	1	1	3	2	1	

Figure 1.

Drop the "id number" as mentioned before, then Replace the "?" values with the most frequent value(1) then print the data information and count the total number of malignancy and benign :

In [222]:
```
df['Bare_Nuclei'].value_counts()
```

Out[222]:
```
1     402
10    132
5      30
2      30
3      28
8      21
4      19
?      16
9       9
7       8
6       4
Name: Bare_Nuclei, dtype: int64
```

Figure 2.

In [4]:

```
df.replace('?',1,inplace=True)
#drop the id column as it does not effect the output
df.drop(['id_number'],1,inplace=True)
#change the data type
df['Bare_Nuclei'] = df['Bare_Nuclei'].astype(np.int64)
df.info()

```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699 entries, 0 to 698
Data columns (total 10 columns):
Clump_Thickness                699 non-null int64
Uniformity_of_Cell_Size        699 non-null int64
Uniformity_of_Cell_Shape       699 non-null int64
Marginal_Adhesion              699 non-null int64
Single_Epithelial_Cell_Size    699 non-null int64
Bare_Nuclei                    699 non-null int64
Bland_Chromatin                699 non-null int64
Normal_Nucleoli                699 non-null int64
Mitose                         699 non-null int64
Class                          699 non-null int64
dtypes: int64(10)
memory usage: 54.7 KB
```

In [26]:

```
df['Class'].value_counts()
```

Out[26]:

```
2    458
4    241
Name: Class, dtype: int64
```

Figure 3.

Check if there is any null value:

In [7]:

```
df.shape

```

Out[7]: (699, 10)

In [8]:

```
df.isnull()
```

Out[8]:

	Clump_Thickness	Uniformity_of_Cell_Size	Uniformity_of_Cell_Shape	Marginal_Adhesion	Single_Epithelial_Cell_Size	Bare_Nuclei	Bland_Chromatin	Normal
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
5	False	False	False	False	False	False	False	
6	False	False	False	False	False	False	False	
7	False	False	False	False	False	False	False	
8	False	False	False	False	False	False	False	
9	False	False	False	False	False	False	False	
10	False	False	False	False	False	False	False	
11	False	False	False	False	False	False	False	
12	False	False	False	False	False	False	False	
13	False	False	False	False	False	False	False	

Figure 4.

Correlation matrix of the data-frame:

```
In [5]:
corr = df.corr()
corr.style.background_gradient(cmap='coolwarm')
```

Out[5]:

	Clump_Thickness	Uniformity_of_Cell_Size	Uniformity_of_Cell_Shape	Marginal_Adhesion	Single_Epithelial_Cell_Size	Bare_Nuclei	Bla
Clump_Thickness	1	0.644913	0.654589	0.486356	0.521816	0.590008	
Uniformity_of_Cell_Size	0.644913	1	0.906882	0.705582	0.751799	0.686673	
Uniformity_of_Cell_Shape	0.654589	0.906882	1	0.683079	0.719668	0.707474	
Marginal_Adhesion	0.486356	0.705582	0.683079	1	0.599599	0.666971	
Single_Epithelial_Cell_Size	0.521816	0.751799	0.719668	0.599599	1	0.583701	
Bare_Nuclei	0.590008	0.686673	0.707474	0.666971	0.583701	1	
Bland_Chromatin	0.558428	0.755721	0.735948	0.666715	0.616102	0.674215	
Normal_Nucleoli	0.535835	0.722865	0.719446	0.603352	0.628881	0.574778	
Mitose	0.350034	0.458693	0.438911	0.417633	0.479101	0.342397	
Class	0.716001	0.817904	0.818934	0.6968	0.682785	0.818968	

Figure 5.

The malignant or benign tumors cells can have (or not) different values for the features. plotting the distribution of each type of diagnosis for each of the mean features-BOXPLOT :

```
for i, feature in enumerate(features_mean):
    rows = int(len(features_mean)/2)

    plt.subplot(rows, 2, i+1)

    sns.boxplot(x='Class', y=feature, data=df, palette="Set1")

