Machine Learning: Breast Cancer Classification in python

By: Sheikh-Sedat Touray

Project Overview

- The dataset I will be using to do this classification is collected from Kaggle under the UCI repository. It is real data collected in Wisconsin on the Diagnosis of Breast Cancer patients.
- The diagnosis shows whether the conditions were Benign or Malignant.
- I will be doing a visual inspection of the dataset and clean it up using python and its many libraries including pandas, numpy, keras etc.
- I will also do some transformation to the data that I am most interested in this analysis, for example, If a value needs to be converted into integers for the sake of our statistical computation.
- I will create three models for testing and determine the most efficient model by Accuracy and then create a confusion matrix to further test the Accuracies of the models and the best among the rest.

Breast Cancer
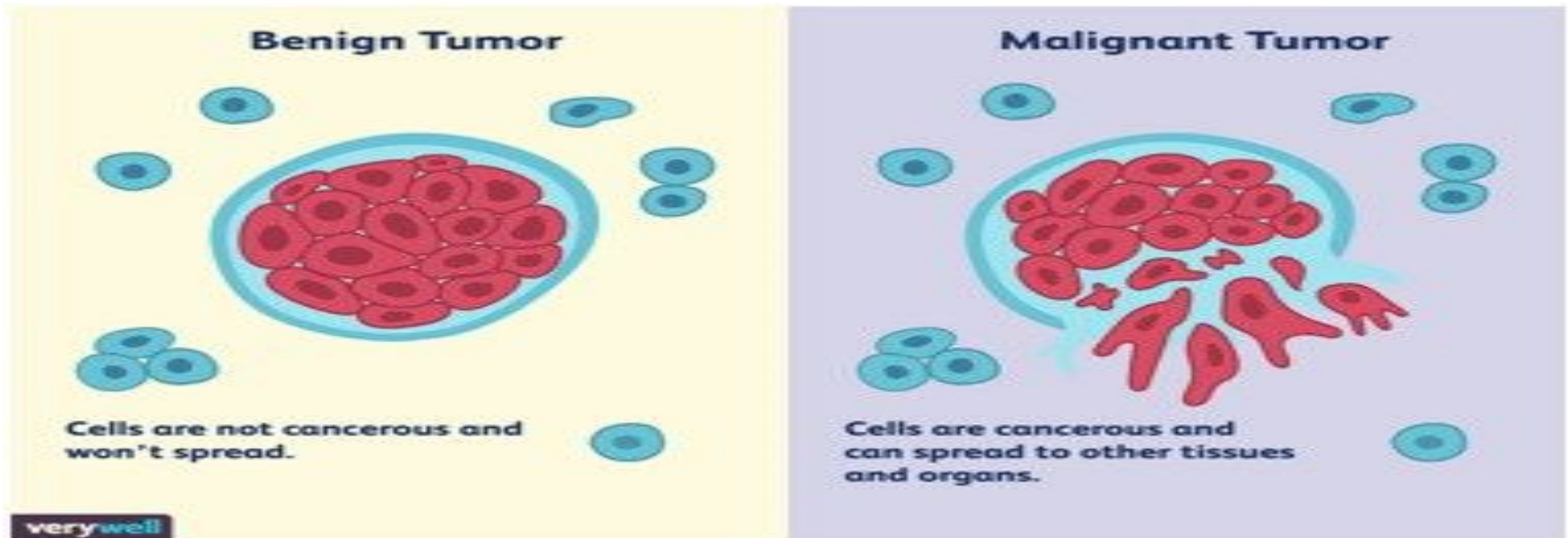
Visually inspecting and Cleaning data

I used the df.head function to examine the first five rows of the data and all the columns. It cannot be seen in this image, however, it has 33 columns and the last column has nan values

```
# Before I go forward I would exxamine my data and to do this i will look at the first 5 with the df.head function
df.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1

Continue Cleaning

```
[70]  #Now i want to examine the shape of my data(how many rows, and columns)
      df.shape
      #when i ran it found that i have 569 rows and 33 columns. so theres 569 patients with 33 data p

      (569, 33)
```

```
  ▶   #from the visual inspection of my data i saw that column 33 has nan values so i will drop
      df = df.dropna(axis=1)
      df.shape

