Data collection

We searched for a dataset to use in our model and our project and we found many datasets but most of the were small datasets and will be bad in learning so we decided to use " Breast Cancer Wisconsin (Diagnostic) Data Set "

Because it was big enough to be learnt by the machine learning models.

In [ ]:

In [4]:
```
print (dataset.shape)
```

```
(569, 33)
```

In [ ]:

In [5]:
```
print (dataset.head(5)) # here we see that all features of data is int unless the diagnosis features
```

```
         id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0    842302         M        17.99         10.38          122.80     1001.0
1    842517         M        20.57         17.77          132.90     1326.0
2  84300903         M        19.69         21.25          130.00     1203.0
3  84348301         M        11.42         20.38           77.58      386.1
4  84358402         M        20.29         14.34          135.10     1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520
4          0.10030           0.13280          0.1980              0.10430

   ...  texture_worst  perimeter_worst  area_worst  smoothness_worst  \
0  ...          17.33           184.60      2019.0            0.1622
1  ...          23.41           158.80      1956.0            0.1238
2  ...          25.53           152.50      1709.0            0.1444
3  ...          26.50            98.87       567.7            0.2098
4  ...          16.67           152.20      1575.0            0.1374

   compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \
0             0.6656           0.7119                0.2654          0.4601
1             0.1866           0.2416                0.1860          0.2750
2             0.4245           0.4504                0.2430          0.3613
3             0.8663           0.6869                0.2575          0.6638
4             0.2050           0.4000                0.1625          0.2364

   fractal_dimension_worst  Unnamed: 32
0                  0.11890          NaN
1                  0.08902          NaN
2                  0.08758          NaN
3                  0.17300          NaN
4                  0.07678          NaN

[5 rows x 33 columns]
```

In [ ]:

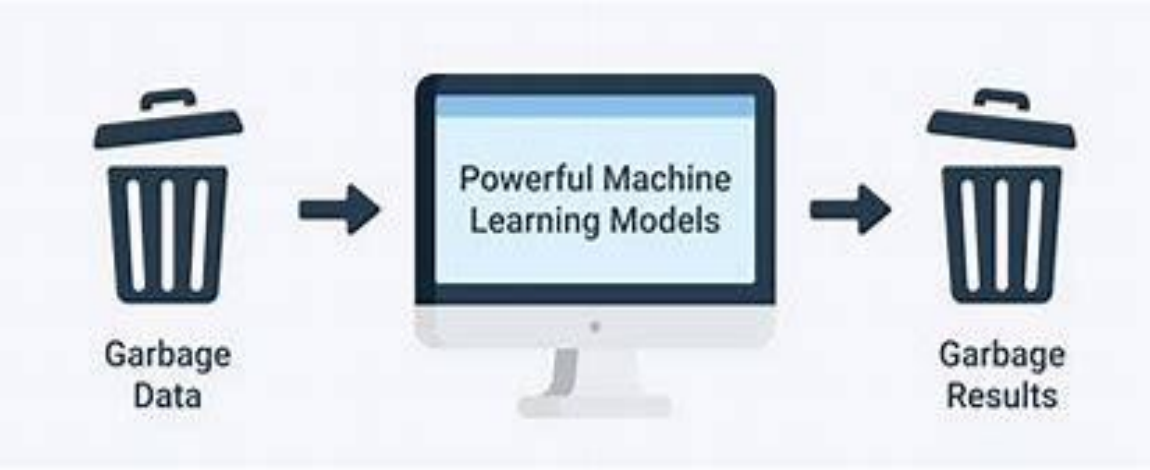
***The dataset has 569 row (attribute) and 33 columns (features).**

Data preprocessing

Data preprocessing is a step in the data mining and data analysis process that takes raw data and transforms it into a format that can be understood and analyzed by computers and machine learning.

Raw, real-world data in the form of text, images, video, etc., is messy. Not only may it contain errors and inconsistencies, but it is often incomplete, and doesn't have a regular, uniform design.

Machines like to process nice and tidy information – they read data as 1s and 0s. So calculating structured data, like whole numbers and percentages is easy. However, unstructured data, in the form of text and images must first be cleaned and formatted before analysis.



So to prevent this problems we so some data preprocessing techniques in our project to build a good model:-

- First we search for a missing values to handle it :-

In [40]:
```
# Assuming your data is stored in a DataFrame called 'dataset'

# Check for missing values in the entire DataFrame
missing_values = dataset.isnull().sum()
print(missing_values)

# Check for missing values in a specific column
missing_values_in_column = dataset['Column_Name'].isnull().sum()
print(missing_values_in_column)
```

```
id                         0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
diagnosis                  0
dtype: int64
```

- Here above we see that the dataset was clean and don’t have any missing values so it's ready to use in the model.

- Text data requires special preparation before you can start using it for predictive modeling.

- So after that we prepare the data by encoding some features to be easy for the model to use , we have different encoders to use like (one_hot_encoder) and (label_encoder) , here we used the 'label_encoder' :-

In [7]:
```
# Importing LabelEncoder from Sklearn
# library from preprocessing Module.
from sklearn.preprocessing import LabelEncoder

# Creating a instance of label Encoder.
le = LabelEncoder()

# Using .fit_transform function to fit label
# encoder and return encoded label
label = le.fit_transform(dataset['diagnosis'])

# printing label
label
```

Out[7]:
```
array([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, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 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, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 0, 1, 0, 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, 1, 0, 0, 1, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 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, 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, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 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, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 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, 1, 0, 1, 1, 0, 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, 0, 1, 1, 1, 1, 1, 1, 0])
```

In [ ]:

- As we see above we use label encoder to convert all text data which was in the "diagnosis" feature (the label) to a numeric data zeros and ones ; one for "m" Malignant and zero for "b" Benign .

- After that we will remove some data which we don’t need and it has no effect on the label or medel learning (there is no relation between this feature and the other features) example below :-

```
In [12]: dataset.iloc[:,[0]] # not used in training or testing
```

Out[12]:

	id
0	842302
1	842517
2	84300903
3	84348301
4	84358402
...	...
564	926424
565	926682
566	926954
567	927241
568	92751

569 rows × 1 columns

Features selection

Feature selection is also called variable selection or attribute selection.

