FACULTY OF ENGINEERING, ENVIRONMENT & COMPUTING

(7082CEM)

Coursework

Demonstration of a Big Data Program

MODULE LEADER: Dr. Marwan Fuad

Student Name: Shruti Suresh Nair

SID: 9830541

BREAST CANCER CLASSIFICATION USING LOGISTIC REGRESSION

I can confirm that all work submitted is my own: Yes

BREAST CANCER CLASSIFICATION USING LOGISTIC REGRESSION

Introduction:

Breast Cancer

The term Breast cancer is a medical condition in which cells in the breast grow out of control. There are numerous kinds of breast cancer. The kind of breast cancer depends on which cells in the breast turn cancerous.

Breast cancer can occur in different areas of the breast. A breast comprises of three main parts namely the lobules, ducts, and the connective tissue. The lobules are the glands that are responsible for producing milk. The ducts are responsible for carrying milk to the nipple. The connective tissues are nothing but fibrous and fatty tissue that surrounds and holds everything together. Ducts and lobules are highly prone to Breast Cancer.

Breast cancer is invasive and can spread through blood vessels and lymph vessels to other parts of the body. When breast cancer spreads to other parts of the body, the condition is known as metastasized cancer. (Cdc.gov, 2020)

Introduction to Datasets

For this coursework, I have explored the two datasets on breast cancer and developed a Logistic Regression model to classify suspected cells as Benign or Malignant.

The coursework is inspired by *Predicting Breast Cancer - Logistic Regression* https://www.kaggle.com/jagannathrk/predicting-breast-cancer-logistic-regression

Datasets

1) **Dataset link**:
   (Google Docs, 2020)

   Dataset Description (AcadGild, 2020)

Column Name	datatype	Data Description
Complete_TCGA_ID	string	Unique ID for people
Gender	string	Gender of Patient
Age Initial_diag	integer	Patient's age at the time of admission
ER Status	string	Estrogen receptors they are either ER-positive (or ER+) cancers.

PR Status	string	Breast cancer cells known as progesterone receptors; the cancer affected cells are called as PR-positive breast cancer.
HER2 Final Status	string	HER2-positive breast cancer is when patient is positive for a protein called human epidermal growth factor receptor 2 (HER2)
Tumor	string	Numbers after the T describe the severity of the tumour in terms of size and spread. High T number tells us how large the tumour is and/or the more it has grown into nearby tissues.
Node	string	Numbers after the N describe the size, location, and/or the number of nearby lymph nodes affected by cancer. N0 represents cancer free lymph nodes.
Node_Coded	string	Positive represents that the patient has Cancer Negative represents that the patient is cancer free
Metastasis	string	M0 represents cancer free M1 represents cancer affected.
Metastasis_Coded	string	Positive the patient has cancer Negative represents that the patient is cancer free
AJCC_Stage	string	American Joint Committee on Cancer developed a classification system used for describing the extent of disease progression in cancer patients.
Converted_Stage	string	Changes in AJCC_stage. After treatment
Survival_Data_Form	string	
Vital_Status	string	The patient is living or dead/deceased.
Days_to_Date_of_Last_Contact	integer	The number of days passed since patient last contacted
Days_to_date_of_Death	integer	The number of days passed to patient's death

2) **Dataset 2**

Breast Cancer Wisconsin (Diagnostic) Data Set

(Wolberg, Nick Street and L. Mangasarian, 2020)

Attribute Information:

- **id**
- **diagnosis**: M stands for malignant, B stands for benign
- **radius**: distance from centre to perimetre
- **texture**: standard deviation of gray-scale values
- **perimeter**
- **area**
- **smoothness**: local variation in radius lengths
- **compactness**: defines the compactness
- **concavity**: severity of concave portions of the breast
- **concave points**: number of concave portions of the contour
- **symmetry**
- **fractal dimension**: "coastline approximation" – 1 (Wolberg, Nick Street and L. Mangasarian, 2020)

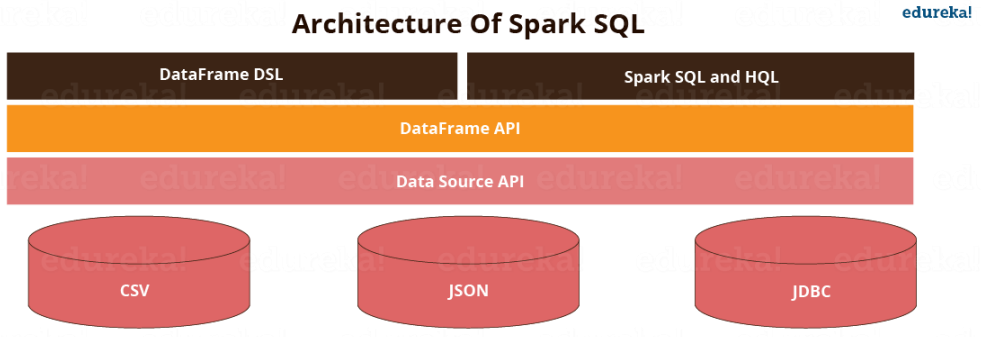
IMPLEMENTATION:

For the coursework I've used Pyspark with hive for the exploratory data analysis along with multiple packages to perform Logistic Regression for classification of the cells.

