Machine Learning Engineer Nanodegree

Capstone Project Report

Poli Naidu Sigilipalli

Jan 17th 2019

Definition

Project Overview

Breast cancer is the most common malignancy among women, accounting for nearly 1 in 3 cancers diagnosed among women in the US, and it is the second leading cause of cancer death among women. It occurs as a result of abnormal growth of cells in the breast tissue, commonly referred to as a tumor. A tumor does not mean cancer – tumor can be benign (non cancerous), pre-malignant (pre-cancerous), or malignant (cancerous). So given results of FNA test , I will predict whether cancer is benign or malignant.

Problem Statement

Given breast cancer results from breast fine needle aspiration (FNA) test (is a quick and simple procedure to perform, which removes some fluid or cells from a breast lesion or cyst (a lump, sore or swelling) with a fine needle similar to a blood sample needle). Since this build a model that can classify a breast cancer tumor using two training classification:

1= Malignant (Cancerous) - Present
0= Benign (Not Cancerous) -Absent

Since the labels in the data are discrete, the predication falls into two categories, (i.e. Malignant or benign). In machine learning this is a classification problem. Thus, the goal is to classify whether the breast cancer is benign or malignant and predict the recurrence and non-recurrence of malignant cases after a certain period. To achieve this we have used machine learning classification methods to fit a function that can predict the discrete class of new input.

Features and Description:

- Id - ID number
- Diagnosis - The diagnosis of breast tissues (M = malignant, B = benign)
- radius_mean - mean of distances from center to points on the perimeter
- texture_mean-standard deviation of gray-scale values
- perimeter_mean-mean size of the core tumor
- area_mean
- smoothness_mean - mean of local variation in radius lengths
- compactness_mean - mean of perimeter^2 / area - 1.0
- concavity_mean - mean of severity of concave portions of the contour
- concave points_mean - mean for number of concave portions of the contour
- symmetry_mean
- fractal_dimension_mean - mean for "coastline approximation" - 1
- radius_se - standard error for the mean of distances from center to points on the perimeter
- texture_se - standard error for standard deviation of gray-scale values
- perimeter_se
- area_se
- smoothness_se - standard error for local variation in radius lengths
- compactness_se - standard error for perimeter^2 / area - 1.0
- concavity_se - standard error for severity of concave portions of the contour
- concave points_se - standard error for number of concave portions of the contour
- symmetry_se
- fractal_dimension_se - standard error for "coastline approximation" - 1
- radius_worst - "worst" or largest mean value for mean of distances from center to points on the perimeter

- texture_worst - "worst" or largest mean value for standard deviation of gray-scale values
- perimeter_worst
- area_worst
- smoothness_worst - "worst" or largest mean value for local variation in radius lengths
- compactness_worst - "worst" or largest mean value for perimeter^2 / area - 1.0
- concavity_worst - "worst" or largest mean value for severity of concave portions of the contour
- concave points_worst - "worst" or largest mean value for number of concave portions of the contour
- symmetry_worst
- fractal_dimension_worst - "worst" or largest mean value for "coastline approximation" – 1

Metrics

I have used confusion matrix and classification report as my metrics. I am more concerned about false negatives in my model and confusion matrix shows every detail so clearly. It shows True positives,true negatives ,false positives and false negatives.

Classification report gives us information about precision,recall and f_score. Out of these, I am more concentrating on recall value.

Recall : Recall is a measure that tells us what proportion of patients that actually had cancer was diagnosed by the algorithm as having cancer.

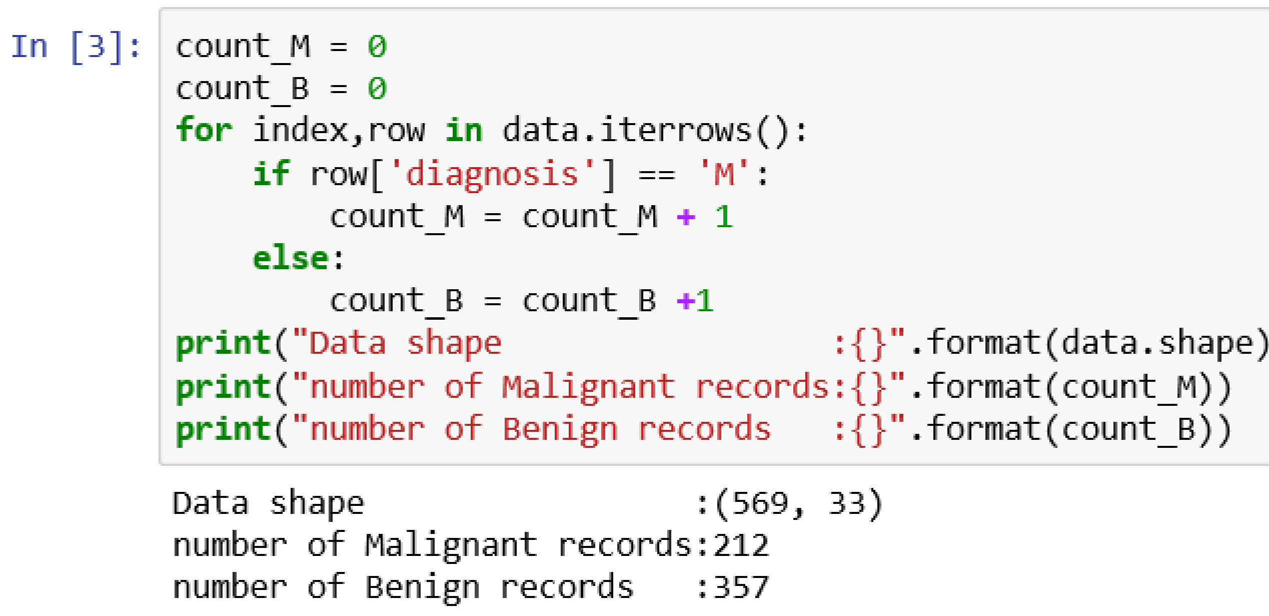
Recall = TP/(TP+FN)

Analysis

Data Exploration

I’ve used this dataset from the link : https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