It is the automatic selection of attributes in your data (such as columns in tabular data) that are most relevant to the predictive modeling problem you are working on.

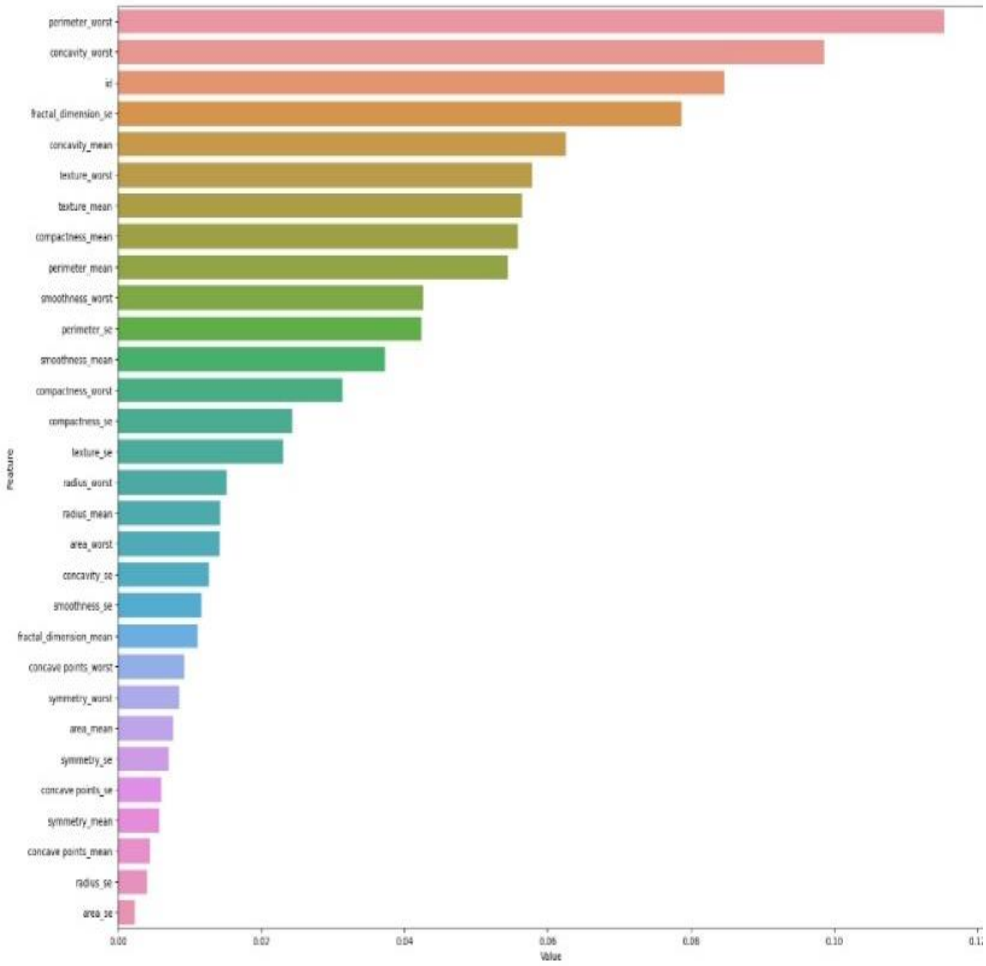
feature selection… is the process of selecting a subset of relevant features for use in model construction

Feature selection is different from dimensionality reduction. Both methods seek to reduce the number of attributes in the dataset, but a dimensionality reduction method do so by creating new combinations of attributes, where as feature selection methods include and exclude attributes present in the data without changing them.

Examples of dimensionality reduction methods include Principal Component Analysis, Singular Value Decomposition and Sammon's Mapping.

Finally : ***The objective of variable selection is three-fold: improving the prediction performance of the predictors, providing faster and more cost-effective predictors, and providing a better understanding of the underlying process that generated the data.***

```
In [42]: feature_imp = pd.DataFrame(sorted(zip(rf_model.feature_importances_,X.columns)), columns=['Value','Feature'])

         plt.figure(figsize=(20, 15))
         sns.barplot(x="Value", y="Feature", data=feature_imp.sort_values(by="Value", ascending=False))

Out[42]: <Axes: xlabel='Value', ylabel='Feature'>
```



- Here we see the features importance graph ,

- In our project we decided to use only 10 features from the all features of the dataset however the data importance, the features we decided to use are :-

1- The "radius_mean" is one of the features in breast cancer datasets that represents the mean of distances from the center to points on the perimeter of the tumor.

2- The "texture_mean" is another feature in breast cancer datasets that represents the mean value of the gray-scale intensities of the pixels in the image of the tumor.

3- The "perimeter_mean" is another feature commonly found in breast cancer datasets. It represents the mean value of the perimeter (the distance around the tumor) of the breast mass.

4- The "area_mean" is another common feature in breast cancer datasets. It represents the mean value of the area (the total number of pixels) of the tumor mass.

5- The "smoothness_mean" is another feature commonly found in breast cancer datasets. It represents the mean variation in the local smoothness of the tumor boundaries.

6- The "compactness_mean" is another feature commonly present in breast cancer datasets. It

represents the mean value of the compactness of the tumor, which is calculated as the perimeter squared divided by the area.

7- The "concavity_mean" is another feature commonly found in breast cancer datasets. It represents the mean severity of concave portions of the contour of the tumor.

8- The "concave points_mean" is another feature commonly present in breast cancer datasets. It represents the mean number of concave portions of the contour that are formed by the tumor.