The libraries used are as listed below:

- **Pyspark.sql**

  Spark SQL one of the latest addition to Spark, integrates relational processing with Spark's functional programming API. Querying data either via SQL or via the Hive Query Language is supported by Pyspark.sql (Dayananda, 2020)



[Image source: Edureka May 2019]

- **Py4j**

Py4J allows Python programs to dynamically access Java objects in a Java Virtual Machine running in a Python interpreter. Methods are called by the Python interpreter whereas Java collections can be accessed easily via standard Python collection methods. Py4J also enables Java programs to call back Python objects. Py4J is distributed under the BSD license. (Py4j.org, 2020)

- **Pandas**
  Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. (Pandas.pydata.org, 2020)

- **Matplotlib**
  Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.. (Matplotlib.org, 2020)

- **NumPy**
  NumPy is the fundamental package for scientific computing with Python. It contains among other things:

  - N-dimensional array object
  - sophisticated functions
  - Methods for integrating C/C++ and Fortran code
  - Support for random number capabilities, linear algebra and Fourier transform. (Numpy.org, 2020)

- **Seaborn**
  It is data visualization library in python which is based on matplotlib. It provides interface for drawing descriptive statistical graphics. (Seaborn.pydata.org, 2020)

- **Scikit learn**
  Scikit-learn is a free software machine learning library for the Python programming language. (En.wikipedia.org, 2020)

- **mpld3**
  The mpld3 project brings together Matplotlib, graphing library, and D3js, the popular JavaScript library for creating interactive data visualizations for the web.(Mpld3.github.io, 2020)

Pyspark

Apache Spark is written in Scala programming language. To support Python with Spark, Apache Spark Community released a tool, PySpark. Using python we can work on RDD as well. Py4j is the library because of which that they are able to achieve all the functionalities. (Tutorialspoint.com, 2020)

Pyspark Installation (Tutorialspoint.com, 2020)

Step1: Install Java8 using the command **sudo apt install openjdk-8-jdk** on ubuntu

Run the command **java -version** to check the version

Step 2: Install Scala using the command **sudo apt-get install scala**
Run the command **scala -version** to check the version

Step 3: Install spark available online from https://www.apache.org/dyn/closer.lua/spark/spark-3.0.0-preview2/spark-3.0.0-preview2-bin-hadoop2.7.tgz

Step 4: Unzip the spark folder using the command **tar -zxvf (filename)**

Step 5: Update the bash file with the command **gedit** ~/**.bashrc**

Bash file:

```
export SPARK_HOME=/home/shruti/spark
export PATH=$PATH:$SPARK_HOME/bin
export PATH=$PATH:~/anaconda3/bin
export PYTHONPATH=$SPARK_HOME/python:$PYTHONPATH
export PYSPARK_DRIVER_PYTHON="jupyter"
export PYSPARK_DRIVER_PYTHON_OPTS="notebook"
export PYSPARK_PYTHON=python3
export PATH=$PATH:$JAVA_HOME/jre/bin
export PYTHONPATH=$SPARK_HOME/python/lib/py4j-0.9
src.zip:$PYTHONPATH
```

Save the update bash file using the command **source .bashrc**

Step 6: To run Pyspark run the command **# ./bin/pyspark**

Step7: Install pip3 which the python package installer using the command **sudo apt-get install python3-pip**

Step 8: Using pip3 install jupyter notebook **sudo apt install jupyter**

Step9: Finally installing all the required python libraries using pip3

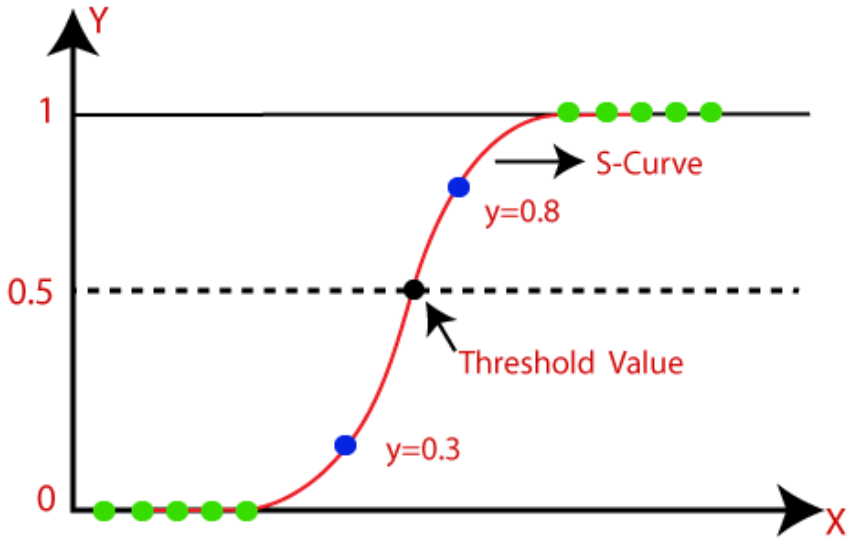
- pip3 install py4j
- pip3 install NumPy
- pip3 install matplotlib
- pip3 install seaborn
- pip3 install pandas

- pip3 install scikit-learn

Step10: Launch Jupyter Notebook from the terminal simply using the command **Jupyter notebook**

Logistic Regression (Pant, 2020)

Logistic Regression is used when that the data is linearly separable or classifiable and the outcome is Binary or Dichotomous. (Pant, 2020)



- The equation for logistic regression is:

$$\log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

[Image source: https://www.javatpoint.com/]

Applications of Logistic Regression

- Logistic regression is widely used in medical research, in microbiology sector to describe bacterial growth in 0/1 format.
- Logistic regression used to predict the risk of development of a particular disease based on patient's diagnostic.
- Logistic regression is extensively used in the banking sector to identify loan defaulters

PROGRAM

The program for my coursework is inspired by two Kaggle kernels listed below:

- **Breast cancer prediction**
- ***Predicting Breast Cancer - Logistic Regression***

JUPYTER NOTEBOOK

The link to my Jupyter notebook is as stated below:
https://drive.google.com/uc?export=download&id=1gqmSyCtMGYC-9kjc-wUNdn64QTOX3lEY