plt.tight_layout()
plt.show()
```

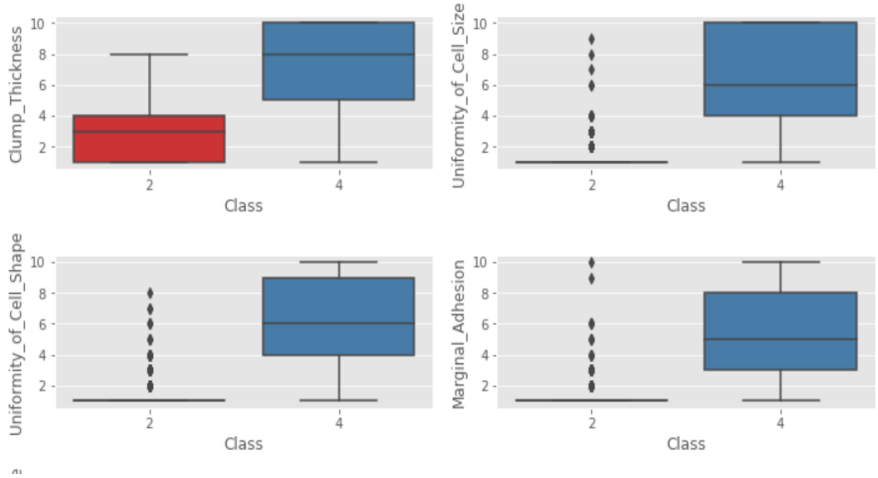


Figure 6.

Define input and output variables:

```
In [10]:
X=np.array(df.drop(['Class'],1)) #independent variables (input)
y=np.array(df['Class']) #output variables

In [11]:
X,y

Out[11]:
(array([[ 5,  1,  1, ...,  3,  1,  1],
        [ 5,  4,  4, ...,  3,  2,  1],
        [ 3,  1,  1, ...,  3,  1,  1],
        ...,
        [ 5, 10, 10, ...,  8, 10,  2],
        [ 4,  8,  6, ..., 10,  6,  1],
        [ 4,  8,  8, ..., 10,  4,  1]], dtype=int64),
 array([2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 4, 2, 4, 4, 2, 2, 4, 2, 4, 4,
        2, 4, 2, 4, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 4, 2, 4, 4, 2, 4, 4, 4,
        4, 2, 4, 2, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 2, 4, 4, 2, 4,
        2, 4, 4, 2, 2, 4, 2, 4, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 2, 4, 4, 4, 4, 4, 2, 4,
        2, 4, 4, 4, 2, 2, 2, 4, 2, 2, 2, 2, 4, 4, 4, 2, 4, 2, 4, 2, 2,
        4, 2, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 4, 2, 2, 4, 2, 4, 4, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        4, 2, 2, 4, 2, 2, 4, 2, 2, 2, 4, 4, 2, 2, 2, 4, 2, 2, 2, 4, 4, 2, 2,
        2, 4, 2, 2, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 4, 4, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 4, 2, 2, 2, 2, 4, 2, 2, 2, 2,
```

Figure 7.

Split the data into test and train dataset:

```
In [12]:
# Splitting the dataset into the Training set and Test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 42)
```

Figure 8.

Perform feature Scaling:

```
In [13]:
# Feature Scaling

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

Figure 9.

Model fitting/building And Evaluation:

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. K-nearest neighbors (KNN) algorithm uses 'feature similarity' to predict the values of new data points which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. It is a versatile algorithm as we can use it for classification as well as regression. Here we have used **KNN Classification** algorithm.

***Here KNN algorithm has been implemented by two techniques**

1. Using Cross validation to directly find out the BEST PARAMETERS & BEST ESTIMATOR.

```
#FIND NUMBER OF BEST PARAMETERS & BEST ESTIMATOR USING CROSS VALIDATION FOR KNN
max_class=20
grid_param = {'n_neighbors': range(1, max_class)}
model = KNeighborsClassifier()
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=2)
clf = GridSearchCV(model, grid_param, cv=cv, scoring='accuracy')
clf.fit(X, y)
print("Best Estimator: \n{}\n".format(clf.best_estimator_))
print("Best Parameters : \n{}\n".format(clf.best_params_))
print("Best Score: \n{}\n".format(clf.best_score_))
Accuracy_CV.append(clf.best_score_)
```

```
Best Estimator:
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=9, p=2,
           weights='uniform')

Best Parameters :
{'n_neighbors': 9}

Best Score:
0.9706723891273248
```

Figure 10.