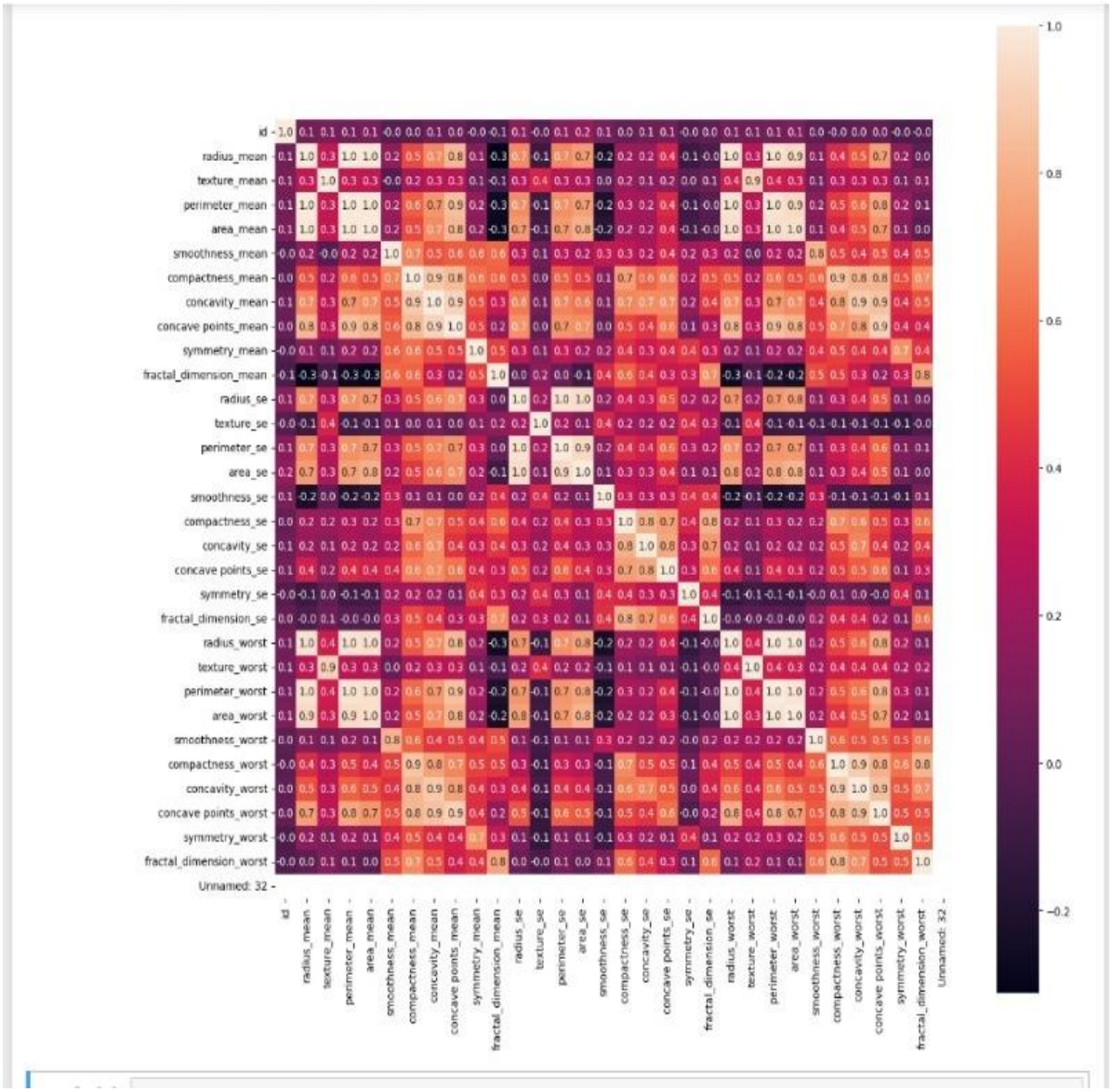
9- The "symmetry_mean" is another feature commonly found in breast cancer datasets. It represents the mean value of the symmetry of the tumor, which measures how symmetric the contour of the tumor is.

10- The "fractal_dimension_mean" is another feature commonly present in breast cancer datasets. It represents the mean value of the fractal dimension of the tumor, which characterizes the complexity of the tumor boundary.

- This are all features we will build our project on and we will learn them to the machine learning model and this are the features which we will ask the patient or the doctor to enter to the applicacion in the final stages to classify the tumer type.

Data visualization

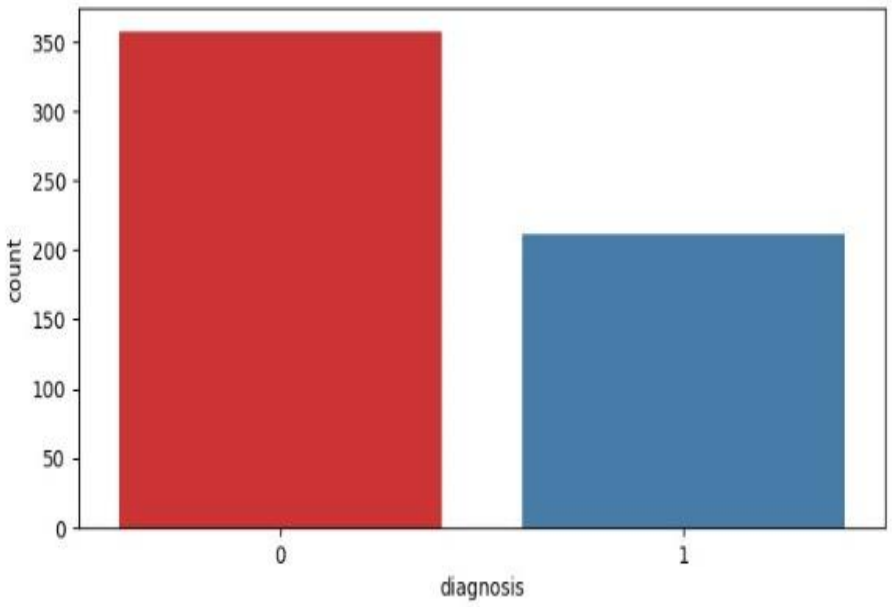
Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. Additionally, it provides an excellent way for employees or business owners to present data to non-technical audiences without confusion.



- This was a heat_map for the whole data (before feature selection) above .

