BREAST CANCER WISCONSIN EXECUTIVE REPORT

MELVIN ACHIENG OWINO

Melvinachieng297@gmail.com

Table of Contents

Abstract

The problem statement describes a binary classification problem where breast cancer patients need to get accurately diagnosed based on the cancer cells. The diagnosis column has two predicted outcomes: ‘M’ or ‘B’. M standing for Malignant and B standing for benign. To solve the problem, we make use of machine learning algorithms and evaluate their performance using real life breast cancer Wisconsin dataset. The binary classification problem will make use of steps such as data exploration, preprocessing, model building, evaluation, hyperparameter tuning and deployment for testing its ability to make future predictions. The aim is to find a model that works best for helping hospitals and cancer patients know the correct diagnosis and hence the correct treatment for their breast cancer condition.

Introduction

Breast Cancer cells are referred to as abnormal cells found in the breast area. There are two types of diagnosis for breast cancer cells: malignant (M) and benign (B). The abnormal growth cells cause tumors which can be cancerous or non-cancerous. The cancerous tumor cells are said to be malignant while those that are non-cancerous are said to be benign. It is, therefore, import to determine the kind of abnormal growth a woman has for treatment purposes. The malignant tumor is that which is primarily referred to as breast cancer because of its cancerous nature. Also, malignant breast cancer poses a risk of metastasizing to other areas of a woman’s body. It is therefore, important for the female population to get tested for breast cancer and get the correct diagnosis. While observing a mass on the breast is one way of symptoms of breast cancer it is important to be sure whether the mass is malignant or benign. The use of Machine Learning models makes it possible for diagnosis of such cancer patients with assurance while taking important measures such as area mean, radius mean, perimeter mean, mass and other important factors. The project made use of machine learning to test whether it is a reliable concept in making correcting prediction of a person’s diagnosis of breast cancer. The project made use of important steps for machine learning prediction purposes that include: data exploration, data preprocessing, model building and evaluation to find the best Machine Learning Algorithm to use for diagnosis of breast cancer.

Methodology

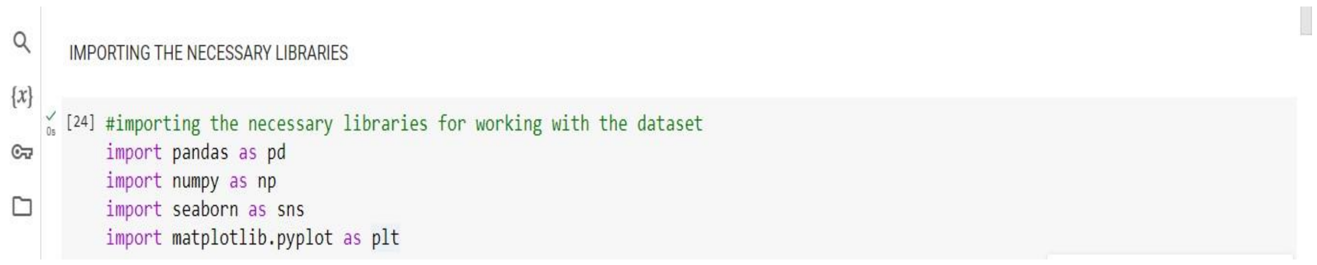
The methodology use aims at finding the best machine learning model for prediction of whether a person's breast cancer is benign or malignant based on different features. The project will make use of python script to figure out the kind of accurate cancer diagnosis. The aim of the project is to find the most accurate machine learning model suited for prediction of breast cancer diagnosis whether malignant or benign.

Data Exploration

Data Exploration was conducted using the breast cancer Wisconsin dataset. The dataset which was initially in csv format was uploaded to the python google colab notebook. The initial dataset comprised of 33 columns and 569 rows before data cleaning and removal of outliers from the dataset.

Importing necessary libraries

We began the data exploration in google colab notebook by importing the necessary libraries required for reading the csv file and data mining, manipulation, and handling missing values.

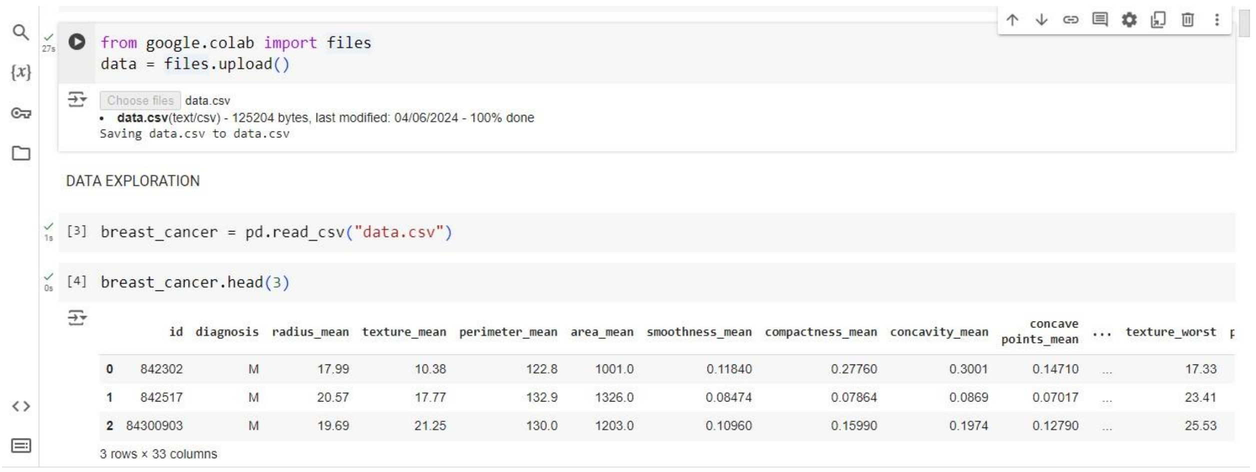


IMPORTING THE NECESSARY LIBRARIES

```
[24] #importing the necessary libraries for working with the dataset
     import pandas as pd
     import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt
```

Loading the dataset

The second step after importing the necessary libraries was loading the csv dataset with the help of pandas library with the variable name pd. After reading the csv file, the next step was printing out the first three rows to check the overall make up of the csv dataset.



```
from google.colab import files
data = files.upload()
```

```
Choose files  data.csv
• data.csv(text/csv) - 125204 bytes, last modified: 04/06/2024 - 100% done
Saving data.csv to data.csv
```