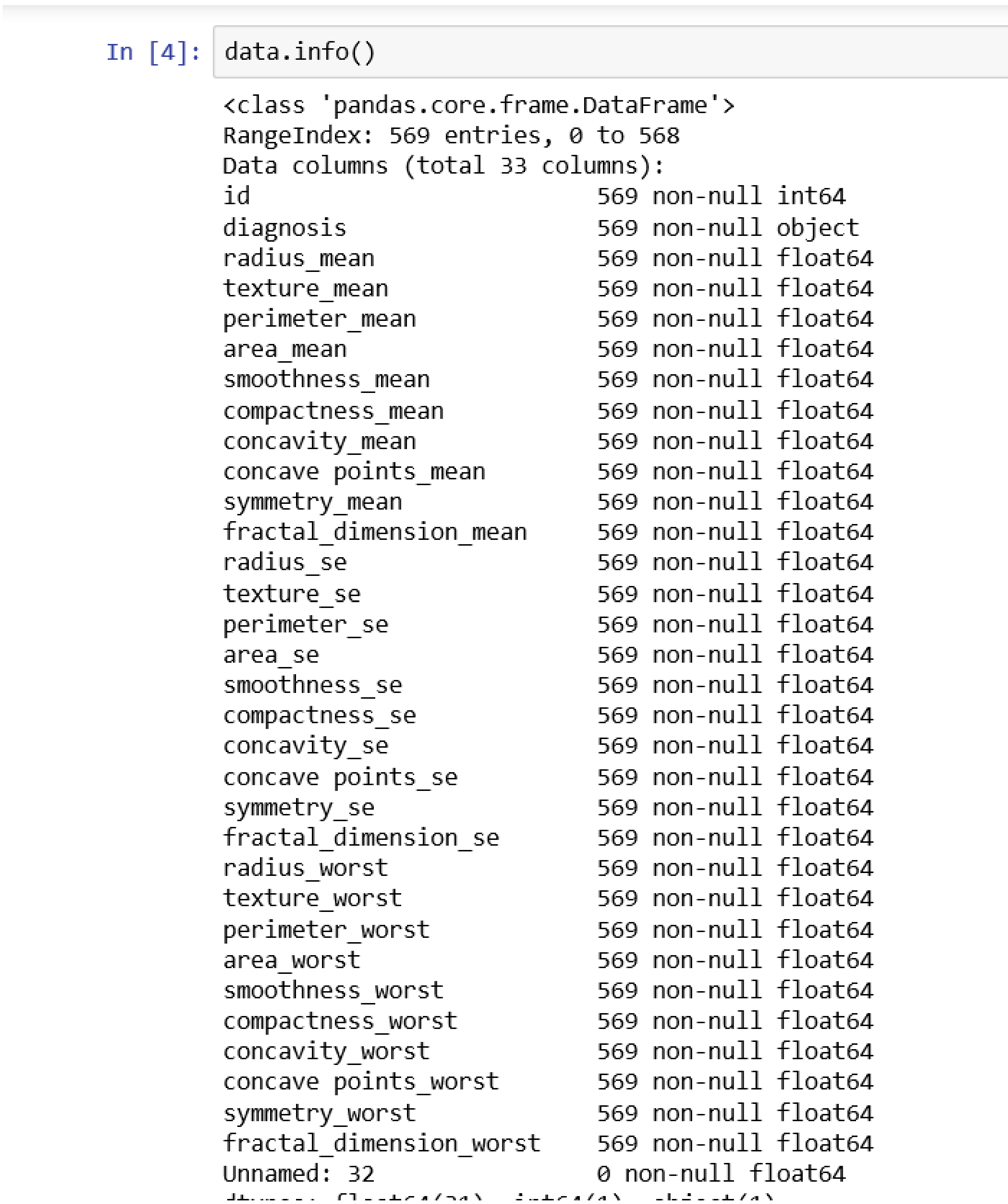
Some basic knowledge about the given data.



```
In [3]: count_M = 0
        count_B = 0
        for index,row in data.iterrows():
            if row['diagnosis'] == 'M':
                count_M = count_M + 1
            else:
                count_B = count_B +1
        print("Data shape                 :{}".format(data.shape)
        print("number of Malignant records:{}".format(count_M))
        print("number of Benign records   :{}".format(count_B))
```

```
Data shape                 :(569, 33)
number of Malignant records:212
number of Benign records   :357
```