2. Elbow Method:

```
In [237]:
# FINDINGING THE BEST FIT K FOR KNN WITHOUT CROSS_VALIDATION
knn = []
KNN=[]
for i in range(1,21):
    classifier = KNeighborsClassifier(n_neighbors=i)
    trained_model=classifier.fit(X_train,y_train)
    trained_model.fit(X_train,y_train )

    # Predicting the Test set results
    y_pred = classifier.predict(X_test)

    # Making the Confusion Matrix
    cm_KNN = confusion_matrix(y_test, y_pred)

    KNN.append(accuracy_score(y_test, y_pred))

    knn.append([accuracy_score(y_test, y_pred),i])

accuracy=max(KNN)
for l in knn:
    if l[0]==accuracy:
        k=l[1]
        break
Accuracy_WO_CV.append(accuracy)
model_names.append("KNN")

print ("best of K using elbow_method:",k,",accuracy",accuracy)
print("accuracy score for tranning data:",accuracy_score(y_train, classifier.predict(X_train)))
trainning_accuracy.append(accuracy_score(y_train, classifier.predict(X_train)))
print ("classification report :",classification_report(y_test,y_pred))
```

Figure 11.

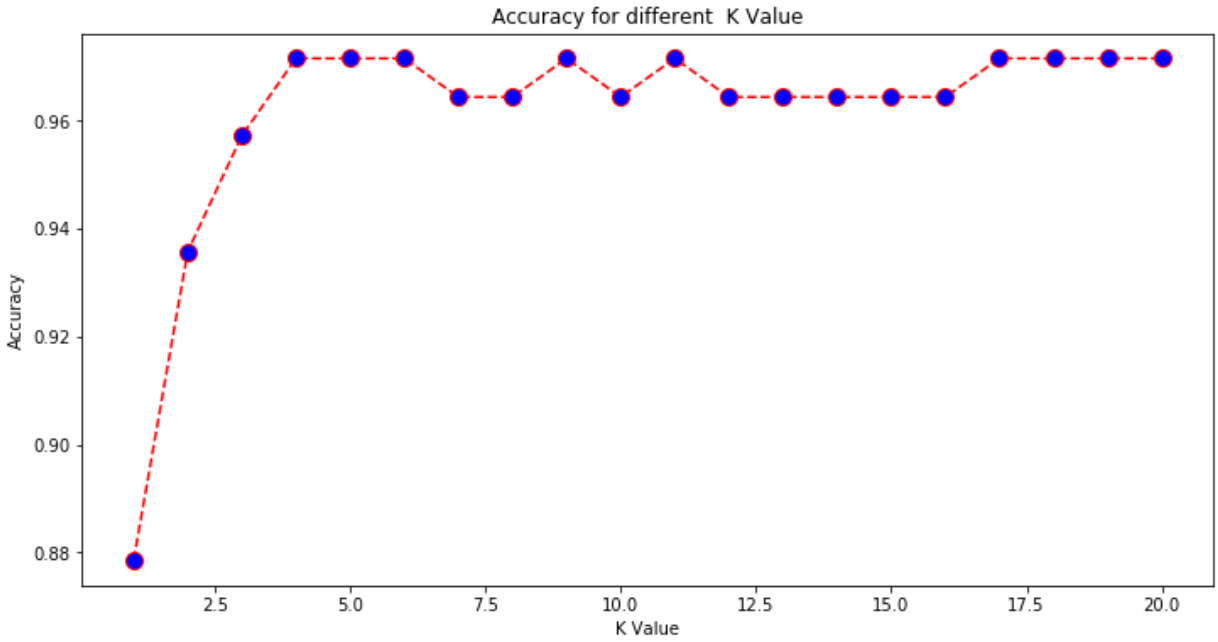


Figure 12.

SVC algorithm:

Support Vector Machine, abbreviated as SVM can be used for both regression and classification tasks. But, it is widely used in classification objectives (SVC). The objective of the support vector machine algorithm is to find a hyper plane in an N-dimensional space (N — the number of features) that distinctly classifies the data points:

In [238]:

```
#Fitting SVC to the Training set WITHOUT CROSS VALIDATION
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train,y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
#CONFUSION MATRIX
cm = confusion_matrix(y_test, y_pred)

#ACCURACY WITHOUT CROSS_VALIDATION
accuracy = accuracy_score(y_test,y_pred)
print ("ACCURACY WITHOUT CROSS_VALIDATION :",accuracy)
Accuracy_WO_CV.append(accuracy)
print("accuracy score for tranning data:",accuracy_score(y_train, classifier.predict(X_train)))
trainning_accuracy.append(accuracy_score(y_train, classifier.predict(X_train)))

print ("classification_report :",classification_report(y_test,y_pred))

#CHECK ACCURACY WITH CROSS-VALIDATION(=10)
acc_cv=cross_val_score(classifier, X, y, cv=10, scoring='accuracy').mean()
print ("ACCURACY WITH CROSS_VALIDATION :",acc_cv)
Accuracy_CV.append(acc_cv)
model_names.append("SVC")
```