DATA EXPLORATION

```
[3] breast_cancer = pd.read_csv("data.csv")
```

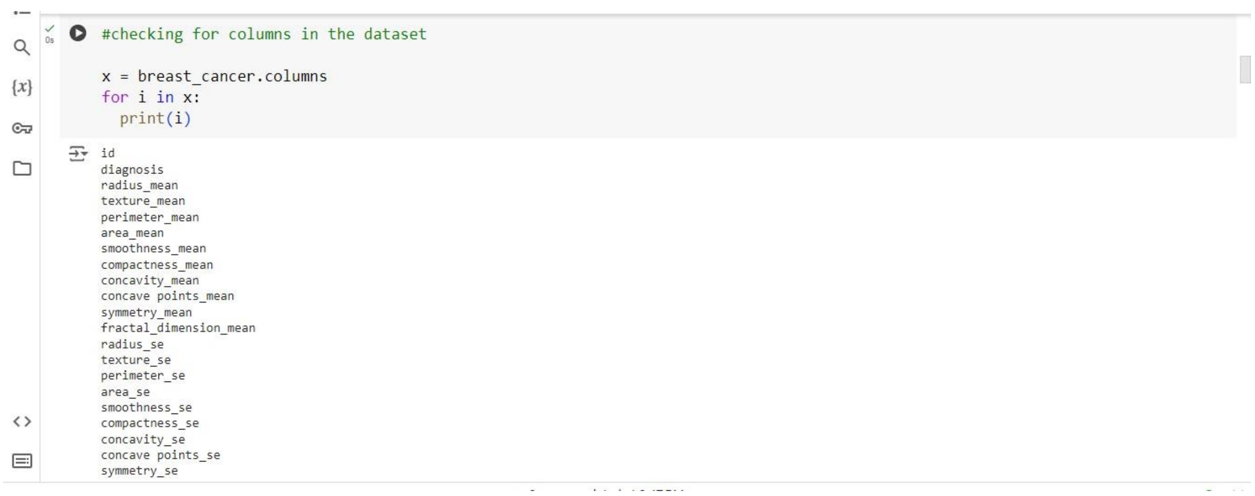
```
[4] breast_cancer.head(3)
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	texture_worst
0	842302	M	17.99	10.38	122.8	1001.0	0.11840	0.27760	0.3001	0.14710	...	17.33
1	842517	M	20.57	17.77	132.9	1326.0	0.08474	0.07864	0.0869	0.07017	...	23.41
2	84300903	M	19.69	21.25	130.0	1203.0	0.10960	0.15990	0.1974	0.12790	...	25.53

3 rows × 33 columns

Checking for columns in the dataset

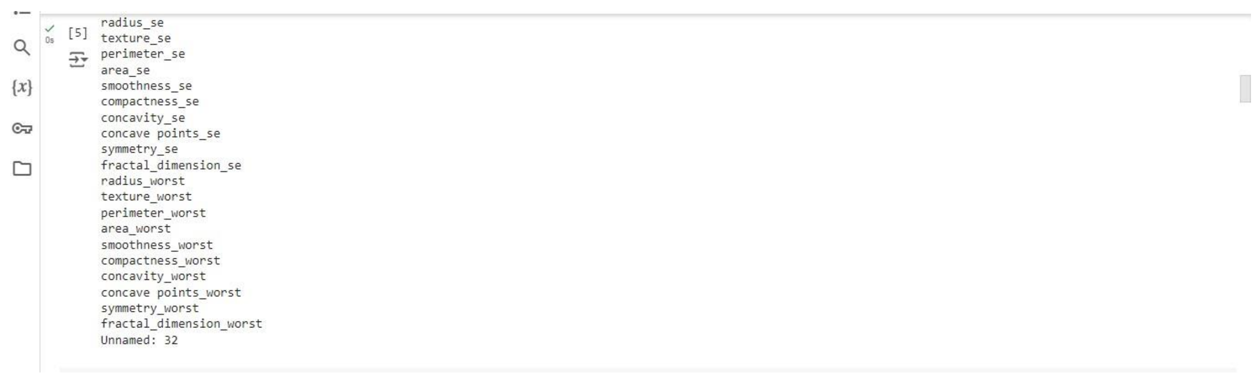
It is important to check for the columns in the dataset as part of analyzing the structure of the dataset. It also gives insights on the type of data that was collected for machine learning purposes. It also gives insights on the important datasets and the ones that need to get dropped.



```
#checking for columns in the dataset

x = breast_cancer.columns
for i in x:
  print(i)
```

```
id
diagnosis
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
```

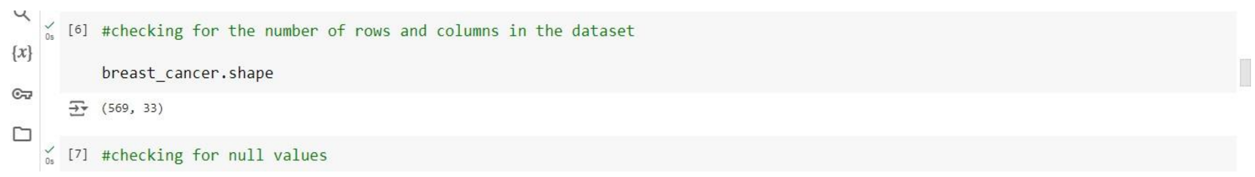
The rest of the columns are seen in the next screenshot since the dataset had 32 total columns



```
[5]
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
Unnamed: 32
```

Checking the number of rows and columns in the dataset

The next step was checking for the shape of the dataset which shows the number of rows and columns in the dataset



