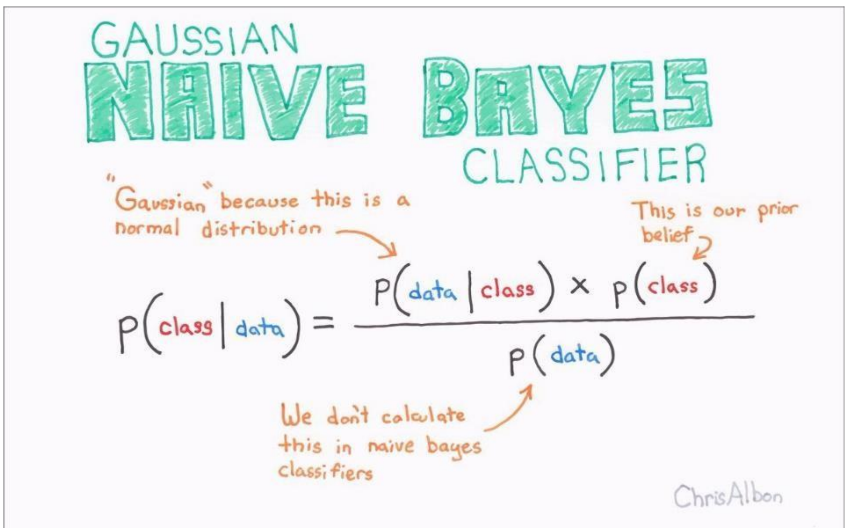
Experiment No. 5

Experiment Title: Naive Bayes

Student Name: YANA SRIVASTAVA **UID:** 20BCS2279
Branch: CSE **Section/Group:** 20BCS-WM-906/B
Semester: 5th **Date of Performance:** 21/10/22
Subject Name: Machine Learning Lab **Subject Code:** 21CSP-317



1. Aim/Overview of the practical:

Implementation of Naïve Bayes Algorithm.

2. Steps of Experiment:

- Import all the required library.
- Import the dataset which you want to implement.
- Split data into x and y and perform some task.
- Split data into training set and testing set.
- Apply Naïve Bayes formula.

3. Source Code/Result/Output:

jupyter Naive Bays Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("darkgrid")
```

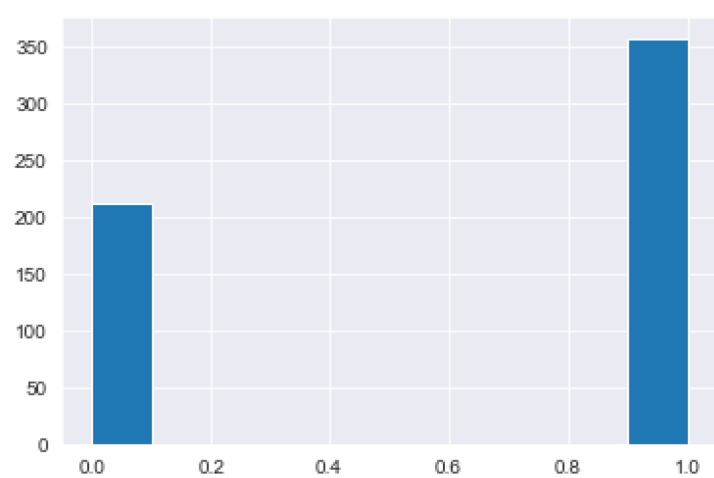
In [18]:
```
data = pd.read_csv(r"C:\Users\G.K.Computer Service\Downloads\Breast_cancer_data.csv")
data.head()
```

Out[18]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
0	17.99	10.38	122.80	1001.0	0.11840	0
1	20.57	17.77	132.90	1326.0	0.08474	0
2	19.69	21.25	130.00	1203.0	0.10960	0
3	11.42	20.38	77.58	386.1	0.14250	0
4	20.29	14.34	135.10	1297.0	0.10030	0

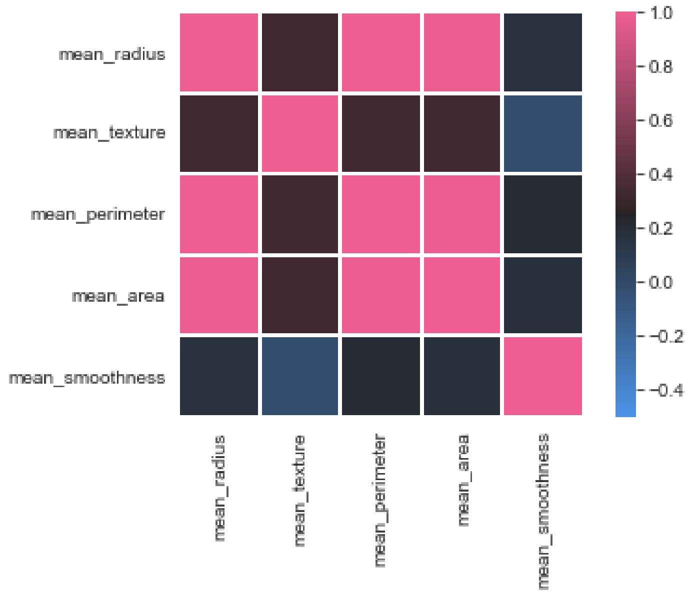
In [6]:
```
data["diagnosis"].hist()
```

Out[6]: <AxesSubplot:>



In [7]:
```
corr = data.iloc[:,:-1].corr(method="pearson")
cmap = sns.diverging_palette(250,354,80,60,center='dark',as_cmap=True)
sns.heatmap(corr, vmax=1, vmin=-.5, cmap=cmap, square=True, linewidths=.2)
```

Out[7]: <AxesSubplot:>



In [19]:
```
data = data[["mean_radius", "mean_texture", "mean_smoothness", "diagnosis"]]
data.head(5)
```

Out[19]:

	mean_radius	mean_texture	mean_smoothness	diagnosis
0	17.99	10.38	0.11840	0
1	20.57	17.77	0.08474	0
2	19.69	21.25	0.10960	0
3	11.42	20.38	0.14250	0
4	20.29	14.34	0.10030	0

In [10]:
```
def calculate_prior(df, Y):
    classes = sorted(list(df[Y].unique()))
    prior = []
    for i in classes:
        prior.append(len(df[df[Y]==i])/len(df))
    return prior
```

In [11]:
```
def calculate_likelihood_gaussian(df, feat_name, feat_val, Y, label):
    feat = list(df.columns)
    df = df[df[Y]==label]
    mean, std = df[feat_name].mean(), df[feat_name].std()
    p_x_given_y = (1 / (np.sqrt(2 * np.pi) * std)) *  np.exp(-((feat_val-mean)**2 / (2 * std**2 )))
    return p_x_given_y
```

In [12]:
```
def naive_bayes_gaussian(df, X, Y):
    # get feature names
    features = list(df.columns)[:-1]