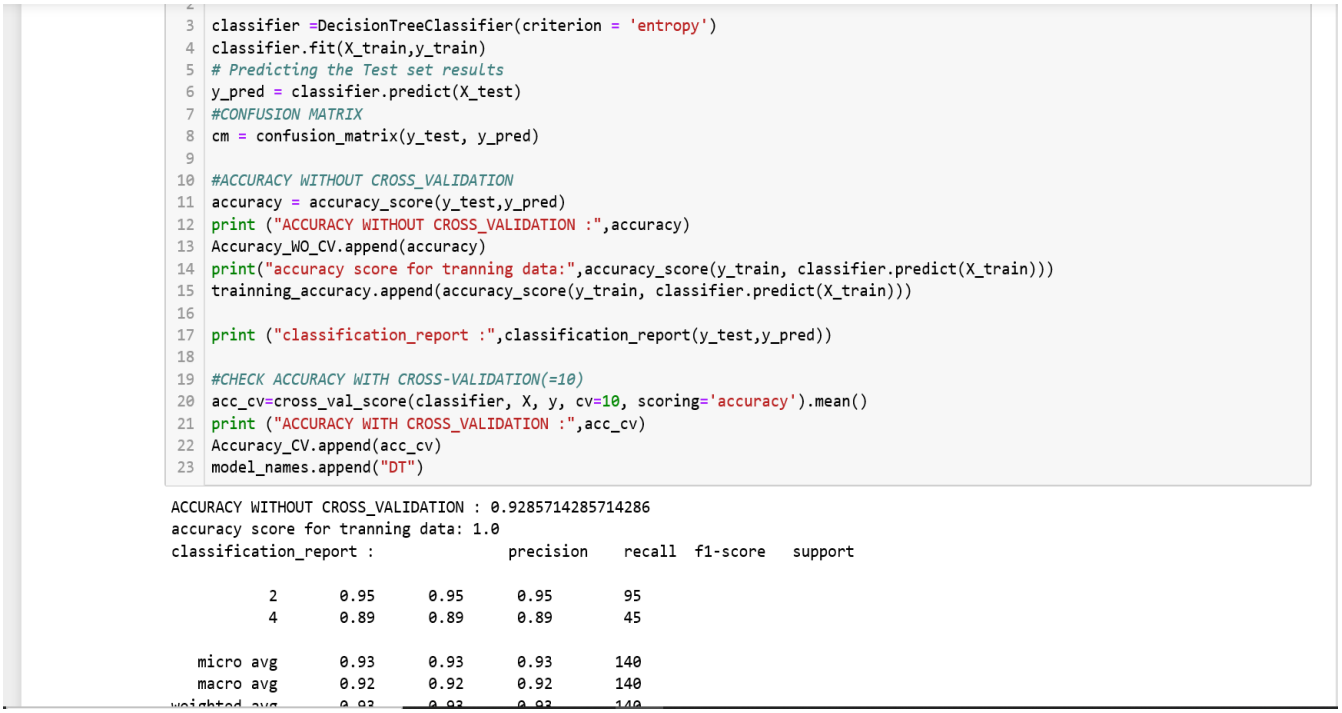
```
ACCURACY WITHOUT CROSS_VALIDATION : 0.9714285714285714
accuracy score for tranning data: 0.9677996422182469
classification_report :               precision    recall  f1-score   support

           2       0.97      0.99      0.98        95
           4       0.98      0.93      0.95        45
```

Administrator: Anaconda Prompt

Figure 13.

Decision Tree:

Decision Tree is a supervised, nonparametric machine learning algorithm. Used for both classification as well as regression problems. It is a graphical representation of tree like structure with all possible solutions. It helps to reach a decision based on certain conditions. Decision on how to split heavily impacts accuracy of decision tree. It splits nodes based on available input variables. Selects the input variable resulting in best homogenous dataset.