SQL QUERIES

```
In [26]: spark.sql("select AVG(Age_at_Initial_Pathologic_Diagnosis) from breastcancerdataset").show()

+-----------------------------------------------------+
|avg(CAST(Age_at_Initial_Pathologic_Diagnosis AS DOUBLE))|
+-----------------------------------------------------+
|                                   58.68571428571428|
+-----------------------------------------------------+
```

```
In [30]: spark.sql("select AVG(Age_at_Initial_Pathologic_Diagnosis) from breastcancerdataset where vital_status = 'LIVING' ").show()

+-----------------------------------------------------+
|avg(CAST(Age_at_Initial_Pathologic_Diagnosis AS DOUBLE))|
+-----------------------------------------------------+
|                                  58.180851063829785|
+-----------------------------------------------------+
```

```
In [29]: spark.sql("select AVG(Age_at_Initial_Pathologic_Diagnosis) from breastcancerdataset where vital_status = 'DECEASED' ").show()

+-----------------------------------------------------+
|avg(CAST(Age_at_Initial_Pathologic_Diagnosis AS DOUBLE))|
+-----------------------------------------------------+
|                                                 63.0|
+-----------------------------------------------------+
```

Let's analyse the queries mentioned above,

1) **Query:** The first query is to select the average age of patient at initial pathologic diagnosis from the dataset.
Output: 58.68 years.

2) **Query:** The second query is to find the average of patients whose vital status is 'Living'
Output: 58.18 years

3) **Query:** The third query is to find the average of patients whose vital status is 'Deceased'
Output: 63.0

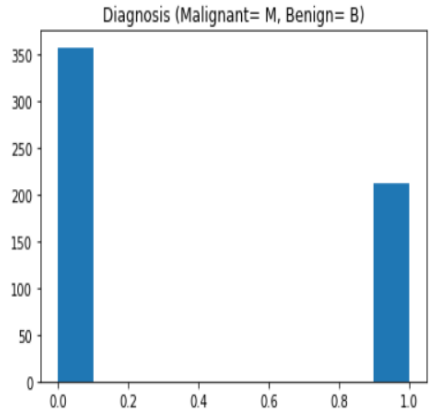
Observation: From the above queries we can understand that early diagnosis can improve the life expectancy. Patients below the age of 63 have a good chance of defeating the cancer rather than patients above 63

VISUALIAZATION

For visualizing the various attributes in the dataset, I have used the python library Matplotlib.

In [4]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# keeps the plots in one place. calls image as static pngs
%matplotlib inline
import matplotlib.pyplot as plt # side-stepping mpl backend
import matplotlib.gridspec as gridspec # subplots
import mpld3 as mpl