```
[6] #checking for the number of rows and columns in the dataset

breast_cancer.shape
```

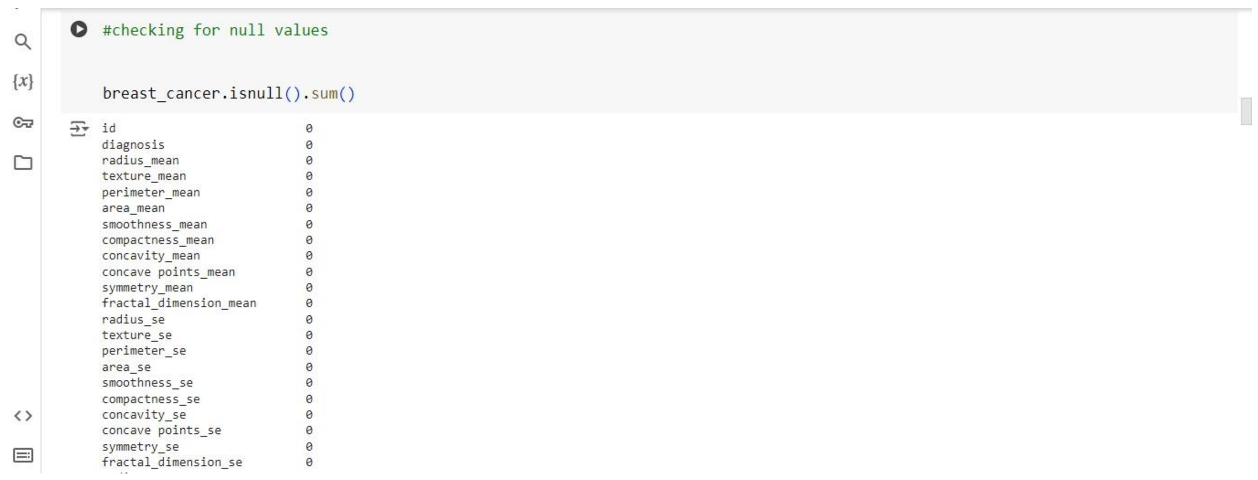
```
(569, 33)
```

```
[7] #checking for null values
```

Before cleaning the dataset, the number of rows and columns were as displayed above 569 rows and 33 columns

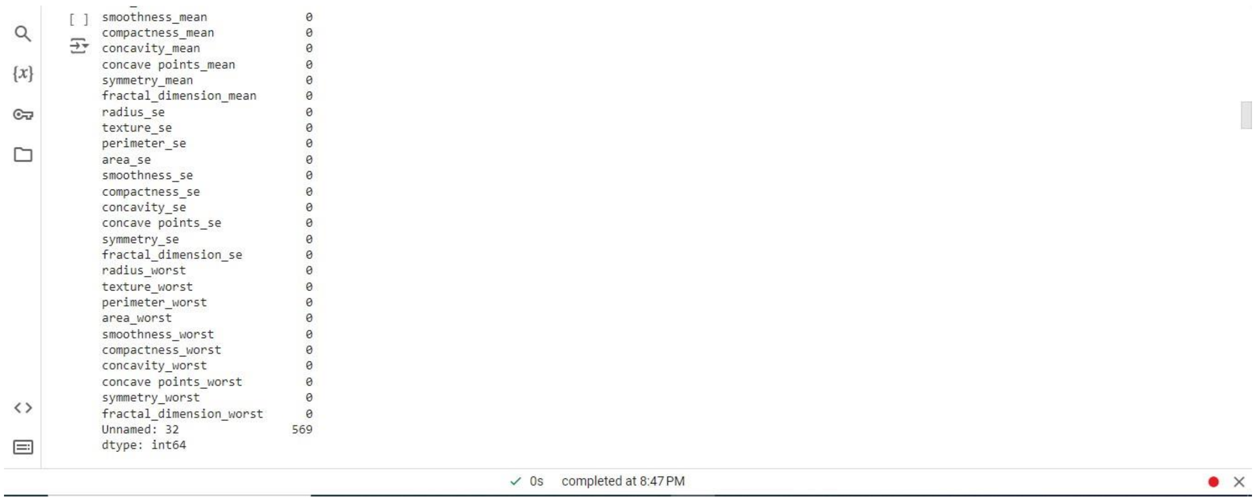
Checking for missing values in the columns

The next step was to check if any of the 33 columns had missing values as part of the data exploration.



```
#checking for null values

breast_cancer.isnull().sum()
```

```
id                         0
diagnosis                  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
```

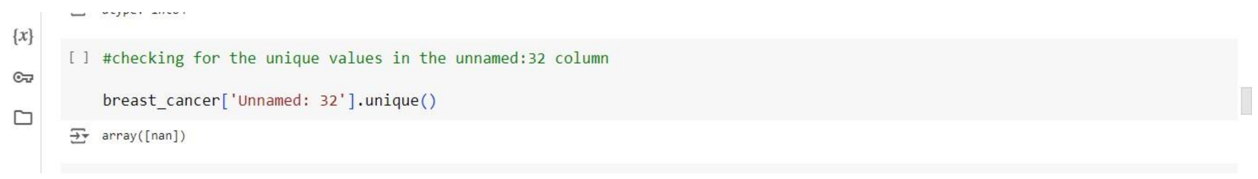


```
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              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
Unnamed: 32              569
dtype: int64
```

✓ 0s completed at 8:47 PM

The column with variable name ‘unnamed:32’ was the only column that contained missing values

Investigating further to find the categories under the column showed that the unnamed:32 column contain no variables hence it was a column that was created with no input created

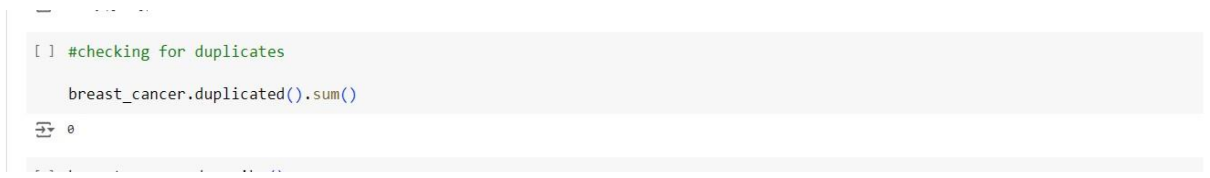


```
#checking for the unique values in the unnamed:32 column

breast_cancer['Unnamed: 32'].unique()
```

```
array([nan])
```

Checking for duplicates in the dataset

The next was to check for duplicated in the dataset. It is important to remove duplicates when they are many and can affect the prediction of the model negatively.