In [28]:
```
plt.figure(figsize=(8, 4))
sns.countplot(x=dataset['diagnosis'], palette='Set1')

# Count number of observations in each class
benign, malignant = dataset['diagnosis'].value_counts()
print('No. Benign: ', benign)
print('No. Malignant : ', malignant)
print('')
print('% of cells labeled Benign', round(benign / len(dataset) * 100, 2), '%')
print('% of cells labeled Malignant', round(malignant / len(dataset) * 100, 2), '%')
```

```
No. Benign:  357
No. Malignant :  212

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



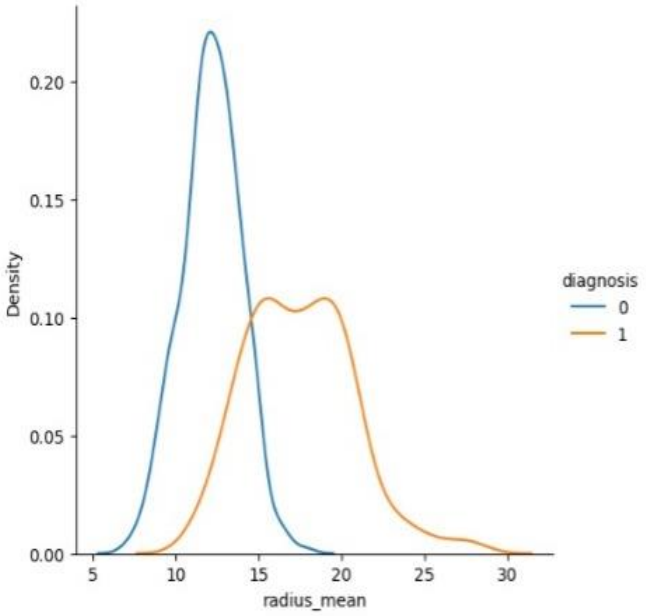
- here we can see the amount of of malignant and also the amount of benign and the percentage of them .

- Now we will do some graphs for the selected features in order :-

1- radius_mean (very effective) :

In [ ]:

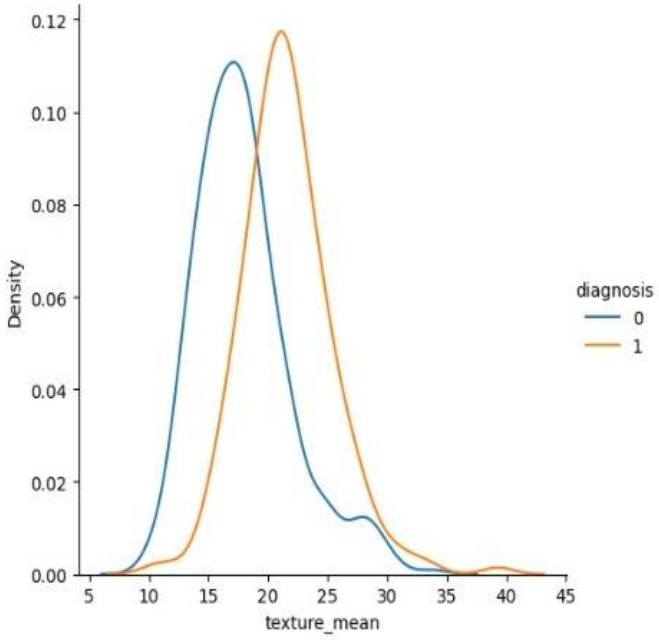
In [28]:
```
sns.FacetGrid(dataset,hue='diagnosis',height=5).map(sns.kdeplot,"radius_mean").add_legend()
plt.show()
```



In [ ]:

2- texture_mean :

In [29]:
```
sns.FacetGrid(dataset,hue='diagnosis',height=5).map(sns.kdeplot,"texture_mean").add_legend()
plt.show()
```

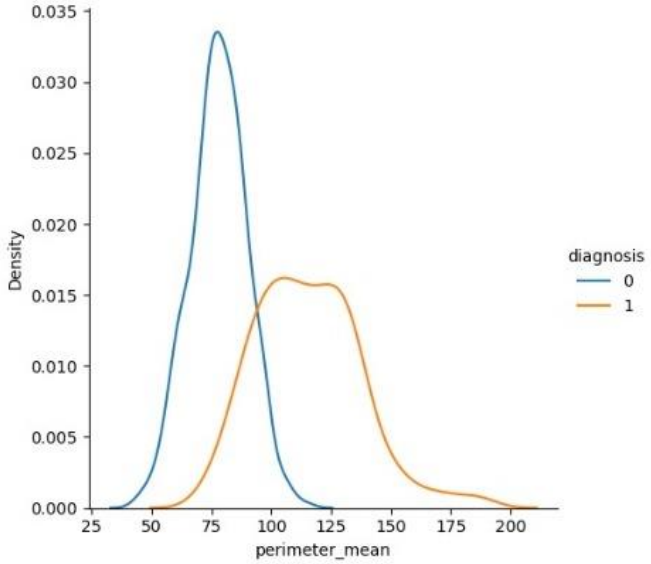


In [ ]:

3- perimeter_mean (very effective

```
In [ ]:
```

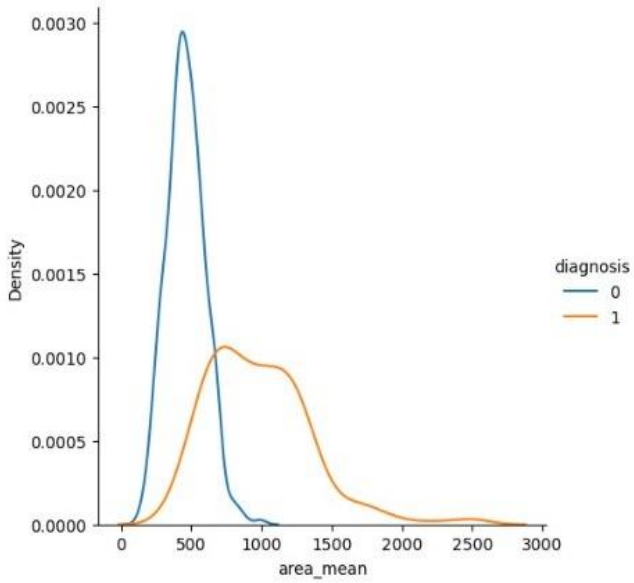
```
In [30]: sns.FacetGrid(dataset,hue='diagnosis',height=5).map(sns.kdeplot,"perimeter_mean").add_legend()
         plt.show()
```



4- area_mean (very effective) :