#Import models from scikit learn module:
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn import metrics
```

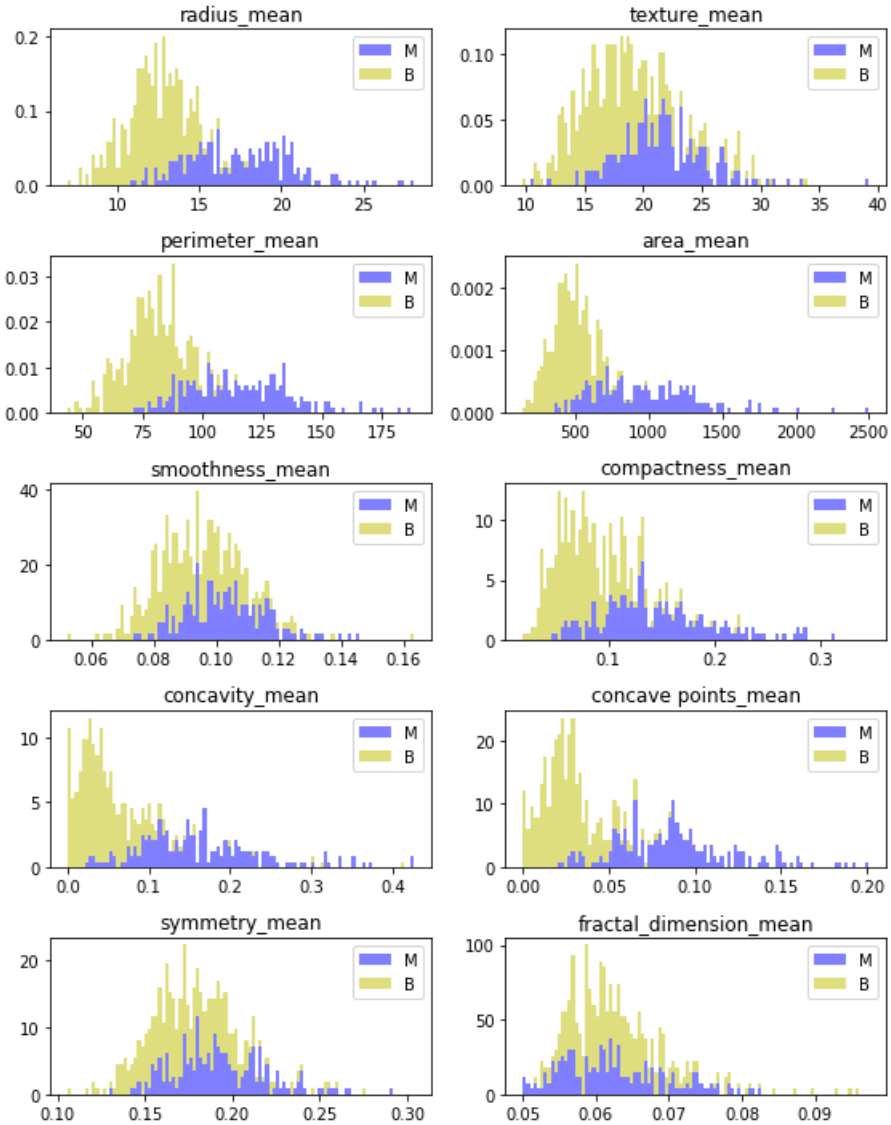
In [26]:
```
breastdf2.describe()
plt.hist(breastdf2['diagnosis'])
plt.title('Diagnosis (Malignant= M, Benign= B)')
plt.show()
```



The graph above illustrates the number of patients, and classify them based on their diagnosis. In the above graph Malignant is represented as 1.0 and Benign is represented as 0.0. We can see a drastic difference between the two stages, wherein clearly there are large number of patients diagnosed with Benign tumor as compared to the ones with Malignant

In [28]:
```
features_mean=list(breastdf2.columns[1:11])
# split dataframe into two based on diagnosis
dfM=breastdf2[breastdf2['diagnosis'] == 0]
dfB=breastdf2[breastdf2['diagnosis'] == 1]
```

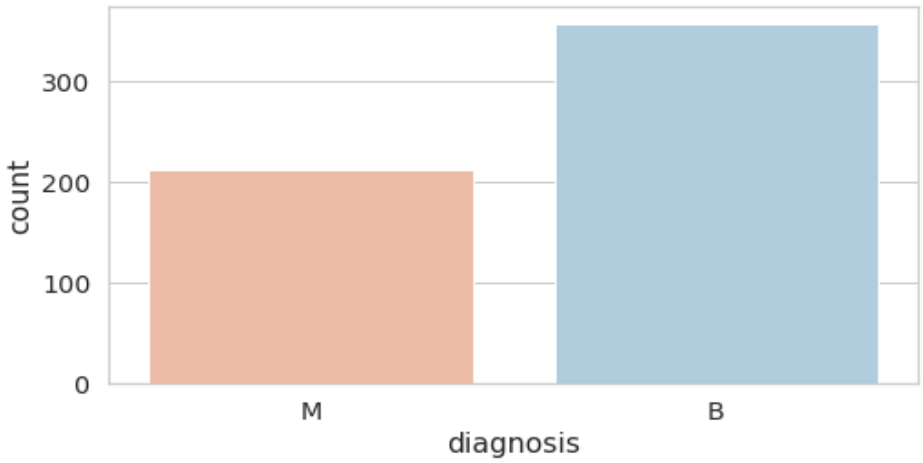
In [11]:
```
#Stack the data
plt.rcParams.update({'font.size': 10})
fig, axes = plt.subplots(nrows=5, ncols=2, figsize=(8,10))
axes = axes.ravel()
for idx,ax in enumerate(axes):
    ax.figure
    binwidth= (max(breastdf2[features_mean[idx]]) - min(breastdf2[features_mean[idx]]))/100
    ax.hist([dfM[features_mean[idx]],dfB[features_mean[idx]]], bins=np.arange(min(breastdf2[features_mean[idx]]), max(breastdf2
[features_mean[idx]]) + binwidth, binwidth) , alpha=0.5,stacked=True, density = True, label=['M','B'],color=['b','y'])
    ax.legend(loc='upper right')
    ax.set_title(features_mean[idx])
plt.tight_layout()
plt.show()
```



In the above figure, we have multiple histograms for the various attributes in the dataset. The histograms give us an idea as to how the different attributes of breast cells can help us understand if the cells are malignant or benign.

In [8]:
```
import pandas as pd
import numpy as np

# data visualization
import matplotlib.pyplot as plt
import seaborn as sns

# machine learning
from sklearn.preprocessing import StandardScaler

import sklearn.linear_model as skl_lm
from sklearn import preprocessing
from sklearn import neighbors
from sklearn.metrics import confusion_matrix, classification_report, precision_score
from sklearn.model_selection import train_test_split


import statsmodels.api as sm
import statsmodels.formula.api as smf


# initialize some package settings
sns.set(style="whitegrid", color_codes=True, font_scale=1.3)

%matplotlib inline

plt.figure(figsize=(8, 4))
sns.countplot(breastdf2['diagnosis'], palette='RdBu')