      (569, 32)
```

Examining Diagnosis Column

```
[72]  #After visually inspecting my data and cleaning it out dropping nan values, now i c
      #by looking at the diagnosis column to determine number of Benign (B) or Malignant
      df['diagnosis'].value_counts()
```
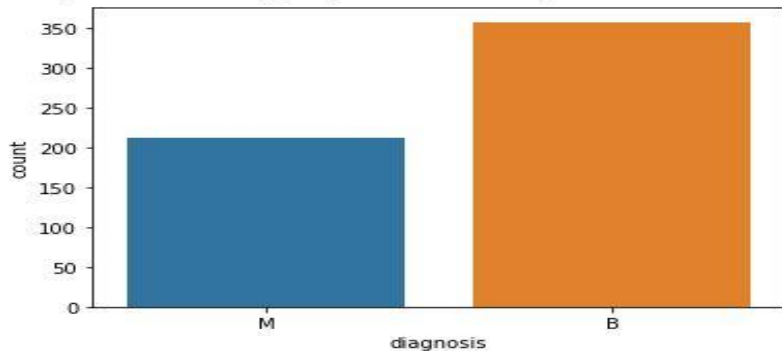
```
B    357
M    212
Name: diagnosis, dtype: int64
```

```
#creating a visualization for the the cells to see a better picture
sns.countplot(df['diagnosis'], label='A count of the Diagnosis')
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pa
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f2e7ca44d68>
```

Looking at dtypes

So in the diagnosis colum the data stored are object need to encode this the B replaced by 1s and 0s

```
[74]    # I am going to look at the data types to see whi
        #some of the strings into a number value int or fl
        df.dtypes
```