```
#checking for duplicates

breast_cancer.duplicated().sum()
```

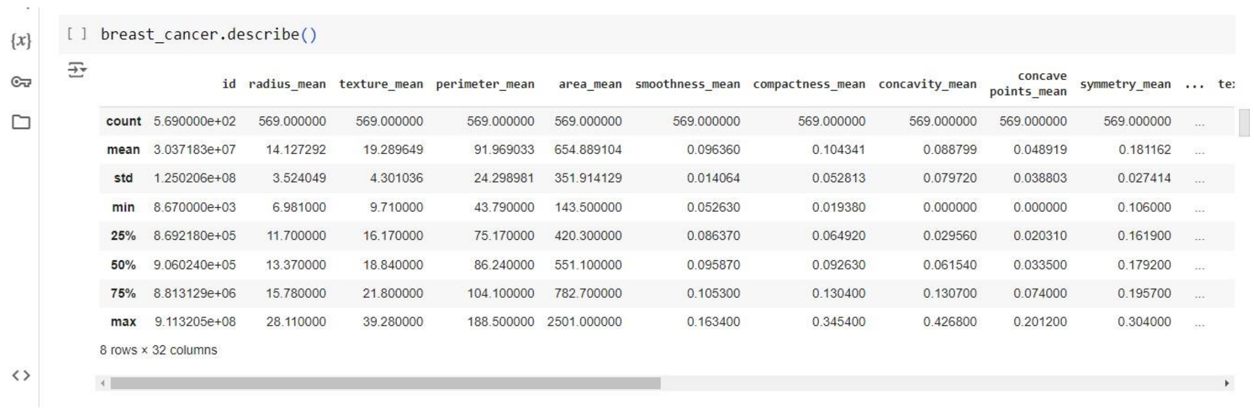
```
0
```

The result showed that the breast cancer Wisconsin dataset did not have any duplicated information hence we did not need to drop any duplicates.

Statistical Analysis of the Dataset

The next step was to check the overall statistical view of the features of the dataset.

The statistical analysis would also point out to the cases of outliers in the dataset.



```
breast_cancer.describe()
```

```
                 id  radius_mean  texture_mean  perimeter_mean    area_mean  smoothness_mean  compactness_mean  concavity_mean     concave  symmetry_mean  ...  te
                                                                                                                                           points_mean
count  5.690000e+02   569.000000    569.000000      569.000000   569.000000       569.000000        569.000000      569.000000  569.000000     569.000000  ...
mean   3.037183e+07    14.127292     19.289649       91.969033   654.889104         0.096360          0.104341        0.088799    0.048919       0.181162  ...
std    1.250206e+08     3.524049      4.301036       24.298981   351.914129         0.014064          0.052813        0.079720    0.038803       0.027414  ...
min    8.670000e+03     6.981000      9.710000       43.790000   143.500000         0.052630          0.019380        0.000000    0.000000       0.106000  ...
25%    8.692180e+05    11.700000     16.170000       75.170000   420.300000         0.086370          0.064920        0.029560    0.020310       0.161900  ...
50%    9.060240e+05    13.370000     18.840000       86.240000   551.100000         0.095870          0.092630        0.061540    0.033500       0.179200  ...
75%    8.813129e+06    15.780000     21.800000      104.100000   782.700000         0.105300          0.130400        0.130700    0.074000       0.195700  ...
max    9.113205e+08    28.110000     39.280000      188.500000  2501.000000         0.163400          0.345400        0.426800    0.201200       0.304000  ...
8 rows × 32 columns
```

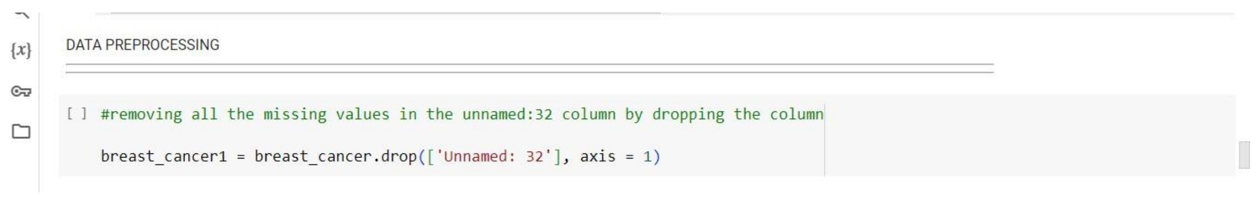
The statistical analysis showed the maximum and minimum values. Also, the 25th, 50th, and 75th quantile which is very useful when calculating outliers and removing them from the dataset.

DATA PREPROCESSING

After conducting exploration of the dataset, it was important to ensure the data was clean and any missing values and outliers were handled before using the data for machine learning purposes.

Removing the missing values

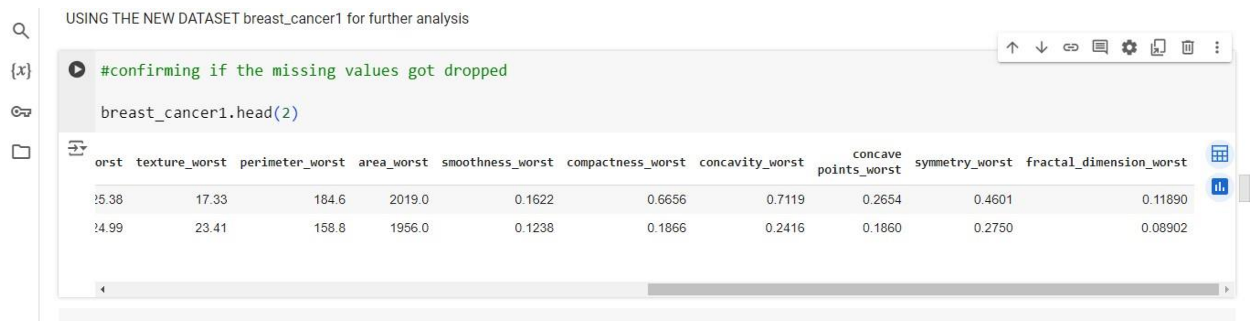
The first step was removing the missing values from the breast cancer dataset which made up the 'unnamed:32' column and creating a new variable name for the new dataset without the dropped column



DATA PREPROCESSING