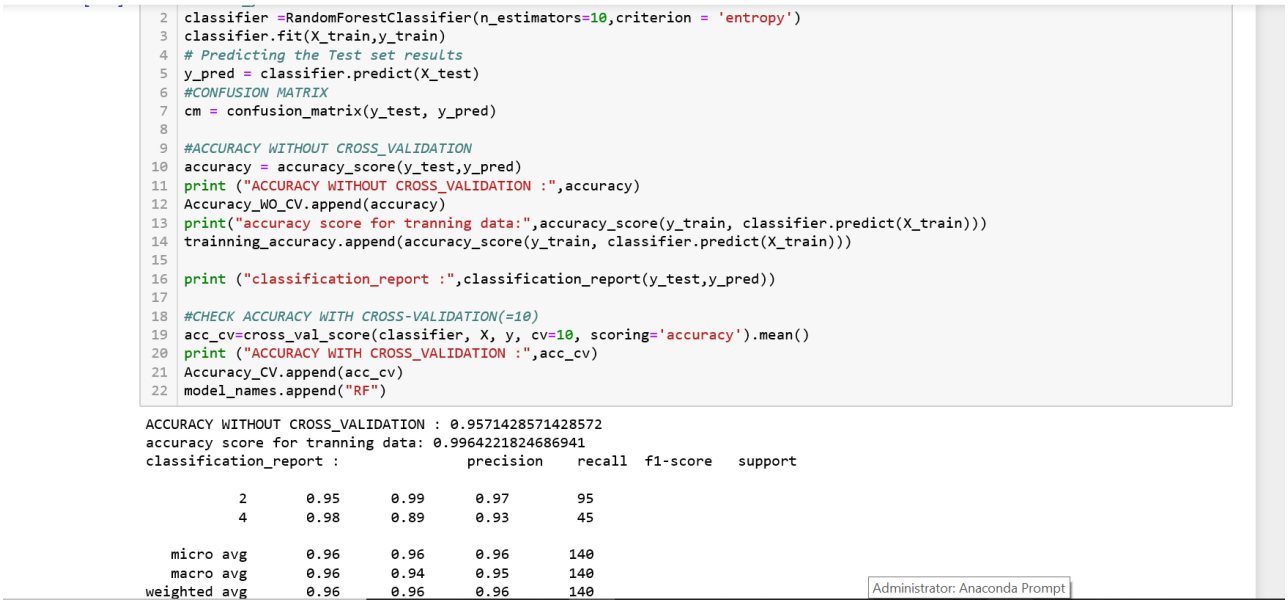
```
classifier =DecisionTreeClassifier(criterion = 'entropy')
classifier.fit(X_train,y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
#CONFUSION MATRIX
cm = confusion_matrix(y_test, y_pred)

#ACCURACY WITHOUT CROSS_VALIDATION
accuracy = accuracy_score(y_test,y_pred)
print ("ACCURACY WITHOUT CROSS_VALIDATION :",accuracy)
Accuracy_WO_CV.append(accuracy)
print("accuracy score for tranning data:",accuracy_score(y_train, classifier.predict(X_train)))
trainning_accuracy.append(accuracy_score(y_train, classifier.predict(X_train)))

print ("classification_report :",classification_report(y_test,y_pred))

#CHECK ACCURACY WITH CROSS-VALIDATION(=10)
acc_cv=cross_val_score(classifier, X, y, cv=10, scoring='accuracy').mean()
print ("ACCURACY WITH CROSS_VALIDATION :",acc_cv)
Accuracy_CV.append(acc_cv)
model_names.append("DT")
```

```
ACCURACY WITHOUT CROSS_VALIDATION : 0.9285714285714286
accuracy score for tranning data: 1.0
classification_report :               precision    recall  f1-score   support

           2       0.95      0.95      0.95        95
           4       0.89      0.89      0.89        45

   micro avg       0.93      0.93      0.93       140
   macro avg       0.92      0.92      0.92       140
weighted avg       0.93      0.93      0.93       140
```

Figure 14.

Random Forest:

Random forest is the most simple and widely used algorithm. Used for both classification and regression. It is an ensemble of randomized decision trees. Each decision tree gives a vote for the prediction of target variable. Random forest choses the prediction that gets the most vote.An ensemble learning model aggregates multiple machine learning models to give a better performance. In random forest we use multiple random decision trees for a better accuracy. Random Forest is a ensemble bagging algorithm to achieve low prediction error. It reduces the variance of the individual decision trees by randomly selecting trees and then either average them or picking the class that gets the most vote**.**



```
classifier =RandomForestClassifier(n_estimators=10,criterion = 'entropy')
classifier.fit(X_train,y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
#CONFUSION MATRIX
cm = confusion_matrix(y_test, y_pred)

#ACCURACY WITHOUT CROSS_VALIDATION
accuracy = accuracy_score(y_test,y_pred)
print ("ACCURACY WITHOUT CROSS_VALIDATION :",accuracy)
Accuracy_WO_CV.append(accuracy)
print("accuracy score for tranning data:",accuracy_score(y_train, classifier.predict(X_train)))
trainning_accuracy.append(accuracy_score(y_train, classifier.predict(X_train)))

print ("classification_report :",classification_report(y_test,y_pred))

#CHECK ACCURACY WITH CROSS-VALIDATION(=10)
acc_cv=cross_val_score(classifier, X, y, cv=10, scoring='accuracy').mean()
print ("ACCURACY WITH CROSS_VALIDATION :",acc_cv)
Accuracy_CV.append(acc_cv)
model_names.append("RF")
```