```
id                          int64
diagnosis                   object
radius_mean                 float64
texture_mean                float64
perimeter_mean              float64
area_mean                   float64
smoothness_mean             float64
compactness_mean            float64
concavity_mean              float64
concave points_mean         float64
symmetry_mean               float64
fractal_dimension_mean      float64
radius_se                   float64
texture_se                  float64
perimeter_se                float64
area_se                     float64
smoothness_se               float64
compactness_se              float64
concavity_se                float64
concave points_se           float64
symmetry_se                 float64
fractal_dimension_se        float64
radius_worst                float64
texture_worst               float64
perimeter_worst             float64
area_worst                  float64
smoothness_worst            float64
compactness_worst           float64
concavity_worst             float64
concave points_worst        float64
symmetry_worst              float64
fractal_dimension_worst     float64
dtype: object
```
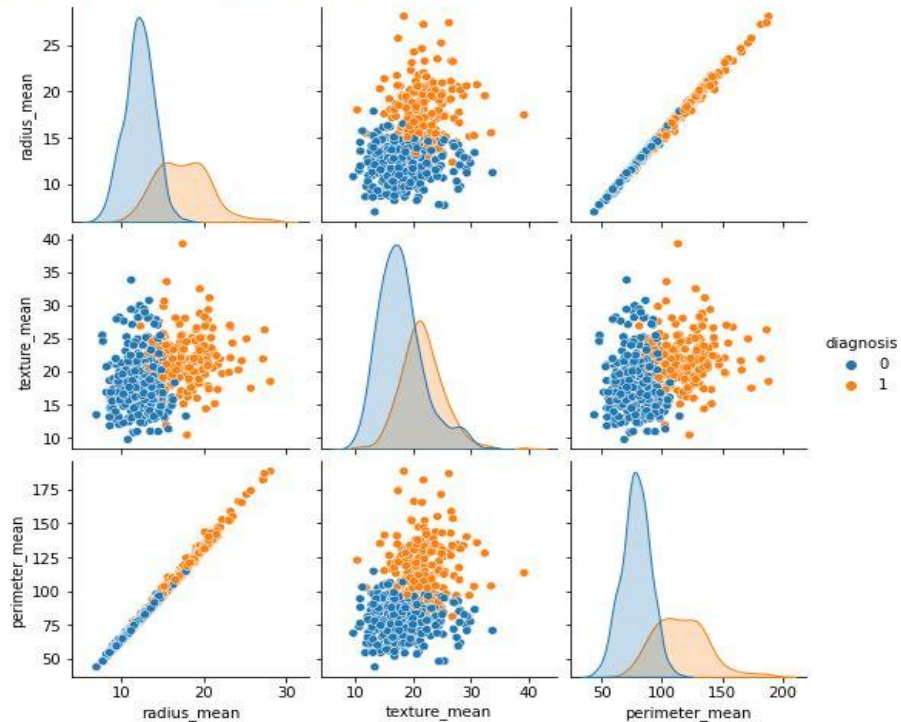
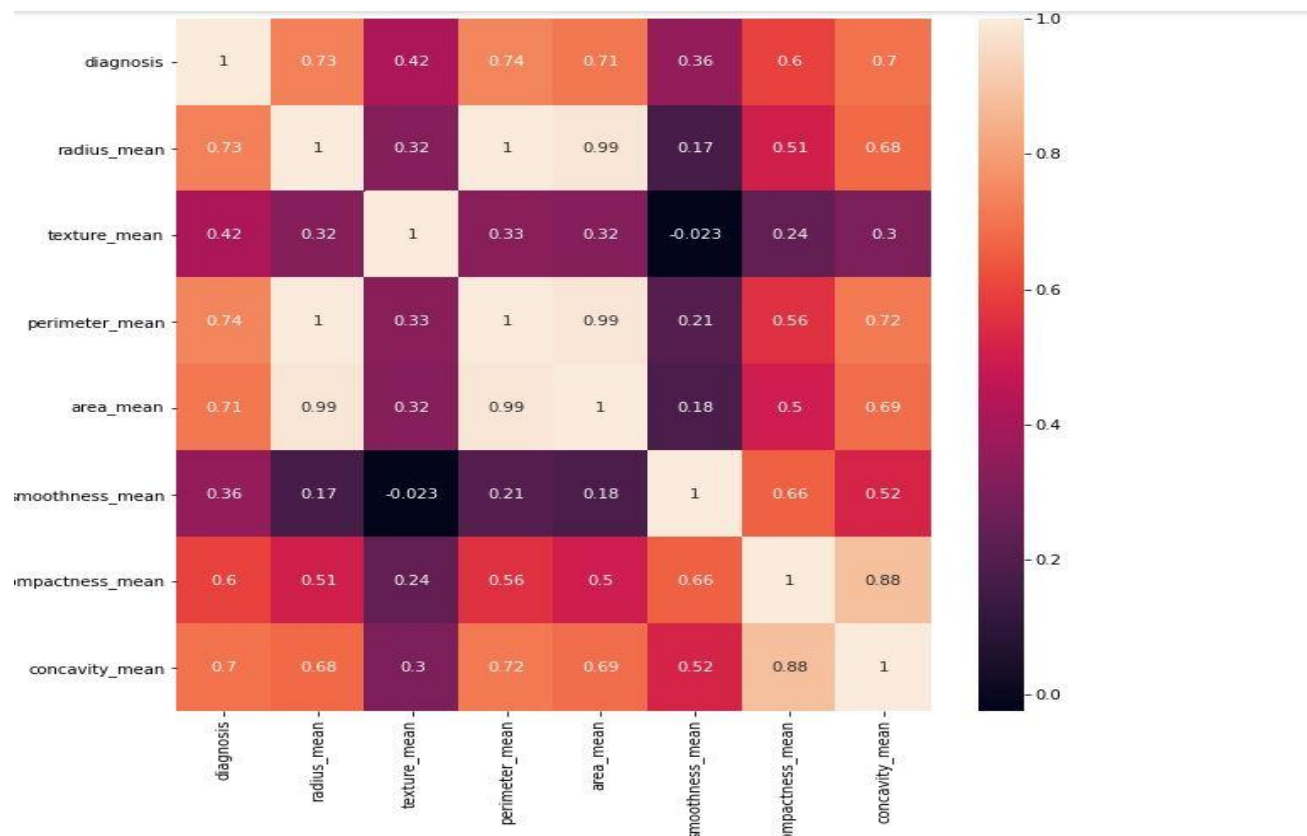# Encoding data to change objects to positive or negative, (1,0) respectively

```python
#encode categorical datavalues using sklearn and i will do this using iloc to i
from sklearn.preprocessing import LabelEncoder
labelencoder_Y = LabelEncoder()
labelencoder_Y.fit_transform(df.iloc[:,1].values)
```

```
array([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, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 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, 0, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 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, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 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, 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, 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, 1, 1, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 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, 1, 1, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 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, 0])
```

```python
# create a pair plot from to comapre including all rows but stops at column 4
sns.pairplot(df.iloc[:,1:5], hue='diagnosis')
```

```
<seaborn.axisgrid.PairGrid at 0x7f2e75734b00>
```

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
diagnosis	1	0.73	0.42	0.74	0.71	0.36	0.6	0.7
radius_mean	0.73	1	0.32	1	0.99	0.17	0.51	0.68
texture_mean	0.42	0.32	1	0.33	0.32	-0.023	0.24	0.3
perimeter_mean	0.74	1	0.33	1	0.99	0.21	0.56	0.72
area_mean	0.71	0.99	0.32	0.99	1	0.18	0.5	0.69
smoothness_mean	0.36	0.17	-0.023	0.21	0.18	1	0.66	0.52
compactness_mean	0.6	0.51	0.24	0.56	0.5	0.66	1	0.88
concavity_mean	0.7	0.68	0.3	0.72	0.69	0.52	0.88	1

Correlation is better shown on the heatmap with the values annotated

Testing & Training Models

Now i am going to split my dataset into dependent and independent (x,y) variables where y will be diagnosis and x is cancer features and use 80% for training and 20% testing.