```
#removing all the missing values in the unnamed:32 column by dropping the column

breast_cancer1 = breast_cancer.drop(['Unnamed: 32'], axis = 1)
```

The next step was confirming that the dropped column actually got dropped in the new dataset with a new variable name ‘breast_cancer1’.



USING THE NEW DATASET breast_cancer1 for further analysis

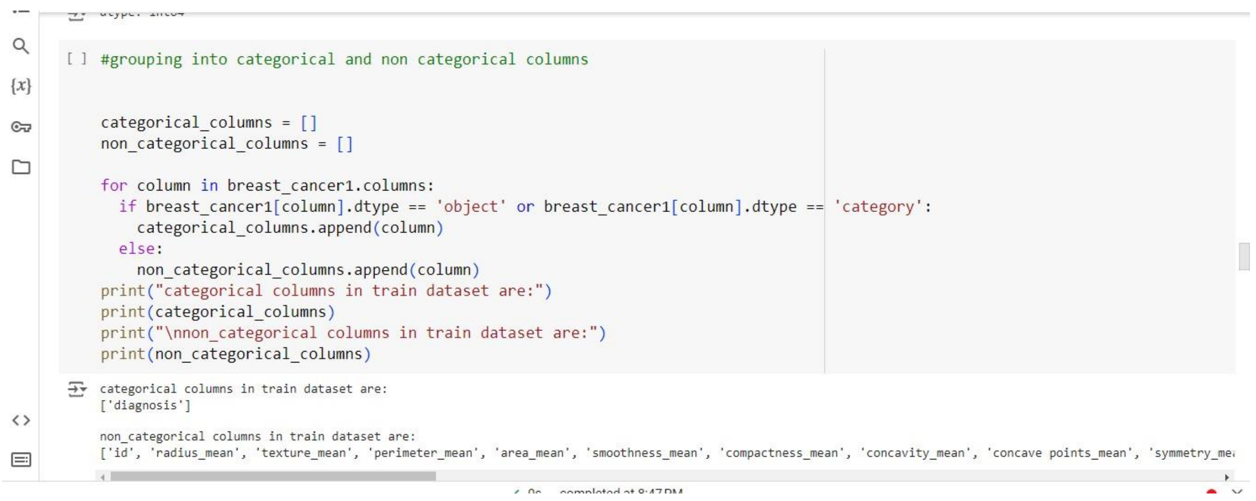
```
#confirming if the missing values got dropped

breast_cancer1.head(2)
```

orst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	fractal_dimension_worst
›5.38	17.33	184.6	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890
›4.99	23.41	158.8	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902

The next step was confirming that no missing values existed in the new dataset with a new variable name.

Classifying the columns

The next step was classifying the columns into categorical and non-categorical columns. It is important to identify the categorical and non-categorical columns for encoding purposes, visualizations and correlation purposes.

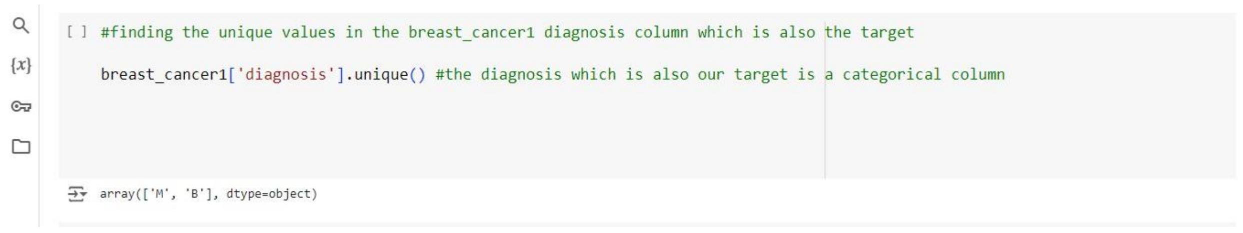


```
#grouping into categorical and non categorical columns


categorical_columns = []
non_categorical_columns = []

for column in breast_cancer1.columns:
  if breast_cancer1[column].dtype == 'object' or breast_cancer1[column].dtype == 'category':
    categorical_columns.append(column)
  else:
    non_categorical_columns.append(column)
print("categorical columns in train dataset are:")
print(categorical_columns)
print("\nnon_categorical columns in train dataset are:")
print(non_categorical_columns)
```

```
categorical columns in train dataset are:
['diagnosis']

non_categorical columns in train dataset are:
['id', 'radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean', 'concave points_mean', 'symmetry_me
```

Finding Unique Values

The next step was identifying the unique values in the target categorical variable which is the ‘diagnosis’ column.



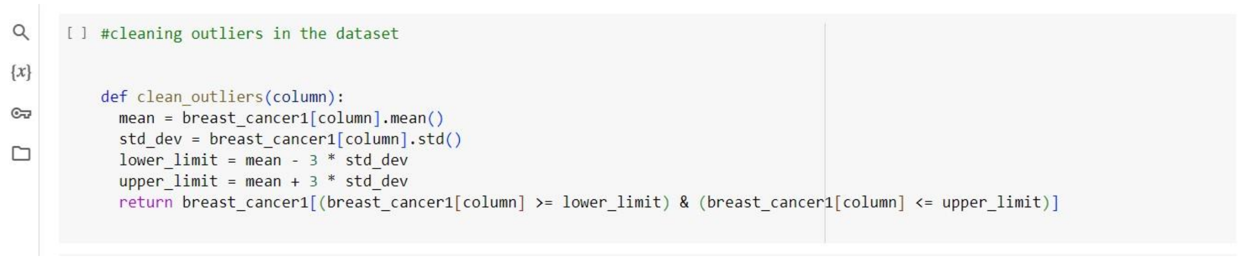
```
#finding the unique values in the breast_cancer1 diagnosis column which is also the target

breast_cancer1['diagnosis'].unique() #the diagnosis which is also our target is a categorical column
```