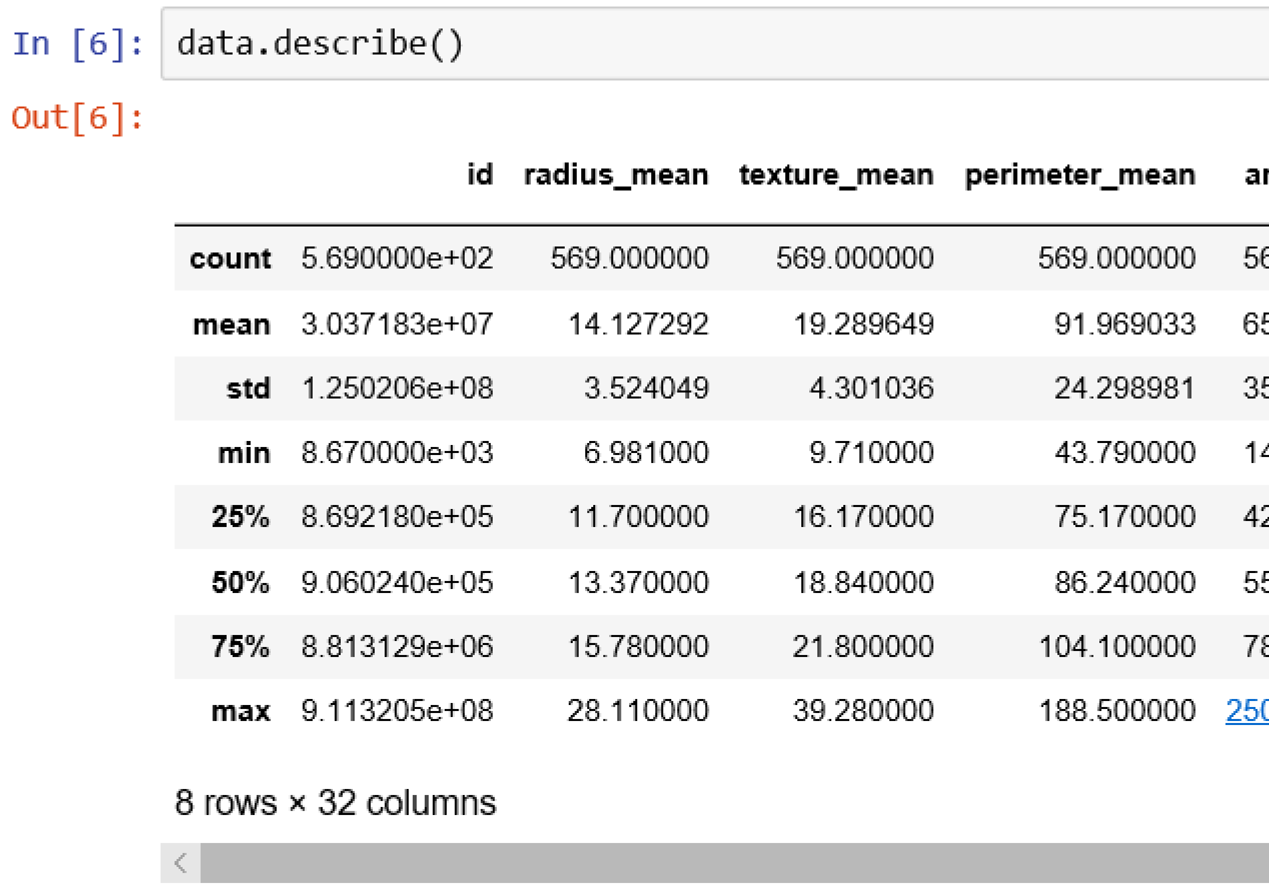
The **“info()”** method provides a concise summary of the data; from the output, it provides the type of data in each column, the number of non-null values in each column, and how much memory the data frame is using.



```
In [4]: data.info()

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

Descriptive statistics about the data.

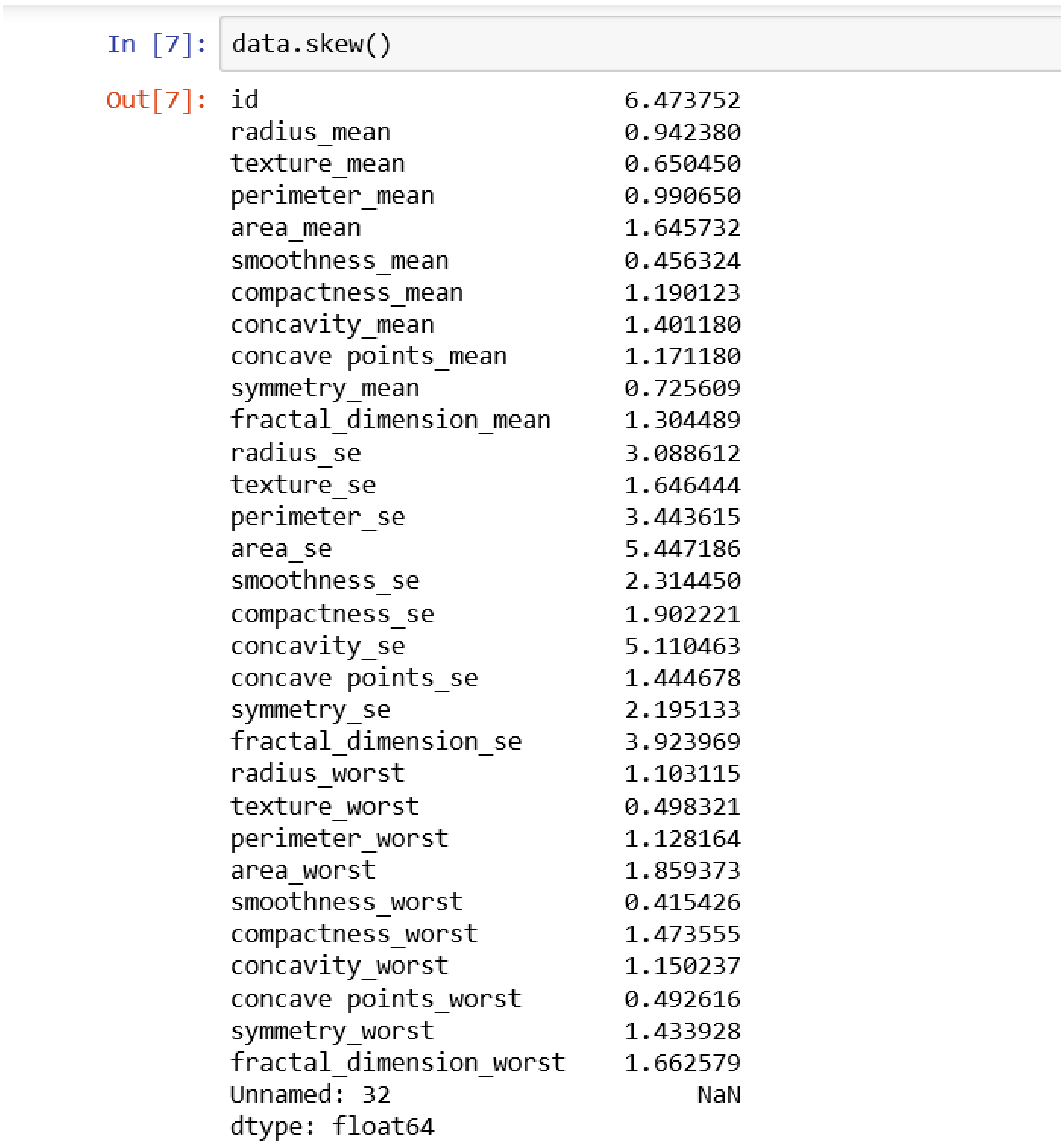


```
In [6]: data.describe()
Out[6]:
```

	id	radius_mean	texture_mean	perimeter_mean	ar
count	5.690000e+02	569.000000	569.000000	569.000000	56
mean	3.037183e+07	14.127292	19.289649	91.969033	65
std	1.250206e+08	3.524049	4.301036	24.298981	35
min	8.670000e+03	6.981000	9.710000	43.790000	14
25%	8.692180e+05	11.700000	16.170000	75.170000	42
50%	9.060240e+05	13.370000	18.840000	86.240000	55
75%	8.813129e+06	15.780000	21.800000	104.100000	78
max	9.113205e+08	28.110000	39.280000	188.500000	250

8 rows × 32 columns

skew values of the data to identify whether the distribution is left skewed or right skewed.