```python
#split the dataset into independent variables , y has the diagnosis and x has the cancer features

x = df.iloc[:,2:32].values
y = df.iloc[:,1].values
```

```python
[89] #split the dataset 80% training and 20% testing
     x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state = 0)

     #now I am going to scale my data
     sc = StandardScaler()
     x_train = sc.fit_transform(x_train)
     x_test = sc.fit_transform(x_test)
```

Three Models and their Accuracy

```
[95]  # I am going to create a function for the the models I am going to use for my regress
      def models(x_train, y_train):
        #logistic regression
        log = LogisticRegression(random_state=0)
        log.fit(x_train, y_train)

        #Decision tree
        tree = DecisionTreeClassifier(random_state = 0)
        tree.fit(x_train, y_train)

        #Random forest classifier
        forest = RandomForestClassifier(n_estimators = 10,random_state = 0)
        forest.fit(x_train, y_train)

        # I also want to print the models accuracy on the trainig data
        print('[0] Logistics Regression Training Accuracy:', log.score(x_train, y_train))
        print('[1] Decision Tree Training Accuracy:', tree.score(x_train, y_train))
        print('[2] Random forest classifier Training Accuracy:', forest.score(x_train, y_tr

        return log, tree, forest
```

```
model=models(x_train, y_train)
```

```
[0] Logistics Regression Training Accuracy: 0.9906103286384976
[1] Decision Tree Training Accuracy: 1.0
[2] Random forest classifier Training Accuracy: 0.9953051643192489
```

Create a confusion matrix and determine Best Model

I am going to create a confusion matrix or error matrix to easily account for errors. It is usually in a table layout with true positive, false positive, true negative and false negative values.

```
#So now we will test the accuracy of out model on our test da
#to print the confusion matrix for all models and their accur
for i in range(len(model)):
    print('model',i)
    cm = confusion_matrix(y_test, model[i].predict(x_test))
    TP = cm[0][0] #true positive
    TN = cm[0][1] #true negative
    FN = cm[1][0] #false negative
    FP = cm[0][1] #false positive
    print(cm)
    print('Testing Accuracy = ', (TP+TN)/(TP+TN+FN+FP))
    print()
```

```
model 0
[[86  4]
 [ 2 51]]
Testing Accuracy =  0.9375

model 1
[[79 11]
 [ 2 51]]
Testing Accuracy =  0.8737864077669902

model 2
[[86  4]
 [ 1 52]]
Testing Accuracy =  0.9473684210526315
```

Comparing a prediction using the Random Forest Classifier vs An
actual Test shows this: Very accurate

```
#lets examine the prediction of the Random forest classifer
pred = model[2].predict(x_test)
print(pred)
print()
print(y_test)
```

```
[1 0 0 0 0 0 0 0 0 0 1 0 0 0 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 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0 1 0 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 1 1 0 0 1 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 1 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 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 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0 1 0 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 1 0 1 0 1 1 0
 1 1 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]
```
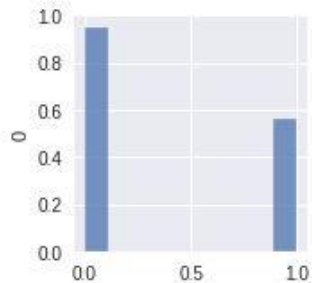
Visualizing our prediction of the cancer results comparing it with an actual test

`<seaborn.axisgrid.PairGrid at 0x7fbc5f837438>`
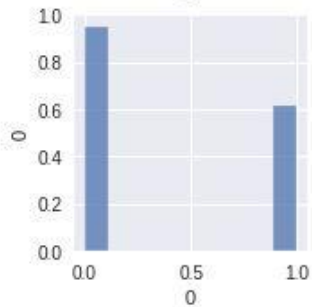
Prediction plot

Diagnosis results plot

Model Accuracy and Loss