```
In [ ]:
```

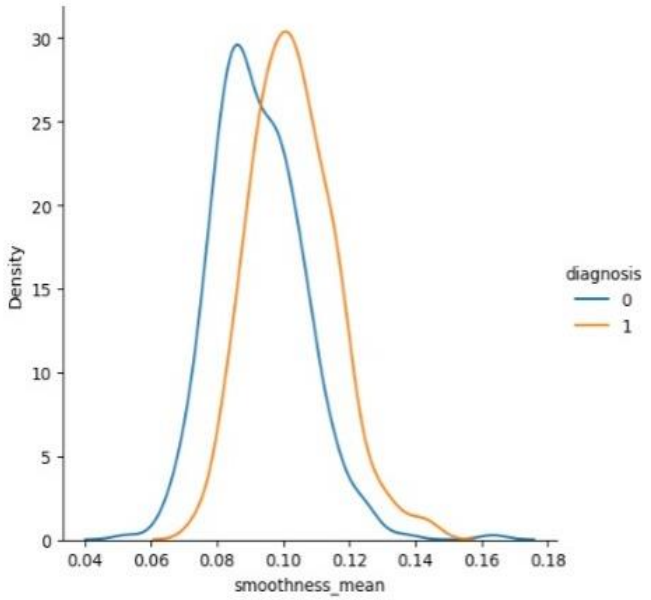
```
In [31]: sns.FacetGrid(dataset,hue='diagnosis',height=5).map(sns.kdeplot,"area_mean").add_legend()
         plt.show()
```



5- smoothness_mean :

```
In [ ]:
```

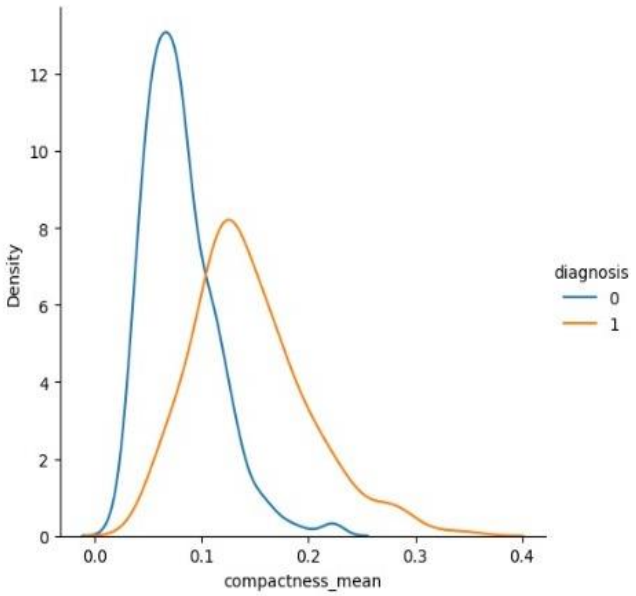
```
In [32]: sns.FacetGrid(dataset,hue='diagnosis',height=5).map(sns.kdeplot,"smoothness_mean").add_legend()
         plt.show()
```




```
In [ ]:
```

6- compactness_mean (very effective) :

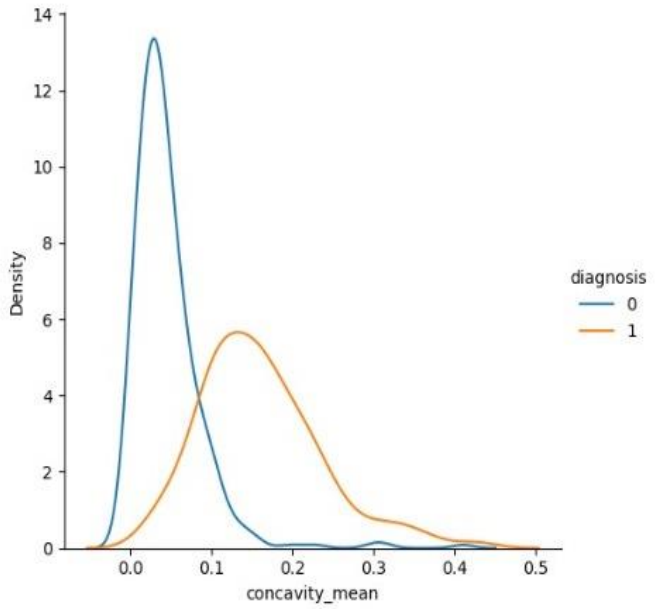
```
In [33]: sns.FacetGrid(dataset,hue='diagnosis',height=5).map(sns.kdeplot,"compactness_mean").add_legend()
         plt.show()
```




```
In [ ]:
```

7- concavity_mean (very effective) :

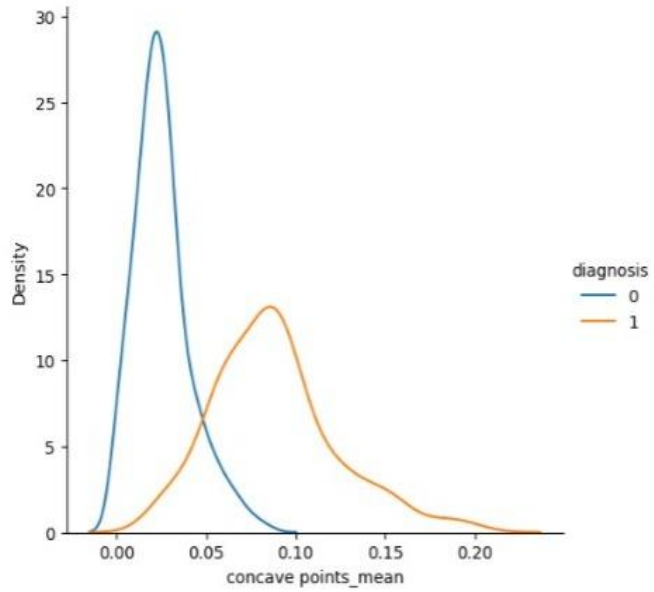
```
In [ ]:

In [35]: sns.FacetGrid(dataset,hue='diagnosis',height=5).map(sns.kdeplot,"concavity_mean").add_legend()
         plt.show()
```




```
In [ ]:
```

8- concave points_mean (very effective) :