```
In [7]: data.skew()
```

```
Out[7]: id                         6.473752
        radius_mean                0.942380
        texture_mean               0.650450
        perimeter_mean             0.990650
        area_mean                  1.645732
        smoothness_mean            0.456324
        compactness_mean           1.190123
        concavity_mean             1.401180
        concave points_mean        1.171180
        symmetry_mean              0.725609
        fractal_dimension_mean     1.304489
        radius_se                  3.088612
        texture_se                 1.646444
        perimeter_se               3.443615
        area_se                    5.447186
        smoothness_se              2.314450
        compactness_se             1.902221
        concavity_se               5.110463
        concave points_se          1.444678
        symmetry_se                2.195133
        fractal_dimension_se       3.923969
        radius_worst               1.103115
        texture_worst              0.498321
        perimeter_worst            1.128164
        area_worst                 1.859373
        smoothness_worst           0.415426
        compactness_worst          1.473555
        concavity_worst            1.150237
        concave points_worst       0.492616
        symmetry_worst             1.433928
        fractal_dimension_worst    1.662579
        Unnamed: 32                     NaN
        dtype: float64
```

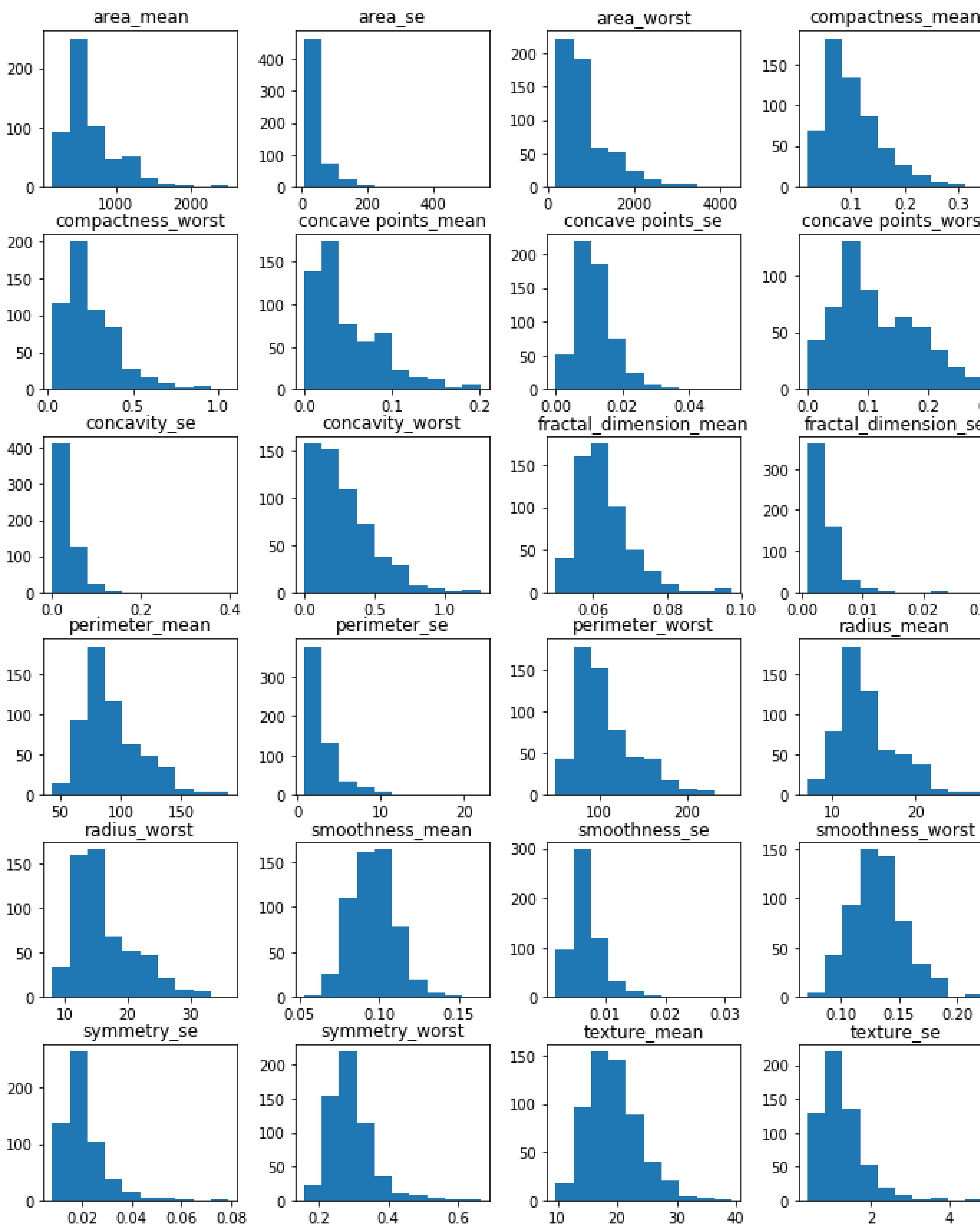
Visualization

Histogram

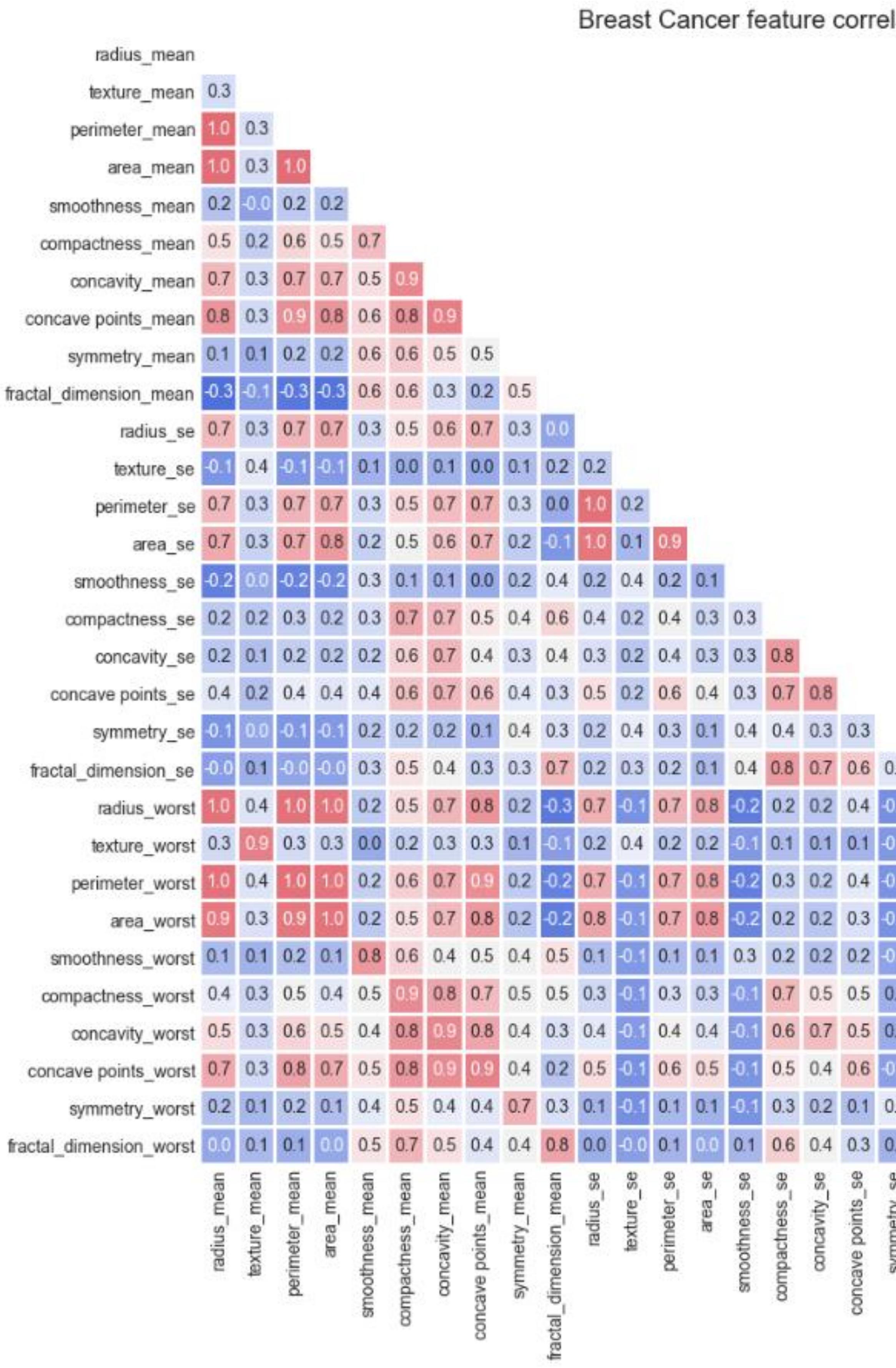
Histograms are commonly used to visualize numerical variables. A histogram is similar to a bar graph after the values of the variable are grouped (binned) into a finite number of intervals (bins).

Histograms group data into bins and provide you a count of the number of observations in each bin. From the shape of the bins you can quickly get a feeling for whether an attribute is Gaussian, skewed or even has an exponential distribution. It can also help you see possible outliers.