# count number of obvs in each class
benign, malignant = breastdf2['diagnosis'].value_counts()
print('Number of cells labeled Benign: ', benign)
print('Number of cells labeled Malignant : ', malignant)
print('')
print('% of cells labeled Benign', round(benign / len(breastdf2) * 100, 2), '%')
print('% of cells labeled Malignant', round(malignant / len(breastdf2) * 100, 2), '%')
```

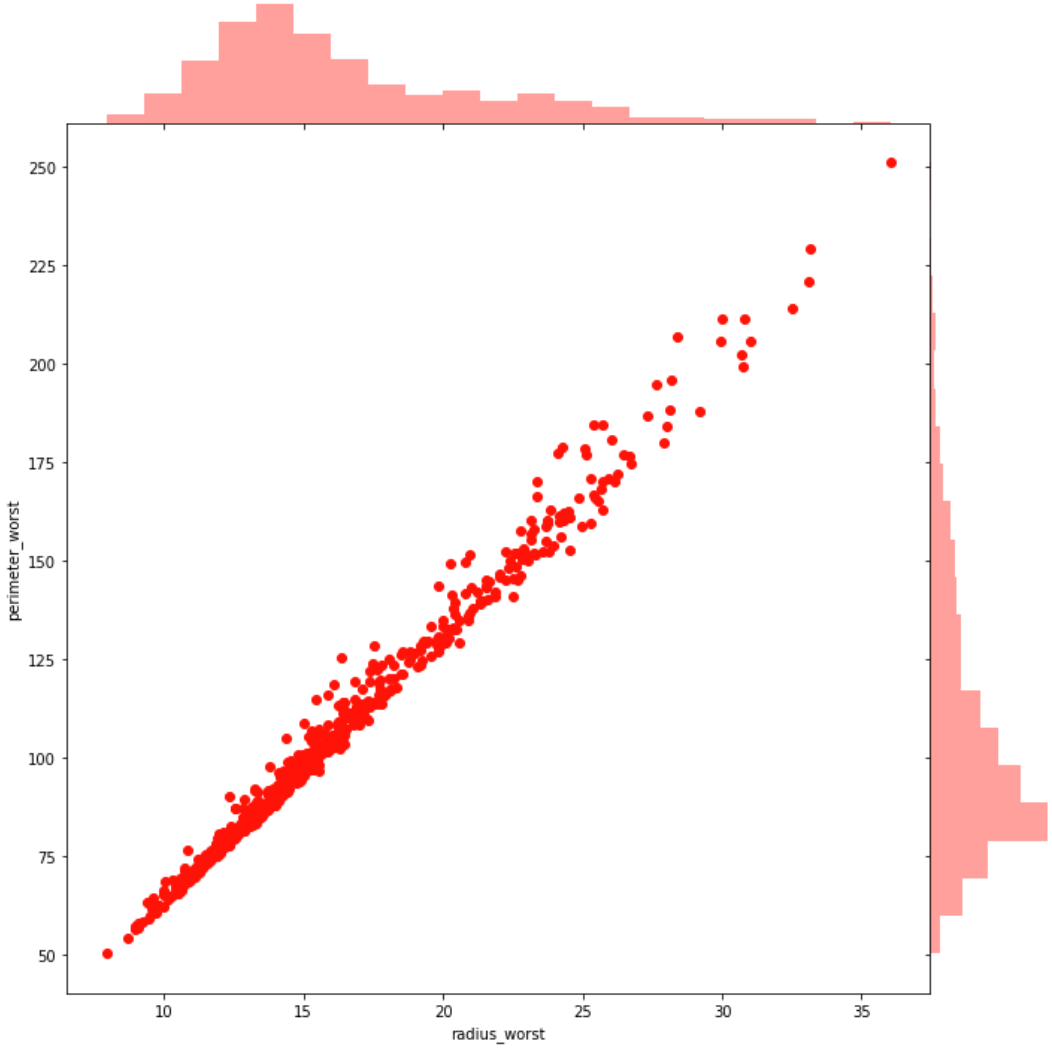
```
Number of cells labeled Benign:  357
Number of cells labeled Malignant :  212

% of cells labeled Benign 62.74 %
% of cells labeled Malignant 37.26 %
```



In the above diagram we can see the percentage values for the diagnosis of breast cells. The graph is called as count plot, as it provides us with the count of the desired attributes. The

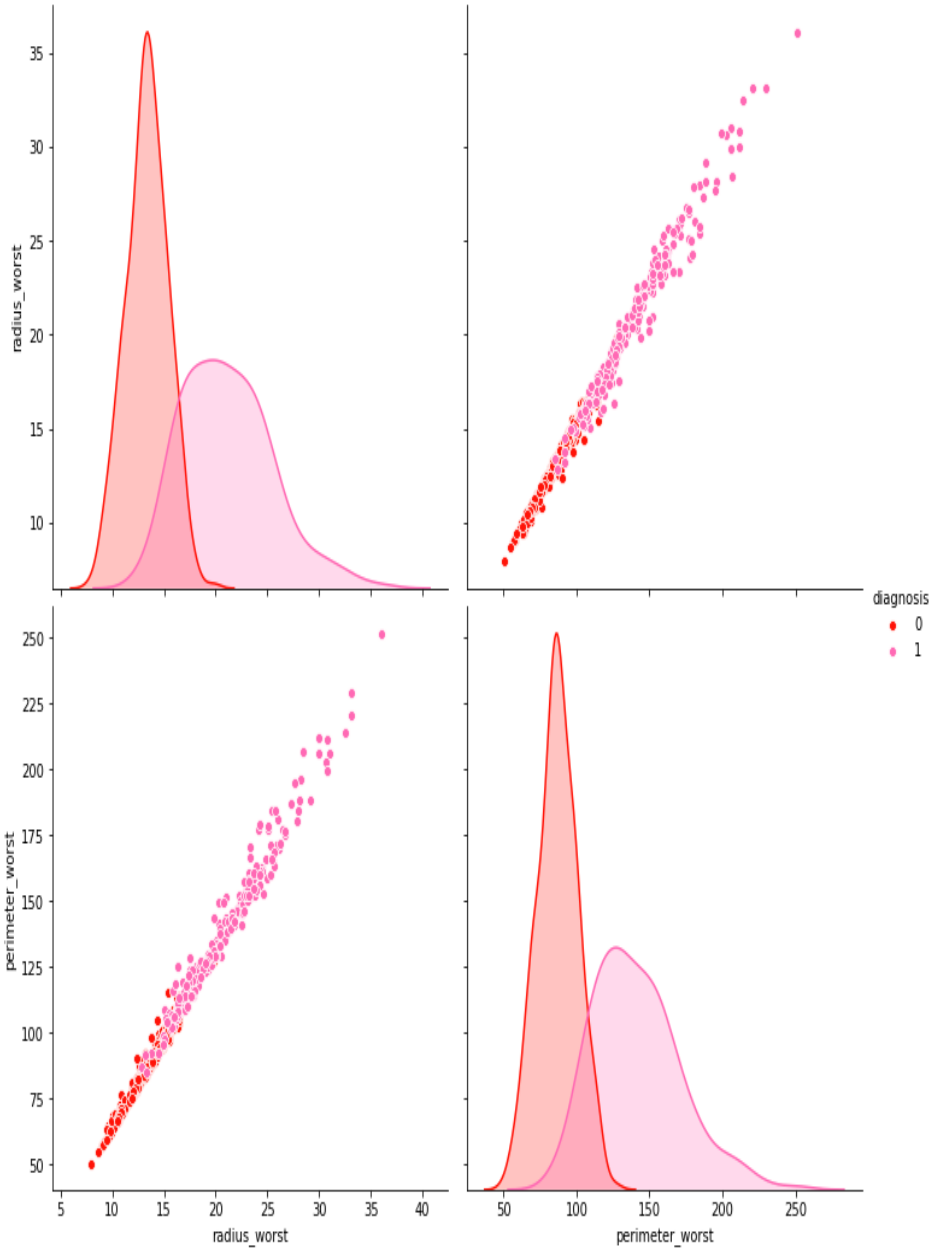
percentage value is obtained by using the percentage formula on the count obtained out of the total length.