```
In [ ]:

In [36]: sns.FacetGrid(dataset,hue='diagnosis',height=5).map(sns.kdeplot,"concave points_mean").add_legend()
         plt.show()
```

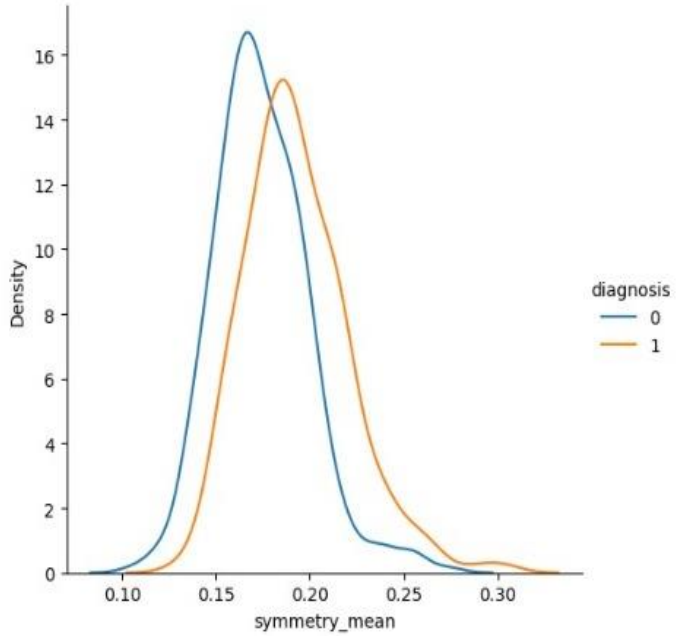



```
In [ ]:
```

9- symmetry_mean :

In [ ]:

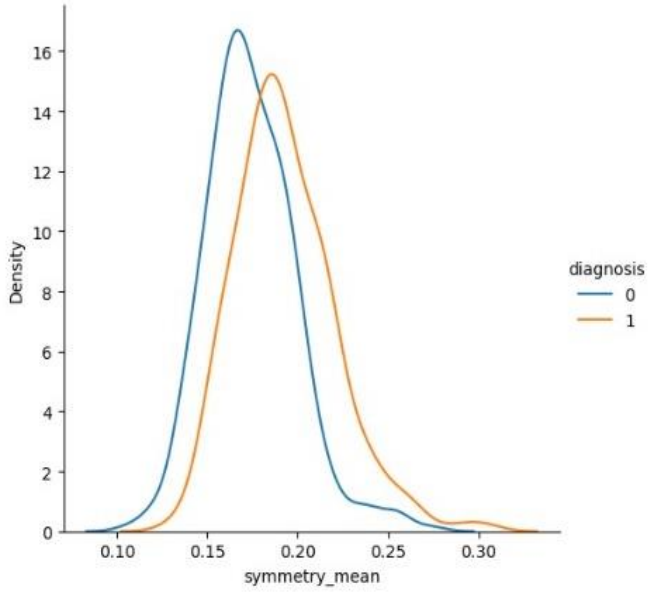
In [37]:
```
sns.FacetGrid(dataset,hue='diagnosis',height=5).map(sns.kdeplot,"symmetry_mean").add_legend()
plt.show()
```



In [ ]:

10- fractal_dimension_mean :

In [37]:
```
sns.FacetGrid(dataset,hue='diagnosis',height=5).map(sns.kdeplot,"fractal_dimension_mean").add_legend()
plt.show()
```



Machine learning model and the classifier

Machine learning classifiers are algorithms used to classify or predict the target variable based on input features.

We searched for many algoritms to use in our project and we try many classifier many times to select the best one to use it in the model .

Types of classifier :-

1- Decision Trees: A non-parametric classifier that makes decisions by recursively partitioning the input space based on feature values. Decision trees can handle both classification and regression tasks.

2- Random Forest: An ensemble classifier that combines multiple decision trees. It reduces overfitting and improves generalization by averaging the predictions of individual trees.

3- Logistic Regression: A linear classifier used for binary classification problems. It models the probability of the input belonging to a particular class using a logistic function.

4- Support Vector Machines (SVM): A classifier that finds an optimal hyperplane in a high-dimensional space to separate different classes. SVMs can handle both linear and non-linear classification problems.

5- K-Nearest Neighbors (KNN): A lazy learning classifier that assigns labels to new instances based on the majority vote of their k nearest neighbors in the training set.

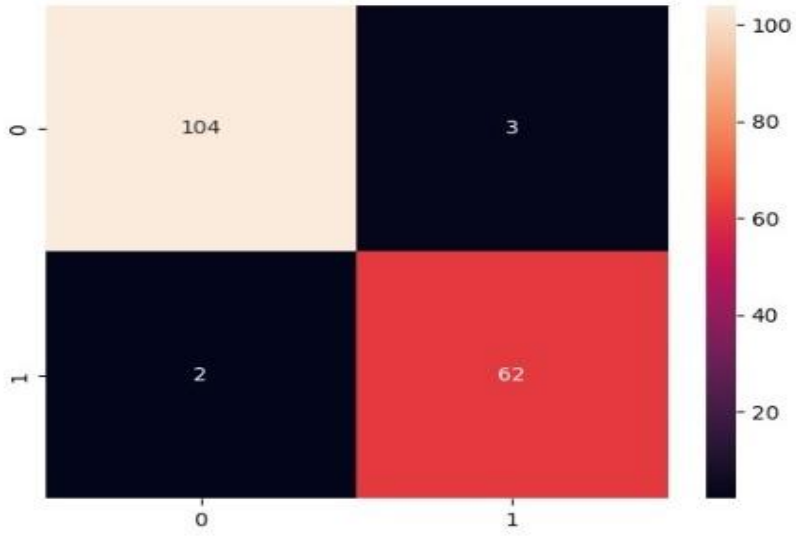
- And there are some other types of classifiers like : Gradient Boosting Machines (GBM), Deep Learning Classifiers , Neural Networks , Ensemble Methods: These methods combine multiple classifiers to improve performance .

*** In our project we decided to use the random forest algorithm (classifier) because it was suitable for the data set and achieve a good performance with a good acuuracy and good with some other measures .**

In [23]:
```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, classification_report, confusion_matrix

f1 = f1_score(y_test, y_pred)

prec = precision_score(y_test, y_pred)

recall = recall_score(y_test, y_pred)