```
hist_mean = data_mean.hist(bins=10,figsize=(15,15),grid=False)
```



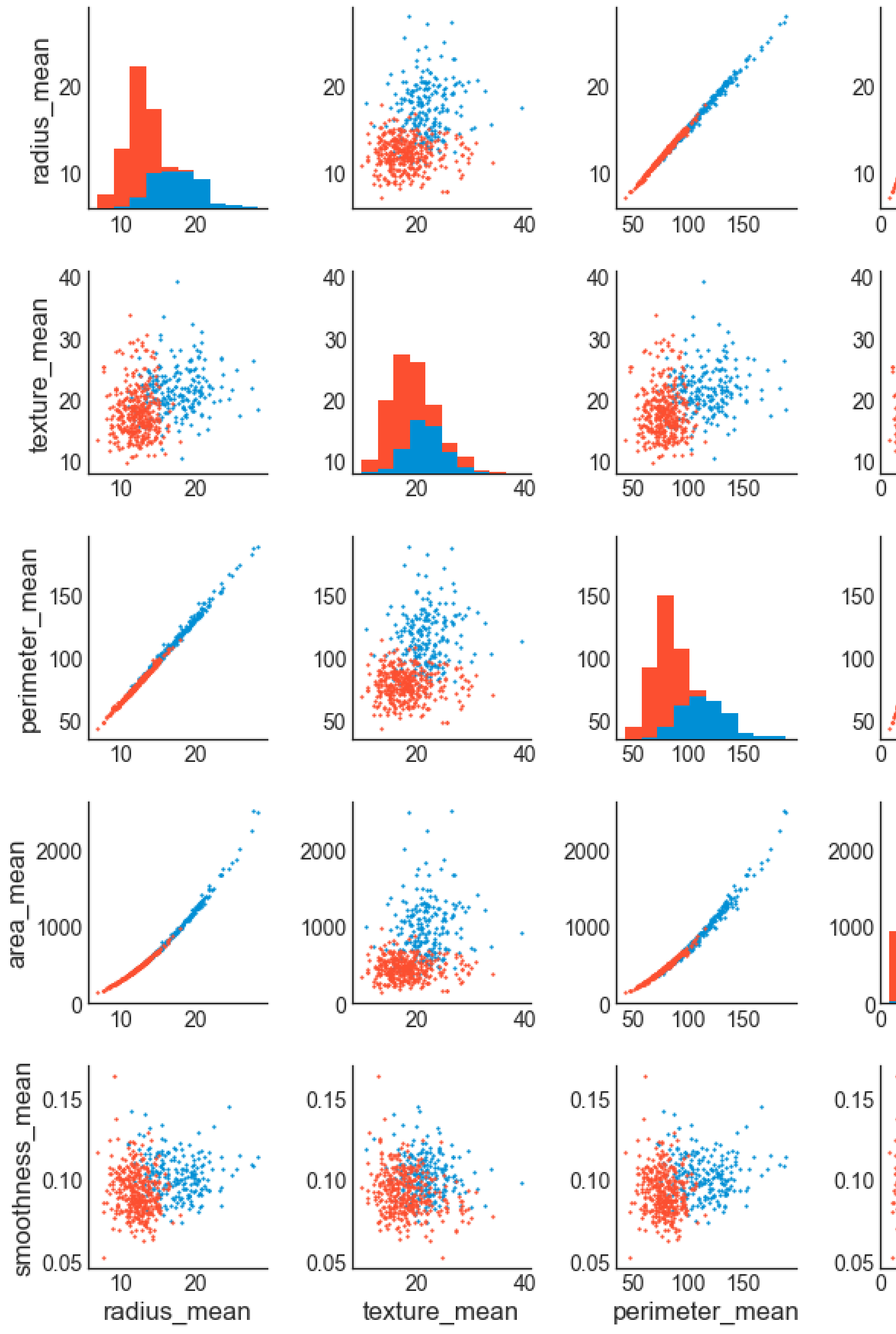
Correlation Matrix


Breast Cancer feature correl
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave points_se
symmetry_se
fractal_dimension_se
radius_worst
texture_worst
perimeter_worst
area_worst
smoothness_worst
compactness_worst
concavity_worst
concave points_worst
symmetry_worst
fractal_dimension_worst

We can see correlation between so many attributes . Mean,se and worst values of radius are strongly correlated with mean,se and worst values of concave_points,area,perimeter etc. likewise, so many other values are correlating with many other attributes.

I’ve used scatter plot pair grid to identify relation between any two attributes