```
n [19]: import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        from matplotlib.colors import ListedColormap

        sns.jointplot("radius_worst", "perimeter_worst", data=breastdf2, kind="scatter",
                      space=0, color="#FF1308", height=10, ratio=7)
        plt.show()
```



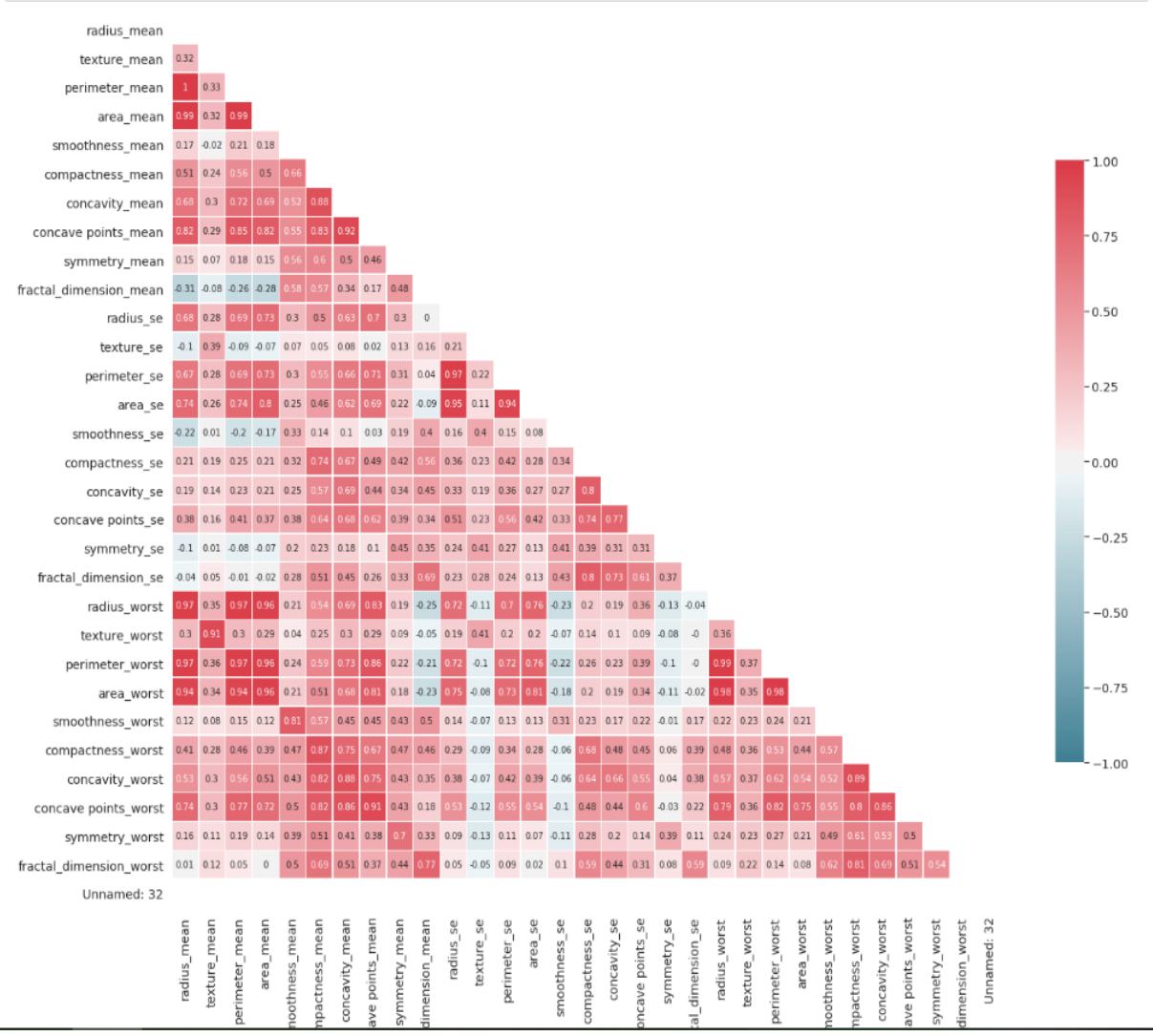
The above diagram is called as the join plots. Join plots are used to show the correlation between two attributes of datasets. Here we can see how perimeter_worst attribute is correlated to radius_worst attribute.