Accuracy = metrics.accuracy_score(y_pred , y_test)

# Model Accuracy: how often is the classifier correct?
print('Accuracy  :',Accuracy)
print ('precision :',prec )
print ('recall    :',recall )
print ('f1_score  :',f1 )

end_time = datetime.now()
print('Duration  : {}'.format(end_time - start_time))
```

```
Accuracy  : 0.9824561403508771
precision : 0.9830508474576272
recall    : 0.9666666666666667
f1_score  : 0.9747899159663865
Duration  : 0:00:00.663512
```

```
In [21]: #confusion matrix

         from sklearn.metrics import confusion_matrix

         conf = confusion_matrix(y_pred,y_test)

         print('confusion matrix',conf)

         cm_2 = conf
         sns.heatmap(cm_2,annot=True,fmt="d")

         confusion matrix [[104   3]
          [  2  62]]

Out[21]: <Axes: >
```



- Here we can see the measures and the confusion matrix to know the performance of the classifier (algorithm) which was "random_forest" .

- we also did a comparison between some classifier:-

Algorithm	Accuracy	precision	recall	F1_Score	duration
knn	0.909	0.925	0.917	0.941	0:00:00.2 8315
svm	0.9705	0.96345	0.9588	0.96015	0:00:00.6 365
Random f	0.9726	0.9737	0.9532	0.9612	0:00:00.6 3775
logistic r	0.9426	0.95295	0.8958	0.92215	0:00:00.9 1825
decision t	0.93455	0.92795	0.89755	0.17735	0:00:00.6 794

- in this table we calculate the mean of Accuracy ,precision ,recall ,F1_Score ,duration for algorithms.

Saving the model

-After we set the classifier and trining it and evaluation we need to save the model ; so we used the "joblib" library to doing that.

Joblib is a Python library that provides tools for serialization (saving and loading) of Python objects, particularly for objects that have large data structures such as machine learning models.

Joblib is commonly used for efficient persistence and retrieval of machine learning models, especially those trained on large datasets. It offers functionality to store Python objects, such as scikit-learn models, efficiently on disk, and retrieve them later for reuse. It is often used in combination with popular machine learning libraries like scikit-learn.

***This is our code for saving the model:-**

```
In [30]: import joblib

In [31]: joblib.dump(rf_model,"random forest algorithm final")
Out[31]: ['random forest algorithm final']

In [ ]:
```

***And this is also the code for loading the model :-**

```
In [32]: joblib.load("random forest algorithm final")
Out[32]: RandomForestClassifier
         RandomForestClassifier(max_depth=7, max_features=2, min_samples_split=25,
                                random_state=42)