```
g = g.map_diag(plt.hist)
g = g.map_offdiag(plt.scatter,s=3)
```



Algorithm and techniques

In my project, I’ve used Support vector machines as my bench mark model.

Support Vector Machines

A Support Vector Machine (SVM) is a binary linear classification whose decision boundary is explicitly constructed to minimize generalization error. It is a very powerful and versatile Machine Learning model, capable of performing linear or nonlinear classification, regression and even outlier detection.

SVM is well suited for classification of complex but small or medium sized datasets.

It performs exceptionally well with higher dimentional data. Our data is composed of 32 columns which is very complex and less records (only 569). So, I’ve chosen this as benchmark model.

Other techniques

Other supervised learning techniques which I’ve used for this project are:

1. Logistic Regression
2. KNeighbors Classifier
3. Decision Tree Classifier
4. Naïve Bayes

I’ve compared each one of them with the other using confusion matrix. As I’ve mentioned I’ve concentrated more on recall value . Atlast after checking each one of them , I found that svm was giving more recall,fi_score and even more accuracy_scores than other algorithms. When the test was performed on the non-standardised values svm didn’t perform well. But once standardisation was performed. SVM performed exceptionally well.

Benchmark model

I’ve used Support vector machines as my benchmark model.

Methodology

Data Pre-Processing

There are no null or missing values in the data. I’ve normalised all the features using StandardScaler(). I’ve converted target class variables ‘M’ and ‘B’ to 1 and 0. Then I’ve split my data into training and testing sets.

Feature decomposition using PCA

Feature decomposition using PCA

```
In [16]: from sklearn.decomposition import PCA
         pca = PCA(n_components=10)
         fit = pca.fit(NewX)
```

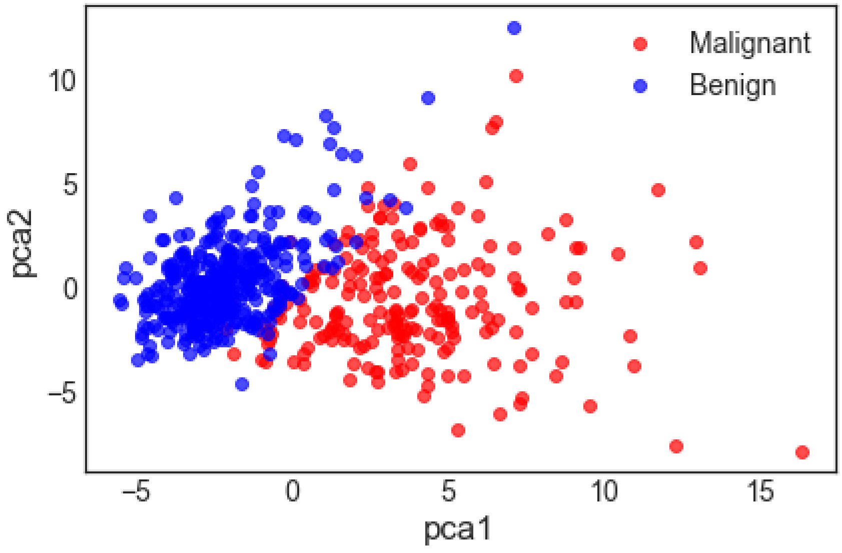
```
In [17]: pca_X = pca.transform(NewX)

         PCA_dataframe = pd.DataFrame()

         PCA_dataframe['pca1'] = pca_X[:,0]
         PCA_dataframe['pca2'] = pca_X[:,1]

         plt.plot(PCA_dataframe['pca1'][data.diagnosis == 'M'],PCA_dataframe['pca2'][
         plt.plot(PCA_dataframe['pca1'][data.diagnosis == 'B'],PCA_dataframe['pca2'][

         plt.xlabel('pca1')
         plt.ylabel('pca2')
         plt.legend(['Malignant','Benign'])
         plt.show()
         print(pca.explained_variance_ratio_)
```