```
In [30]: sns.pairplot(breastdf2, vars=["radius_worst", "perimeter_worst"],
                      palette=sns.color_palette(['#FF1308', '#FF69B4']), hue='diagnosis', height=5)
         plt.show()
```



In, the graph above you can see pair plots. Now what exactly is a pair plot? Pair plots work similar to a scatter plot, pair plots help us to observe relationships between two entities or attributes. In pair plot above, we see a relationship between two columns ‘radius_worst’ and ‘perimeter _worst’.

In [12]:
```
# Generate and visualize the correlation matrix
corr = breastdf2.corr().round(2)

# Mask for the upper triangle
mask= np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set figure size
figure, ax = plt.subplots(figsize=(20, 20))

# Define custom colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap
sns.heatmap(corr, mask=mask, cmap=cmap, vmin=-1, vmax=1, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5}, annot=True)

plt.tight_layout()
```



Heatmaps:

What is a heatmap? (QuantInsti, 2020)

A heatmap is a two-dimensional graphical data representation where the individual values that are contained in a matrix are represented as colours. The seaborn python package allows

you to create annotated heatmaps which can be modified using Matplotlib tools as per the creator's requirement.

The above visualised shows the correlation between various attributes present int the dataset. Looking at the heatmap we can find that there is high multicollinearity between some of the attributes. Say for instance if we look at radius_se we can see that it has high collinearity with attributes perimeter_se and area_se. This is because of the fact that these three attributes contain similar data, that is they all talk about the size parameter of the breast cells. To avoid similar data, we can drop the columns which are subsets of a particular attribute and thus enabling us to focus on the main attributes only!

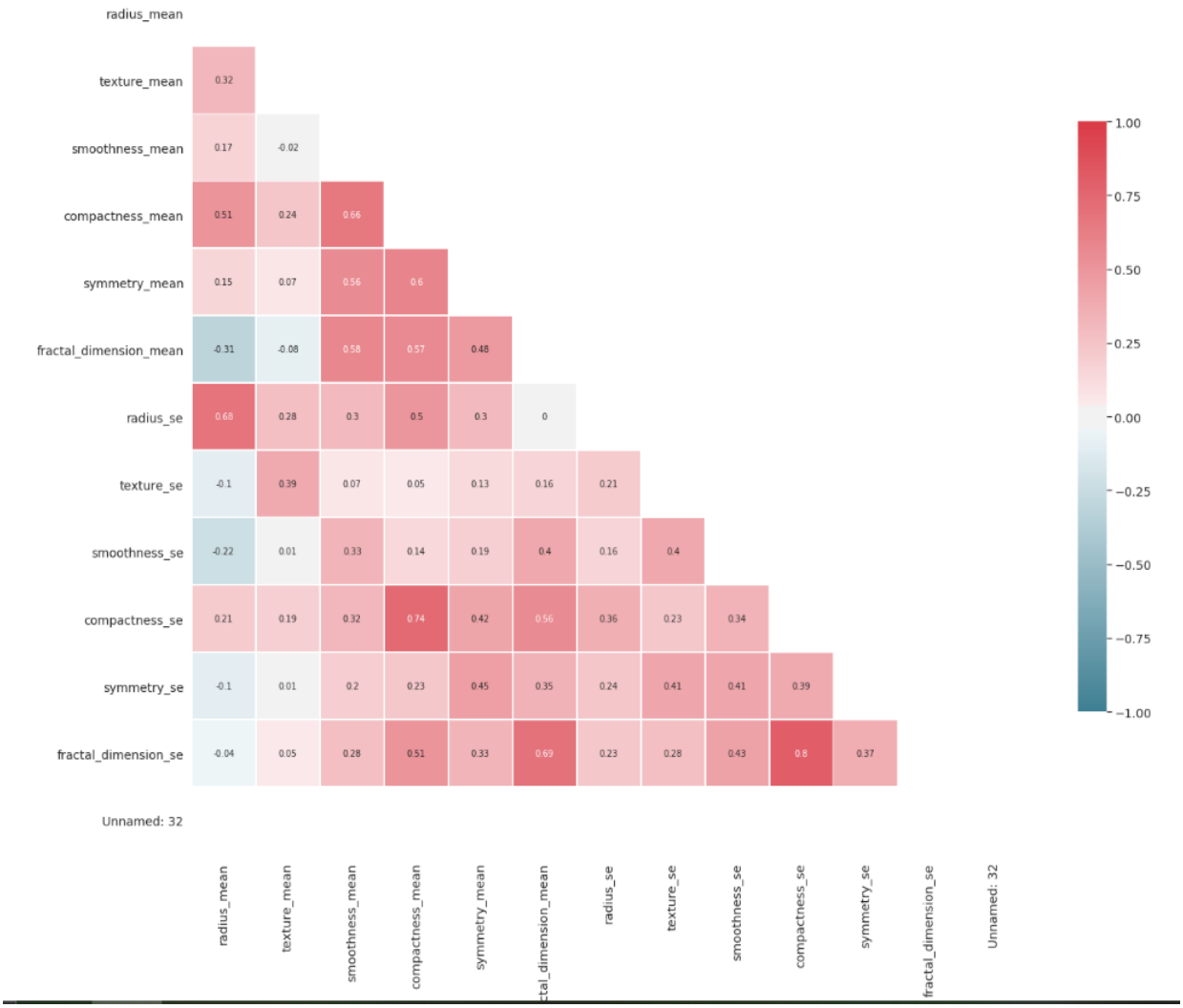
As you see in following images, I'm dropping all the similar columns to focus only on the mains. Since all the 'worst' columns speak about the area of the breast cells I have decided to delete them and concentrate only on the radius as using radius we can derive the other parameters easily.

In [14]:
```
# first, drop all "worst" columns
cols = ['radius_worst',
        'texture_worst',
        'perimeter_worst',
        'area_worst',
        'smoothness_worst',
        'compactness_worst',
        'concavity_worst',
        'concave points_worst',
        'symmetry_worst',
        'fractal_dimension_worst']
breastdf2 = breastdf2.drop(cols, axis=1)
```

In [15]:
```
# then, drop all columns related to the "perimeter" and "area" attributes
cols = ['perimeter_mean',
        'perimeter_se',
        'area_mean',
        'area_se']
breastdf2 = breastdf2.drop(cols, axis=1)
```

Rebuilding the heat map again as stated below with only distinct columns now,

In [20]:
```
# Draw the heatmap again, with the new correlation matrix
corr = breastdf2.corr().round(2)
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