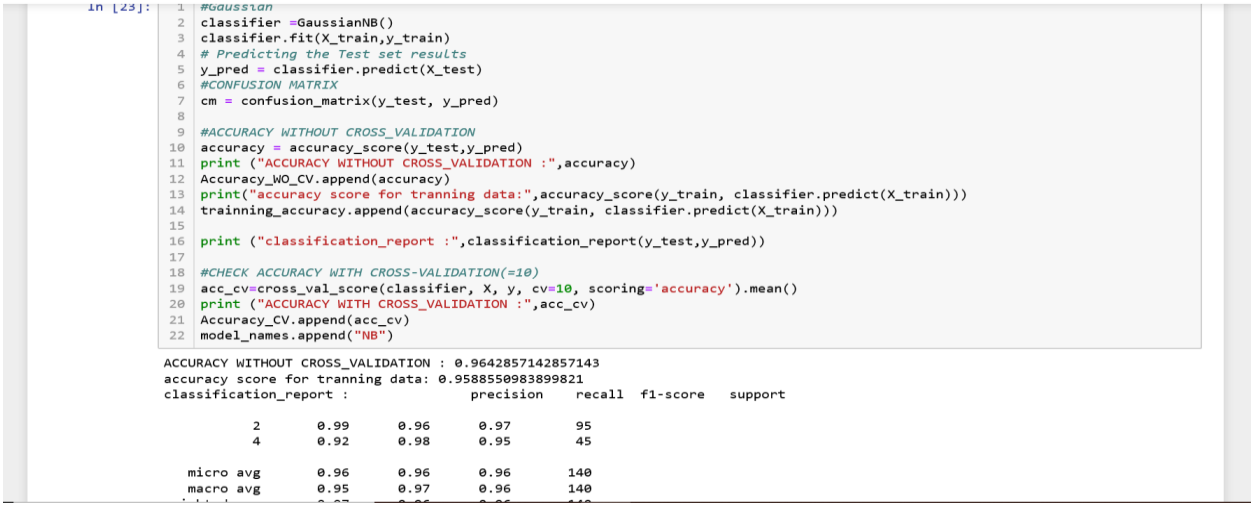
```
ACCURACY WITHOUT CROSS_VALIDATION : 0.9571428571428572
accuracy score for tranning data: 0.9964221824686941
classification_report :               precision    recall  f1-score   support

           2       0.95      0.99      0.97        95
           4       0.98      0.89      0.93        45

   micro avg       0.96      0.96      0.96       140
   macro avg       0.96      0.94      0.95       140
weighted avg       0.96      0.96      0.96       140
```

Administrator: Anaconda Prompt

Figure 15.

Naïve Bayes Algorithm:

Naïve Bayes are probabilistic classifiers, which are based on Bayes Theorem [8]. In Naïve Bayes each value is marked independent of the other values and features. Each value contributes independently to the probability. The higher the probabilistic value the higher are the chances of data point belonging to that class or category. Naïve Bayes algorithm uses the concept of Maximum Likelihood for prediction. This algorithm is fast and can be used for making real time predictions such as sentiment analysis**.**