```
[ 0.44272026  0.18971182  0.09393163  0.06602135  0.05495768  0.04024522
  0.02250734  0.01588724  0.01389649  0.01168978]
```

Implementation

1. Then I've created **predictive model using SVM** and my observations are like this.

   Observation

	Precision	**Recall**	**F1-score**
0	0.97	0.97	0.97
1	0.94	0.96	0.95
Avg/total	0.96	0.96	0.96

1.There are two possible predicted classes: "1" and "0". Malignant = 1 (indicates presence of cancer cells) and Benign= 0 (indicates absence).

2. The classifier made a total of 188 predictions (i.e 188 patients were being tested for the presence of breast cancer).

3. Out of those 188 cases, the classifier predicted "yes" 68 times, and "no" 120 times.

4. In reality, 67 patients in the sample have the disease, and 121 patients do not.

5. It predicted 3 effected patients as Benign and 4 benign patient as malignant.

I have compared it with several other models namely **Logistic regression, Kneighborsclassifier, decision tree classifier**, **Naïve bayes** etc**.**

2. I’ve created another predictive model using **Logistic regression** and my observations are like this.

Observation

	Precision	**Recall**	**F1-score**
0	0.97	0.98	0.98
1	0.97	0.94	0.95
Avg/total	0.97	0.97	0.97

3. Predictive model using **Decision Tree Classifier** and my observations are like this.

   Observation

	Precision	Recall	F1-score
0	0.97	0.93	0.95
1	0.88	0.94	0.91
Avg/total	0.93	0.93	0.93

4. Predictive model using **GaussianNB** and my observations are like this.

   Observation

	Precision	Recall	F1-score
0	0.94	0.93	0.94
1	0.88	0.90	0.89
Avg/total	0.92	0.92	0.92

Refinement

To find the optimized parameters I used Grid Search method where we provide list of parameters in dictionary and model runs and score the model on different parameters combination and optimizes the model. I also did k-fold cross val on train data for splitting. Parameter tuning have improved results for certain models but for svm recall score have decreased.

Results

Model Evaluation and Validation