f, ax = plt.subplots(figsize=(20, 20))
sns.heatmap(corr, mask=mask, cmap=cmap, vmin=-1, vmax=1, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5}, annot=True)
plt.tight_layout()
```



Now, we can observe that instead of having multiple unnecessary attributes we have filtered only the main attributes to establish better collinearity between them.

CONCLUSION:

To conclude, we have successfully classified breast cancer dataset using Logistic Regression and performed Exploratory data analysis on the datasets using different python libraries such as Pandas, Scikit-learn, matplotlib, seaborn etc on Pyspark using Jupyter notebook. The model works accurately and gives about 96.5% of classified data.

REFERENCES:

1. Cdc.gov. (2020). *What Is Breast Cancer?* | *CDC*. [online] Available at: https://www.cdc.gov/cancer/breast/basic_info/what-is-breast-cancer.htm [Accessed 27 Feb. 2020].

2. Google Docs. (2020). *breast_cancer_clinical_data.csv*. [online] Available at: https://drive.google.com/file/d/1yD1GWk2OWgOooq9WI1K7puoaKD7CbS8T/view [Accessed 27 Feb. 2020].

3. Wolberg, D., Nick Street, W. and L. Mangasarian, O. (2020). *UCI Machine Learning Repository: Breast Cancer Wisconsin (Diagnostic) Data Set*. [online] Archive.ics.uci.edu. Available at: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29 [Accessed 27 Feb. 2020].

4. AcadGild. (2020). *Data Analysis using Spark for Breast Cancer Data Analysis* | *Acadgild online*. [online] Available at: https://acadgild.com/blog/breast-cancer-data-analysis-using-spark [Accessed 27 Feb. 2020].

5. Dayananda, S. (2020). *Spark SQL Tutorial* | *Understanding Spark SQL With Examples* | *Edureka*. [online] Edureka. Available at: https://www.edureka.co/blog/spark-sql-tutorial [Accessed 27 Feb. 2020].

6. *Welcome to Py4J — Py4J*. [online] Available at: https://www.py4j.org/ [Accessed 27 Feb. 2020].

7. Pandas.pydata.org. (2020). *pandas - Python Data Analysis Library*. [online] Available at: https://pandas.pydata.org [Accessed 27 Feb. 2020].

8. Numpy.org. (2020). *NumPy — NumPy*. [online] Available at: https://numpy.org/ [Accessed 27 Feb. 2020].

9. Seaborn.pydata.org. (2020). *seaborn: statistical data visualization — seaborn 0.10.0 documentation*. [online] Available at: https://seaborn.pydata.org [Accessed 27 Feb. 2020].

10. Matplotlib.org. (2020). *Matplotlib: Python plotting — Matplotlib 3.1.3 documentation*. [online] Available at: https://matplotlib.org/ [Accessed 27 Feb. 2020].

11. En.wikipedia.org. (2020). *Scikit-learn*. [online] Available at: https://en.wikipedia.org/wiki/Scikit-learn [Accessed 27 Feb. 2020].

12. Mpld3.github.io. (2020). *mpld3 — Bringing Matplotlib to the Browser*. [online] Available at: https://mpld3.github.io [Accessed 27 Feb. 2020].

13. Tutorialspoint.com. (2020). *PySpark Tutorial - Tutorialspoint*. [online] Available at: https://www.tutorialspoint.com/pyspark/index.htm [Accessed 27 Feb. 2020].

14. Pant, A. (2020). *Introduction to Logistic Regression*. [online] Medium. Available at: https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148 [Accessed 27 Feb. 2020].

15. Kaggle.com. (2020). *Breast cancer prediction*. [online] Available at: https://www.kaggle.com/buddhiniw/breast-cancer-prediction [Accessed 27 Feb. 2020].

16. Kaggle.com. (2020). *Predicting Breast Cancer - Logistic Regression*. [online] Available at: https://www.kaggle.com/jagannathrk/predicting-breast-cancer-logistic-regression [Accessed 27 Feb. 2020].

17. Wingate, J. (2020). *Breast Cancer Dataset Analysis, Visualization and Machine Learning in Python*. [online] Engineeringbigdata.com. Available at: https://www.engineeringbigdata.com/breast-cancer-dataset-analysis-visualization-and-machine-learning-in-python [Accessed 27 Feb. 2020].

18. Paradkar, M. (2020). *Using Seaborn Python Package for Creating Heatmap*. [online] QuantInsti. Available at: https://blog.quantinsti.com/creating-heatmap-using-python-seaborn/ [Accessed 27 Feb. 2020].

19. Seaborn.pydata.org. (2020). *seaborn: statistical data visualization — seaborn 0.10.0 documentation*. [online] Available at: https://seaborn.pydata.org/ [Accessed 27 Feb. 2020].

20. Nair, S. (2020). *Breast Cancer Classification*. [online] Drive.google.com. Available at: https://drive.google.com/uc?export=download&id=1gqmSyCtMGYC-9kjc-wUNdn64QTOX3lEY [Accessed 27 Feb. 2020].