    # calculate prior
    prior = calculate_prior(df, Y)

    Y_pred = []
    # loop over every data sample
    for x in X:
        # calculate likelihood
        labels = sorted(list(df[Y].unique()))
        likelihood = [1]*len(labels)
        for j in range(len(labels)):
            for i in range(len(features)):
                likelihood[j] *= calculate_likelihood_gaussian(df, features[i], x[i], Y, labels[j])

        # calculate posterior probability (numerator only)
        post_prob = [1]*len(labels)
        for j in range(len(labels)):
            post_prob[j] = likelihood[j] * prior[j]

        Y_pred.append(np.argmax(post_prob))

    return np.array(Y_pred)
```

In [13]:
```
from sklearn.model_selection import train_test_split
train, test = train_test_split(data, test_size=.2, random_state=41)

X_test = test.iloc[:,:-1].values
Y_test = test.iloc[:,-1].values
Y_pred = naive_bayes_gaussian(train, X=X_test, Y="diagnosis")

from sklearn.metrics import confusion_matrix, f1_score
print(confusion_matrix(Y_test, Y_pred))
print(f1_score(Y_test, Y_pred))
```

```
[[36  4]
 [ 0 74]]
0.9736842105263158
```

In [14]:
```
data["cat_mean_radius"] = pd.cut(data["mean_radius"].values, bins = 3, labels = [0,1,2])
data["cat_mean_texture"] = pd.cut(data["mean_texture"].values, bins = 3, labels = [0,1,2])
data["cat_mean_smoothness"] = pd.cut(data["mean_smoothness"].values, bins = 3, labels = [0,1,2])

data = data.drop(columns=["mean_radius", "mean_texture", "mean_smoothness"])
data = data[["cat_mean_radius", "cat_mean_texture", "cat_mean_smoothness", "diagnosis"]]
data.head(10)
```

Out[14]:

	cat_mean_radius	cat_mean_texture	cat_mean_smoothness	diagnosis
0	1	0	1	0
1	1	0	0	0
2	1	1	1	0
3	0	1	2	0
4	1	0	1	0
5	0	0	2	0
6	1	1	1	0
7	0	1	1	0
8	0	1	2	0
9	0	1	1	0

[15]:

```
def calculate_likelihood_categorical(df, feat_name, feat_val, Y, label):
    feat = list(df.columns)
    df = df[df[Y]==label]
    p_x_given_y = len(df[df[feat_name]==feat_val]) / len(df)
    return p_x_given_y
```

In [16]:

```
def naive_bayes_categorical(df, X, Y):
    # get feature names
    features = list(df.columns)[:-1]

    # calculate prior
    prior = calculate_prior(df, Y)

    Y_pred = []
    # loop over every data sample
    for x in X:
        # calculate likelihood
        labels = sorted(list(df[Y].unique()))
        likelihood = [1]*len(labels)
        for j in range(len(labels)):
            for i in range(len(features)):
                likelihood[j] *= calculate_likelihood_categorical(df, features[i], x[i], Y, labels[j])

        # calculate posterior probability (numerator only)
        post_prob = [1]*len(labels)
        for j in range(len(labels)):
            post_prob[j] = likelihood[j] * prior[j]

        Y_pred.append(np.argmax(post_prob))

    return np.array(Y_pred)
```

In [17]:

```
from sklearn.model_selection import train_test_split
train, test = train_test_split(data, test_size=.2, random_state=41)

X_test = test.iloc[:,:-1].values
Y_test = test.iloc[:,-1].values
Y_pred = naive_bayes_categorical(train, X=X_test, Y="diagnosis")

from sklearn.metrics import confusion_matrix, f1_score
print(confusion_matrix(Y_test, Y_pred))
print(f1_score(Y_test, Y_pred))
```

```
[[38  2]
 [ 5 69]]
0.9517241379310345
```

Learning outcomes (What I have learnt):

1. **Learnt to analyze the data.**
2. **Learnt to import various libraries.**
3. **Learnt to read csv files.**
4. **Learnt to implement Logistic Regression.**
5. **Learnt to train and test the data.**
6. **Learnt the concept of SVM (Support Vector Machine).**

Evaluation Grid:

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.	Student Performance (Conduct of experiment) objectives/Outcomes.		12
2.	Viva Voce		10
3.	Submission of Work Sheet (Record)		8
	Total		30