```
array(['M', 'B'], dtype=object)
```

The diagnosis target variable has only two unique values: ‘M’ and ‘B’. Whereas ‘M’ stands for Malignant and ‘B’ stands for Benign which are the two outcomes that a patient with breast cancer may have and hence our prediction target.

Cleaning outliers in the dataset

The next step was ensuring that the new dataset is cleaned from outliers that can negatively affect the results and prediction of the machine learning model

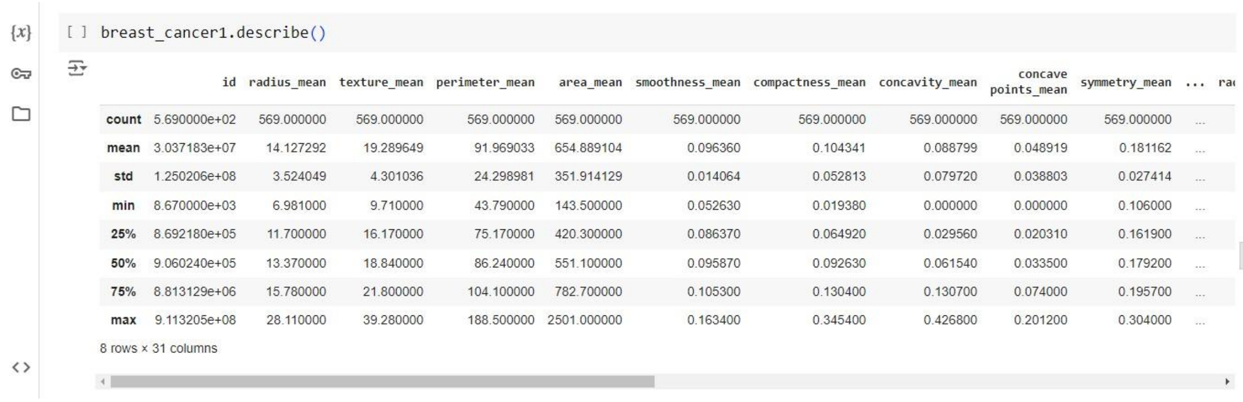


```
#cleaning outliers in the dataset


def clean_outliers(column):
  mean = breast_cancer1[column].mean()
  std_dev = breast_cancer1[column].std()
  lower_limit = mean - 3 * std_dev
  upper_limit = mean + 3 * std_dev
  return breast_cancer1[(breast_cancer1[column] >= lower_limit) & (breast_cancer1[column] <= upper_limit)]
```

The function ensured that the new dataset that was returned did not have outliers after getting removed using the lower and upper limit rule.

Checking for the new statistical Analysis

The new statistical analysis of the dataset will have a deviation from the previous dataset which had outliers.



```
breast_cancer1.describe()
```

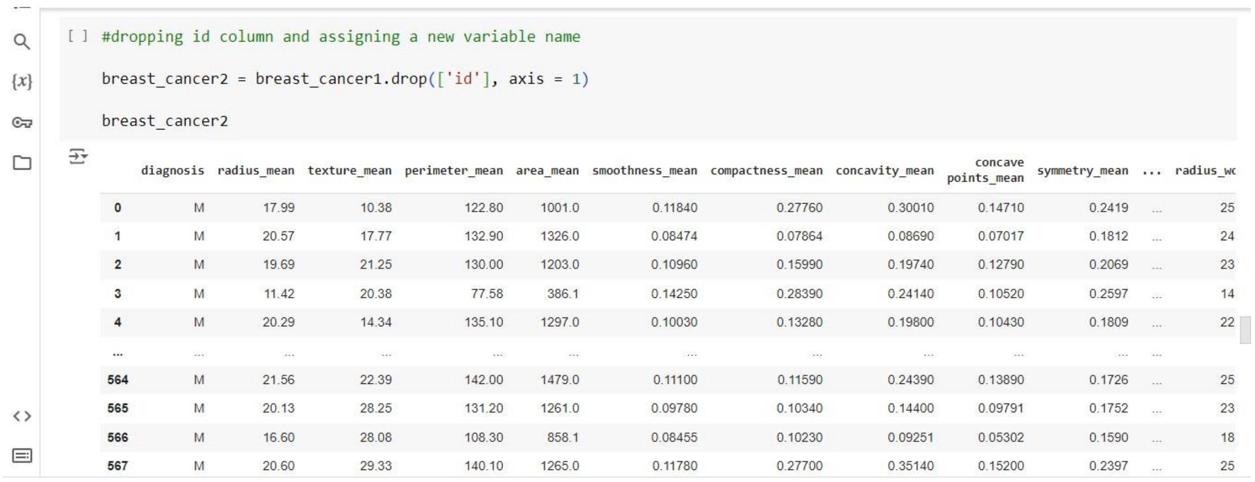
	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	...	ra
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	...	
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	...	
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	...	
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	...	
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	...	
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	...	
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	...	

8 rows × 31 columns

The mean, standard deviation, and quantile figures have changed from the previously done statistical analysis where outliers were present.

Dropping the Id Column

The next step was dropping the id column from the dataset and creating a new dataset with a new variable name without the id column. Dropping the id column is justified since it does not have any effect on the dataset hence to reduce the noise, we drop it since it only serves as a unique identifier for each record input.



```
#dropping id column and assigning a new variable name

breast_cancer2 = breast_cancer1.drop(['id'], axis = 1)

breast_cancer2
```

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	...	radius_wc
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	...	25
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	...	24
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	...	23
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	...	14
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	...	22
...	...	...	...	...	...	...	...	...	...	...	...	
564	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	...	25
565	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	...	23
566	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	...	18
567	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	...	25

The new variable name for the new dataset without the id column was 'breast_cancer2' as shown above.

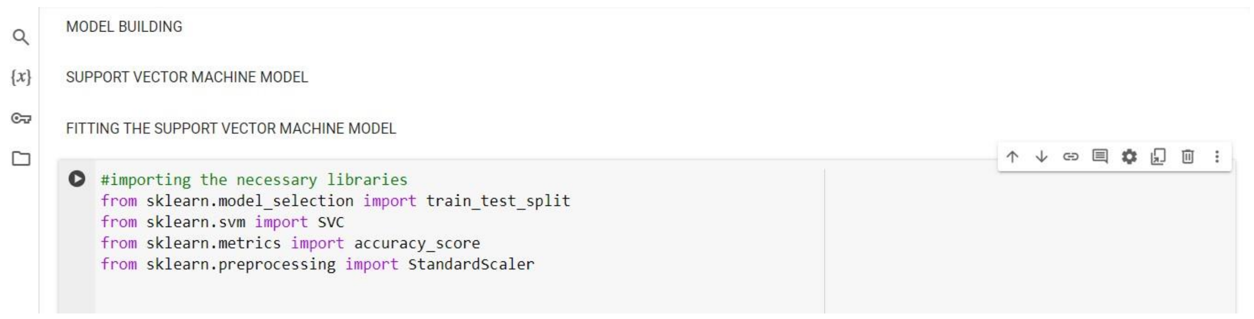
Model Building

Support Vector Machine

The support vector machine learning algorithm is a type of supervised machine learning method. Since, the machine is being trained on data that is labeled and getting used for prediction of future breast cancer diagnosis it forms a supervised machine learning method. Also, the problem is a binary classification problem which is under supervised machine learning. Hence Support vector machine is one of the most suitable methods of predicting the outcome and testing the dataset for its ability to predict future breast cancer diagnosis.

Importing the necessary libraries

We begin by importing the necessary libraries for conducting SVM learning. The needed libraries are imported from sklearn which has the sub-libraries for training, testing, splitting, the dataset and selecting the SVM model and calculating the accuracy score by importing the metrics sub-library.



MODEL BUILDING

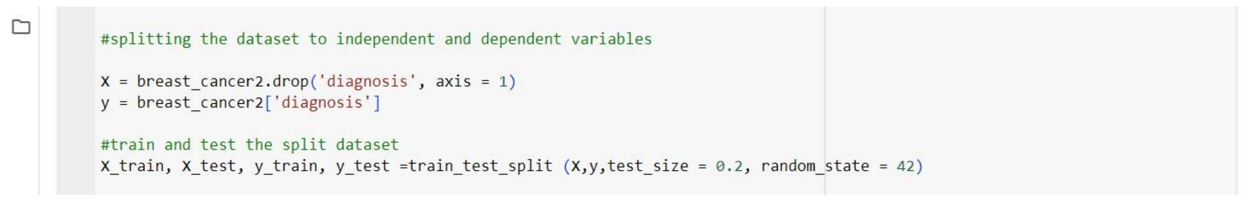
SUPPORT VECTOR MACHINE MODEL

FITTING THE SUPPORT VECTOR MACHINE MODEL