I used SVM as my benchmark model. When I’ve used it with my original data,it didn’t perform good. But after standardisation, it performed exceptionally well. Whichever metric we take, it performed better than any other model. Then I've used k-fold cross val and trained my model which further increased the performance of the model.

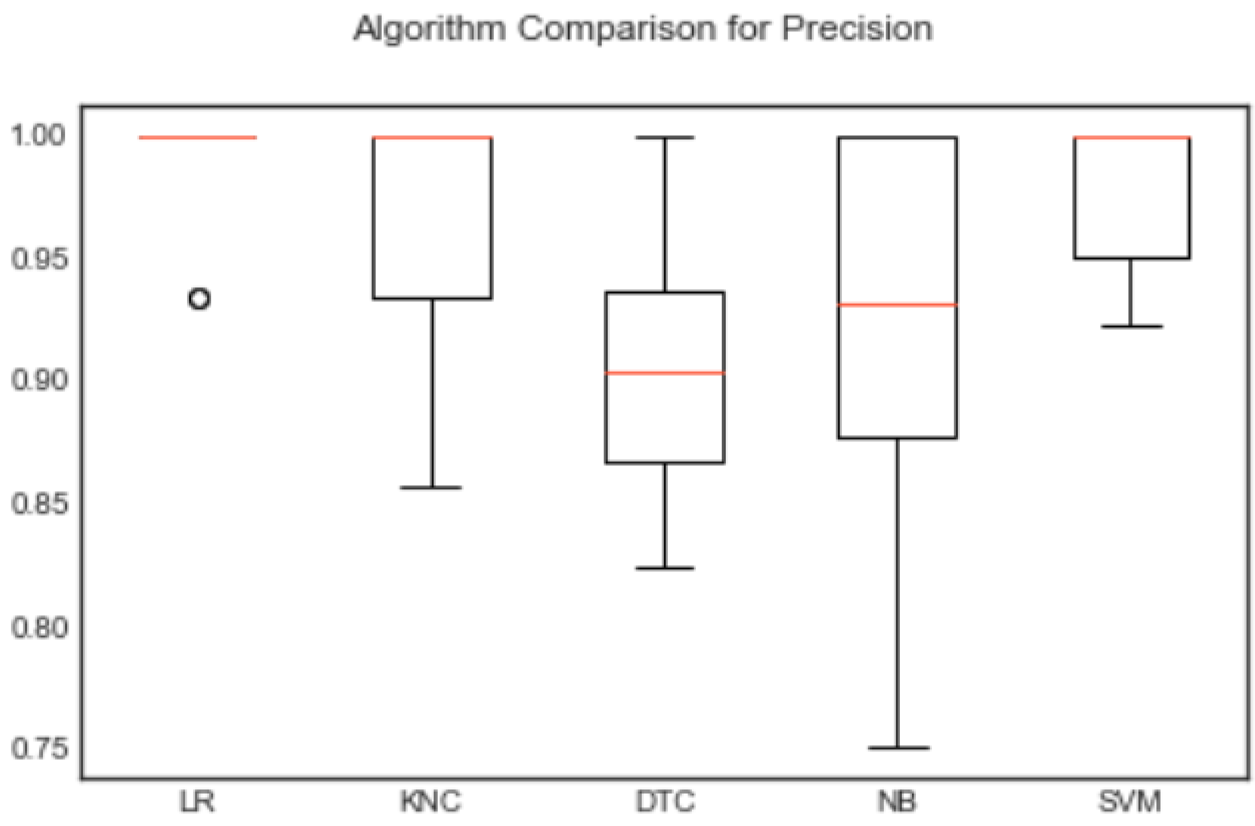
Justification

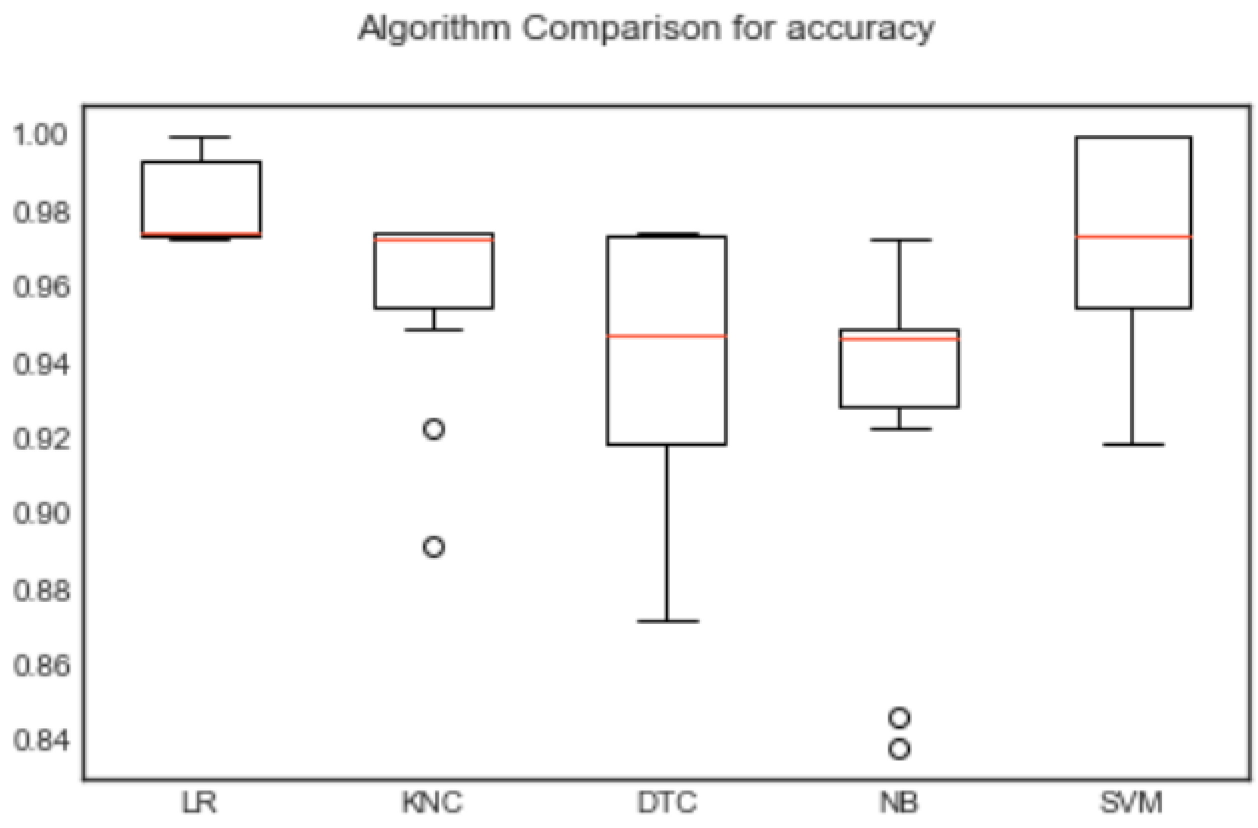
I've used recall as a threshold and also have taken care of getting a considerable precision score also. As you can see the observations which I've mentioned above. SVM's scores are higher than any other model

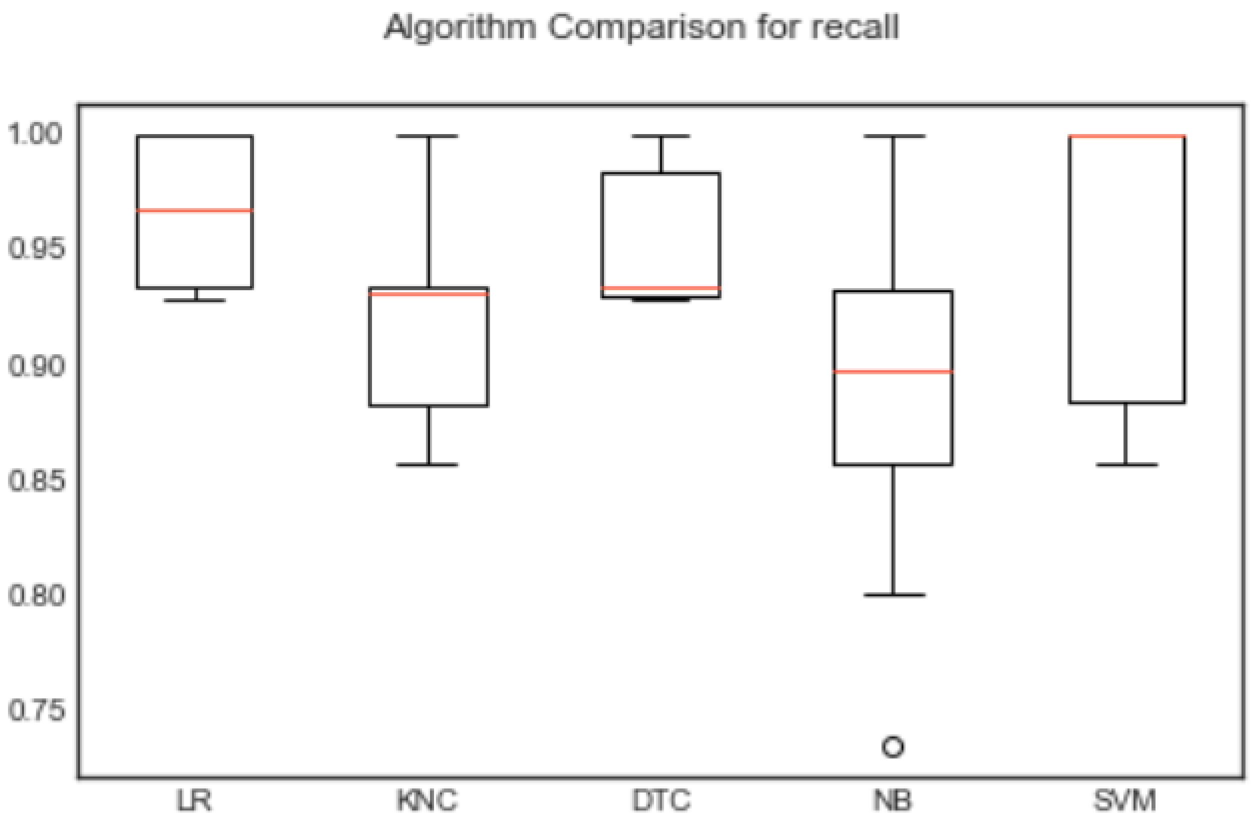
Conclusion

Free–form Visualization

Box plots of different models using different metrics




Algorithm Comparison for accuracy
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
LR
KNC
DTC
NB
SVM



Reflection

Firstly I've analysed data using different functions like **head(),info(),describe(),skew()**. After that I've performed data visualization using various techniques like histograms to check how data is distributed and visualized correlation between attributes using correlation matrix and pair grid. Next, I've done label encoding to convert 'M's and 'B's to 1 and 0. I've preprocessed data by splitting dataset into training and testing sets and I've reduced dimensions in the data using pca . But however I didn't use the transformed dataset as it's giving less score in my chosen metric. Then I've trained data sets using different models and evaluated them using different metrics like confusion matrix,classification report etc. When I trained my model without standardisation my benchmark model,svm gave less score. I thought I was completely wrong choosing it as my model. After that, I've realized that I forgot to standardise it. That part I’ve found it difficult. Remaining everything was good.

Improvement

We can improve the model scores by taking the k-fold validation in Grid Search and tuning should be even more complex rather than taking 2 to 3 parameters experimenting with all the parameters may have given us better results.