In [ ]:
```

Creating the API ,upload the model

API stands for Application Programming Interface. It is a set of rules and protocols that allows different software applications to communicate and interact with each other. APIs define how different components of software systems should interact, specifying the methods, data formats, and rules for accessing and exchanging information.

APIs can be used to access functionality and data from external services or libraries, enabling developers to incorporate pre-built capabilities into their own applications. They provide a way for developers to interact with software components without needing to know the internal implementation details.

There are different types of APIs, including:

1- Web APIs: These are APIs that are exposed over the web and allow applications to communicate and exchange data over HTTP. Web APIs are commonly used for integration with web services, accessing data from external sources, or building web-based applications.

2- Library APIs: These are APIs provided by libraries or frameworks that allow developers to use the

functionality and features provided by the library. Library APIs define the interfaces, classes, and methods that developers can use to interact with the library.

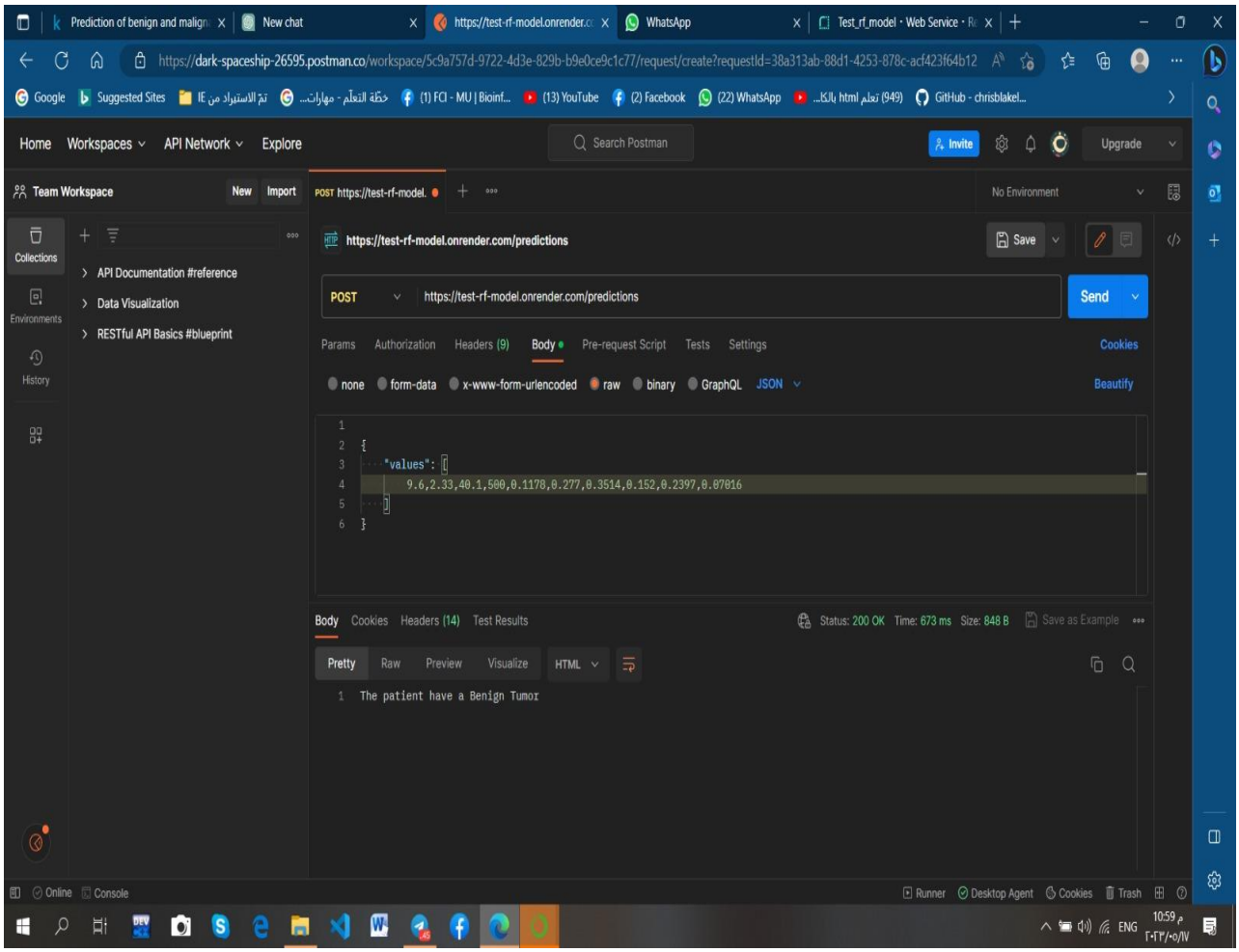
3- Operating System APIs: These are APIs provided by operating systems that enable developers to interact with the underlying system resources and services. Operating system APIs allow applications to access hardware devices, manage files and processes, and perform system-level operations.

4- Database APIs: These are APIs provided by database management systems that allow developers to interact with databases. Database APIs define the methods and protocols for querying, inserting, updating, and deleting data in the database.

APIs are commonly used in software development to enable interoperability, code reusability, and integration with external services. They provide a standardized way for different software components to communicate and exchange data, making it easier to build complex applications by leveraging existing functionalities and services.

***We used Flask library in python to create our web api and upload the model on a server .**

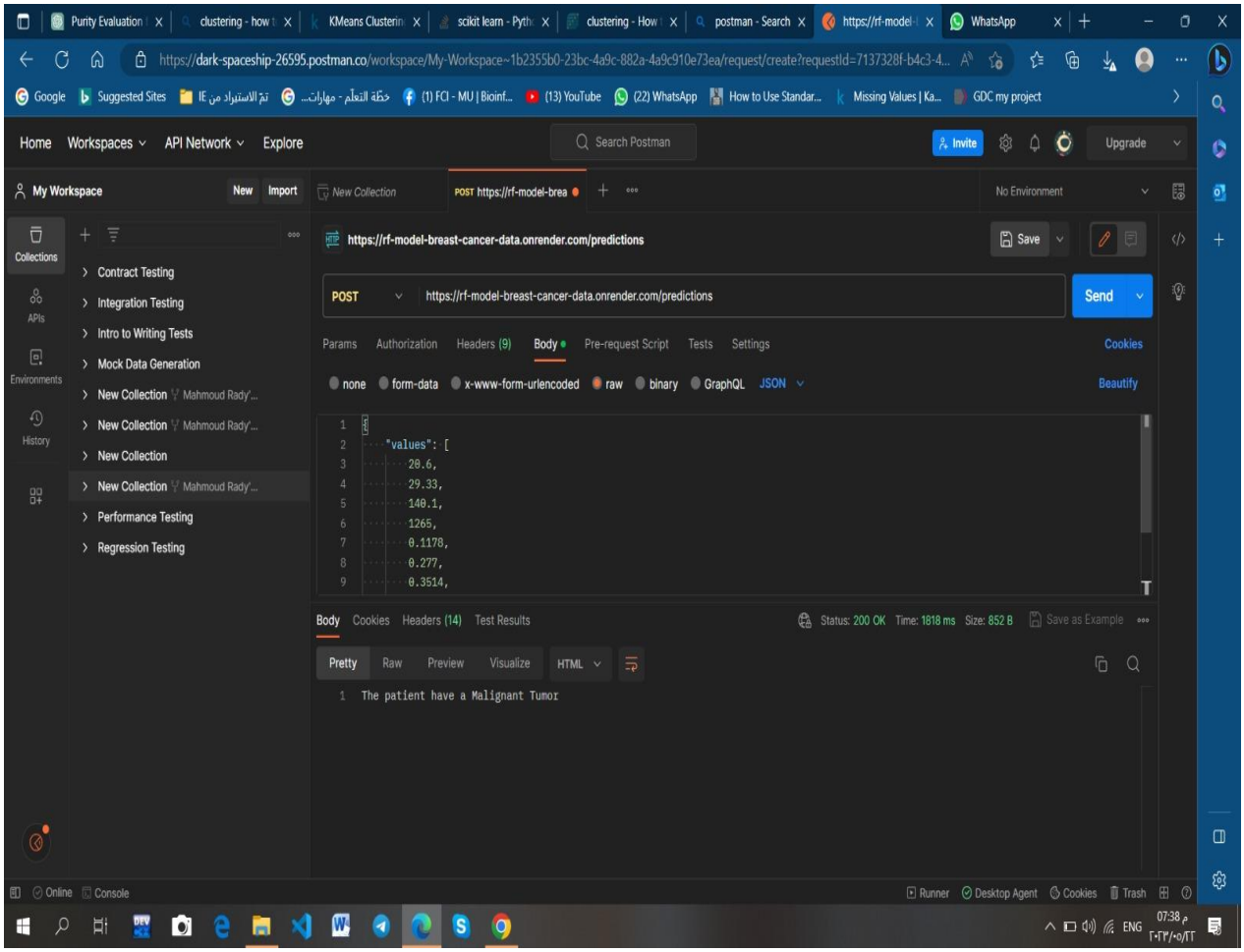
-After we create the Api and upload the model on a server; we test the API and the model using Postman website and Postman agent and we send some requests to the server and we can see the response in the figures below :-



*** here we send a data as numbers in a variable called values from our dataset with label b and the response was that :**

"The patient have a Benign Tumor" and this is true.

***so the api is working and the model is good also.**



- here we can see another values with the another label from the dataset and the result was true and the model is working good because the response was : "The patient have a Malignant Tumor".