```
#importing the necessary libraries
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
```

Defining variables and training them

We then define our X and y variables in which X are the independent variables affecting our y which is the dependent variable that we are also predicting. The diagnosis makes our y since we want to make future accurate predictions of the probability of a breast cancer patient having malignant or benign cancer cells.

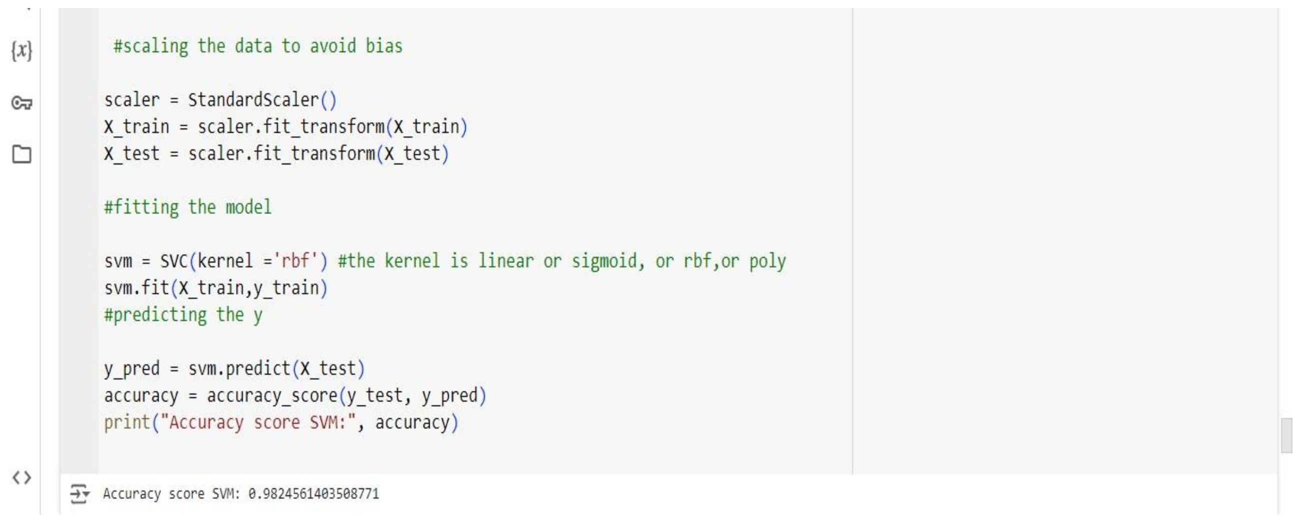


```
#splitting the dataset to independent and dependent variables

X = breast_cancer2.drop('diagnosis', axis = 1)
y = breast_cancer2['diagnosis']

#train and test the split dataset
X_train, X_test, y_train, y_test =train_test_split (X,y,test_size = 0.2, random_state = 42)
```

After defining our X and y we train and test the dataset in a 80-20 ratio respectively.

Scaling, Fitting, and measuring the Accuracy of the model

After defining our X and y variables we therefore scale the model to ensure it does not overfit or underfit hence reducing any bias and noise in the dataset used for prediction. We also fit and measure the accuracy of the model and hence gauge its ability to make accurate predictions of future breast cancer diagnosis.



```
 #scaling the data to avoid bias

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)

#fitting the model

svm = SVC(kernel ='rbf') #the kernel is linear or sigmoid, or rbf,or poly
svm.fit(X_train,y_train)
#predicting the y

y_pred = svm.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy score SVM:", accuracy)
```

```
Accuracy score SVM: 0.9824561403508771
```

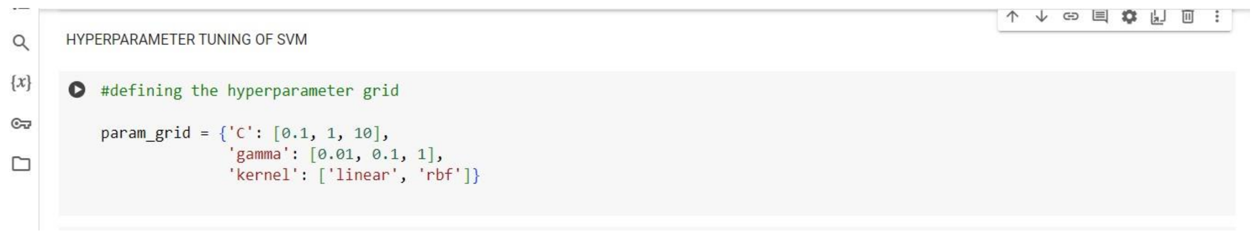
The model accuracy score is 0.9824 which means we are 98% confident that the results predicted using the SVM model are 98%accurate. Hence, it shows that the model is very good in using it to make future predictions on diagnosis of breast cancer patients whether their cancer cells are benign or malignant.

Hyperparameter tuning of SVM

Afte measuring the initial accuracy score of the SVM model, I went a step further to tune the model and see if the accuracy could increase even further despite the model already being very good for making future predictions.

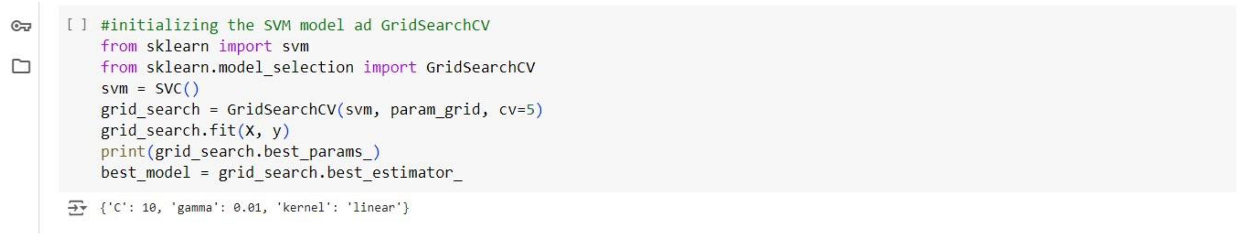
Defining the parameter grid

We first begin by defining the hyperparameters that are relevant for tuning the SVM model.


HYPERPARAMETER TUNING OF SVM