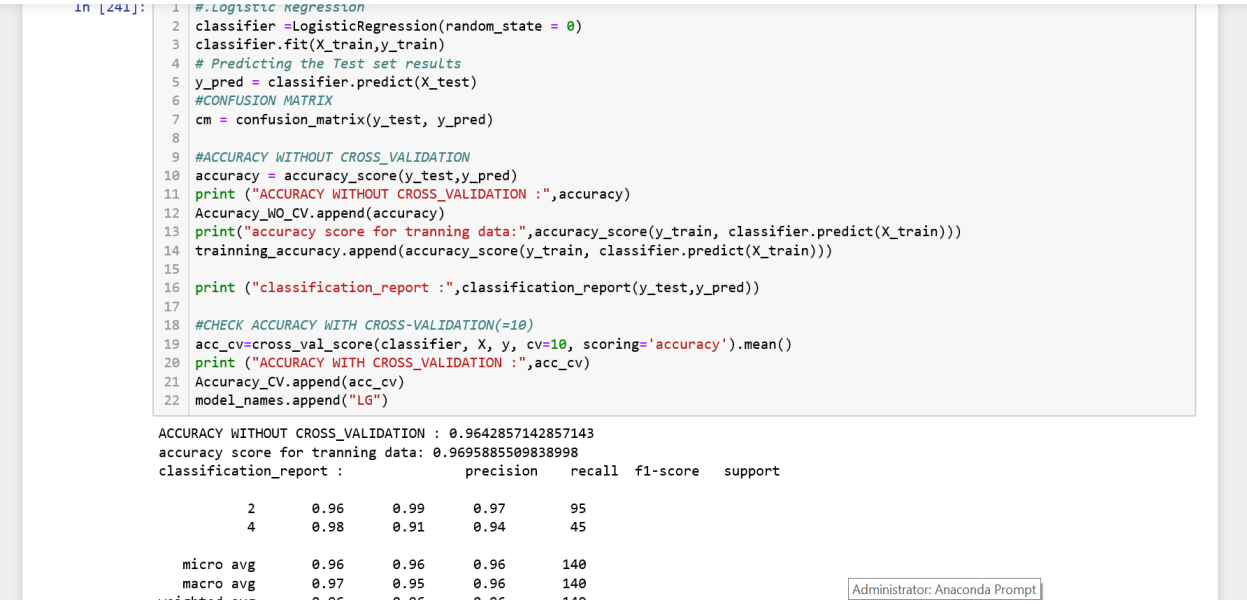
```
In [23]:
#Gaussian
classifier =GaussianNB()
classifier.fit(X_train,y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
#CONFUSION MATRIX
cm = confusion_matrix(y_test, y_pred)

#ACCURACY WITHOUT CROSS_VALIDATION
accuracy = accuracy_score(y_test,y_pred)
print ("ACCURACY WITHOUT CROSS_VALIDATION :",accuracy)
Accuracy_WO_CV.append(accuracy)
print("accuracy score for tranning data:",accuracy_score(y_train, classifier.predict(X_train)))
trainning_accuracy.append(accuracy_score(y_train, classifier.predict(X_train)))

print ("classification_report :",classification_report(y_test,y_pred))

#CHECK ACCURACY WITH CROSS-VALIDATION(=10)
acc_cv=cross_val_score(classifier, X, y, cv=10, scoring='accuracy').mean()
print ("ACCURACY WITH CROSS_VALIDATION :",acc_cv)
Accuracy_CV.append(acc_cv)
model_names.append("NB")
```

```
ACCURACY WITHOUT CROSS_VALIDATION : 0.9642857142857143
accuracy score for tranning data: 0.9588550983899821
classification_report :               precision    recall  f1-score   support

           2       0.99      0.96      0.97        95
           4       0.92      0.98      0.95        45

   micro avg       0.96      0.96      0.96       140
   macro avg       0.95      0.97      0.96       140
```

Figure 16.

Logistic Regression:

The logistic function. The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.



```
In [241]:
#.Logistic Regression
classifier =LogisticRegression(random_state = 0)
classifier.fit(X_train,y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
#CONFUSION MATRIX
cm = confusion_matrix(y_test, y_pred)

#ACCURACY WITHOUT CROSS_VALIDATION
accuracy = accuracy_score(y_test,y_pred)
print ("ACCURACY WITHOUT CROSS_VALIDATION :",accuracy)
Accuracy_WO_CV.append(accuracy)
print("accuracy score for tranning data:",accuracy_score(y_train, classifier.predict(X_train)))
trainning_accuracy.append(accuracy_score(y_train, classifier.predict(X_train)))

print ("classification_report :",classification_report(y_test,y_pred))

#CHECK ACCURACY WITH CROSS-VALIDATION(=10)
acc_cv=cross_val_score(classifier, X, y, cv=10, scoring='accuracy').mean()
print ("ACCURACY WITH CROSS_VALIDATION :",acc_cv)
Accuracy_CV.append(acc_cv)
model_names.append("LG")
```

```
ACCURACY WITHOUT CROSS_VALIDATION : 0.9642857142857143
accuracy score for tranning data: 0.9695885509838998
classification_report :               precision    recall  f1-score   support

           2       0.96      0.99      0.97        95
           4       0.98      0.91      0.94        45

   micro avg       0.96      0.96      0.96       140
   macro avg       0.97      0.95      0.96       140
```

Administrator: Anaconda Prompt

Figure 17.

Compare Model Accuracy:

Compare model accuracy of above mentioned Algorithms using Cross Validation and without Cross Validation.

```
In [29]:
ddf = pd.DataFrame(list(zip(model_names,Accuracy_WO_CV,Accuracy_CV,trainning_accuracy)),
                   columns =['Models', 'Accuracy_without_cv','CV_acc','x_train_ytrain_acc'])
ddf
```

Figure 18.

Out[29]:

	Models	Accuracy_without_cv	CV_acc	x_train_ytrain_acc
0	KNN	0.978571	0.971388	0.967800
1	SVC	0.971429	0.964344	0.967800
2	DT	0.935714	0.935750	1.000000
3	RF	0.957143	0.957262	0.998211
4	NB	0.964286	0.960079	0.958855
5	LG	0.964286	0.961528	0.969589

Figure 19.

Graph plot:

```
ax.set_title('CV_ACC v/s Accuracy_without_cv')
ax.set_xticks(index + bar_width / 2)
ax.set_xticklabels(("KNN","SVC","DT","RF","NB","LG"))
ax.legend()
plt.show()
```

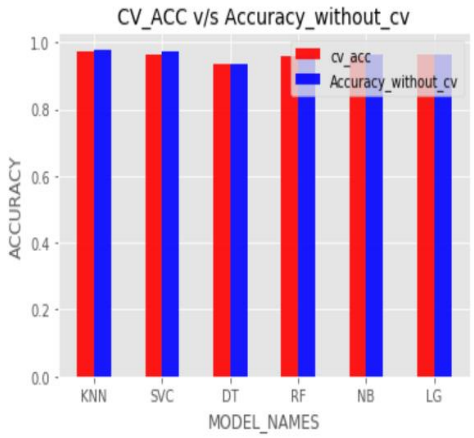
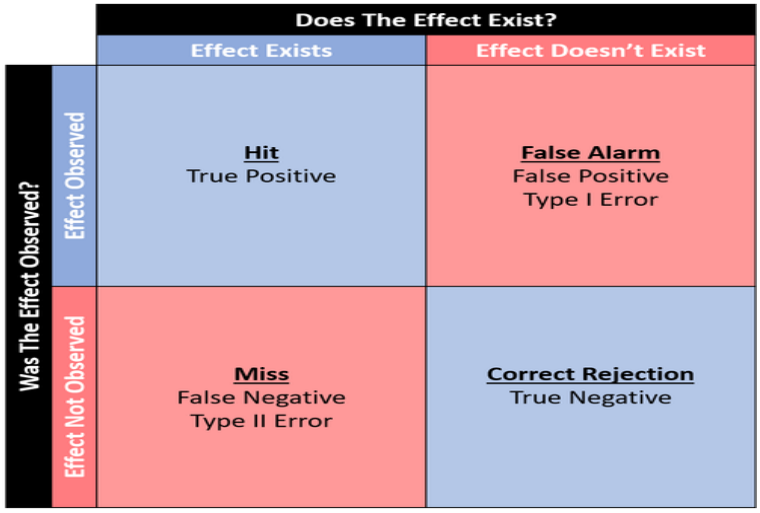


Figure 20.

***NOTES:**

Confusion Matrix:

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

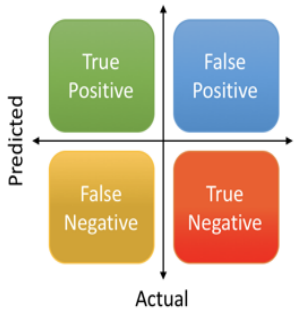


Precession and recall:

$$\text{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \text{ or } \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \text{ or } \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\text{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$



Cross Validation:

Cross-validation is a technique for evaluating ML models by training several ML models on subsets of the available input data and evaluating them on the complementary subset of the data. Use cross-validation to detect overfitting, i.e, failing to generalize a pattern.

Conclusions:

The feature analysis show that there are few features with more predictive value for the diagnosis.. Different algorithms for classification are tried - with all variables and with variable selection. Generally, it is found that errors were increased when variables were decreased .According to the evaluations, in case of both techniques of Cross validation and without cross validation the best fitted model for this data is KNN and SVC algorithm followed by Linear Regression, Naïve Bayes, Random forest and Decision Tree. This is a big research area in biological as well as technical field. These types of analysis can help taking preventive measures to fight breast cancer.

References:

http://letslearn.ritsin.com/wp-content/uploads/2014/12/Sample.html

https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

https://medium.com/datadriveninvestor/decision-tree-and-random-forest-e174686dd9eb

https://www.geeksforgeeks.org/confusion-matrix-machine-learning/

http://scholarpedia.org/article/Support_vector_clustering

https://en.wikipedia.org/wiki/Cross-validation_(statistics)

https://www.dhs.wisconsin.gov/epht/breast.htm

https://towardsdatascience.com/precision-vs-recall-386cf9f89488