```
#defining the hyperparameter grid

param_grid = {'C': [0.1, 1, 10],
              'gamma': [0.01, 0.1, 1],
              'kernel': ['linear', 'rbf']}
```

The relevant parameters for the SVM tuning are ‘C’, ‘gamma’ and ‘kernel’. We then find the best parameters for the model based on the dataset to tune the model to fit even more.



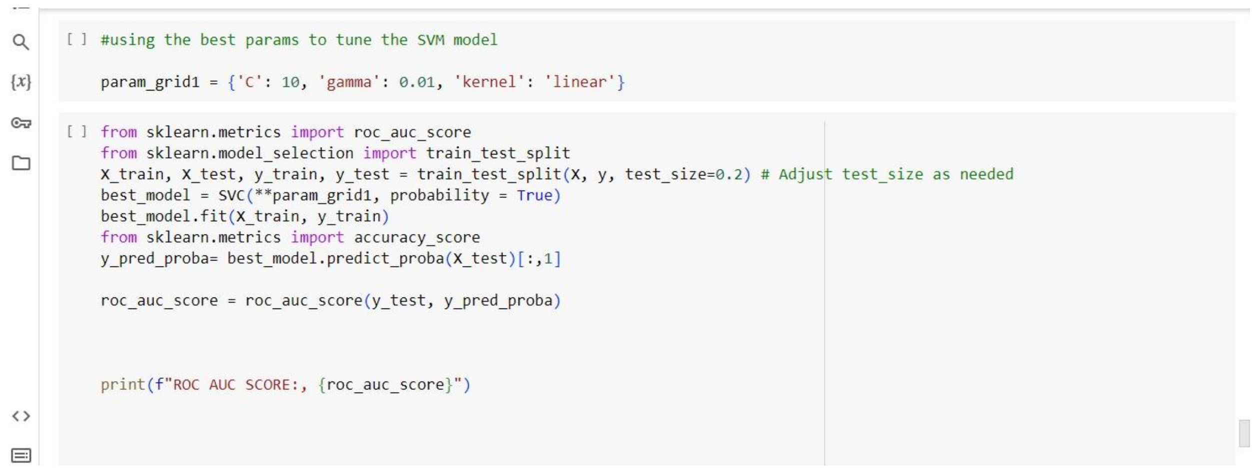
```
#initializing the SVM model ad GridSearchCV
from sklearn import svm
from sklearn.model_selection import GridSearchCV
svm = SVC()
grid_search = GridSearchCV(svm, param_grid, cv=5)
grid_search.fit(X, y)
print(grid_search.best_params_)
best_model = grid_search.best_estimator_
```

```
{'C': 10, 'gamma': 0.01, 'kernel': 'linear'}
```

The best parameters identified for tuning the SVM model are C:10, gamma:0.01 and kernel: linear.

Incorporating the chosen parameters

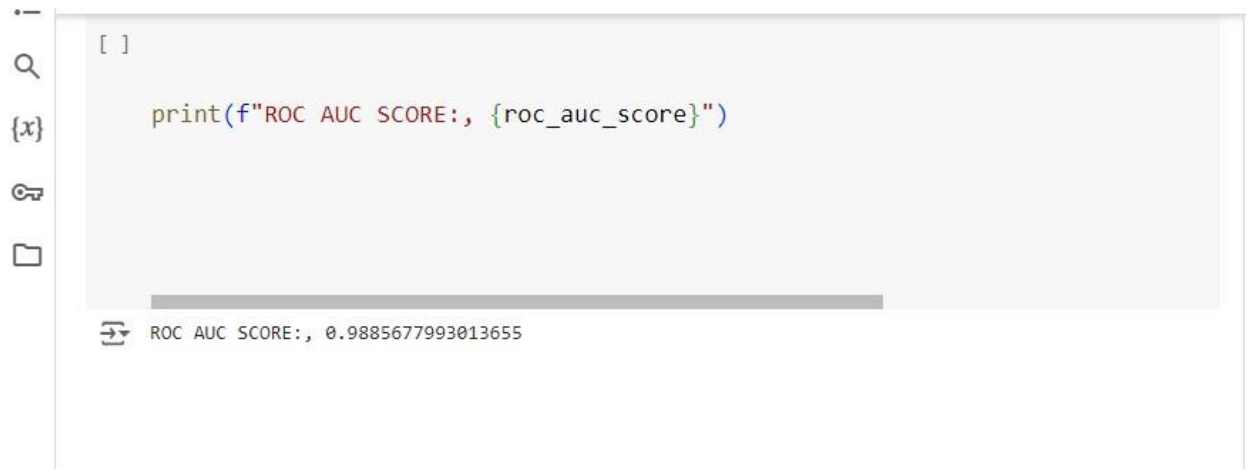
The next step after choosing the best parameters is incorporating it to the model and tune it further and improve its prediction performance.



```
[ ] #using the best params to tune the SVM model

    param_grid1 = {'C': 10, 'gamma': 0.01, 'kernel': 'linear'}
```

```
[ ] from sklearn.metrics import roc_auc_score
    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2) # Adjust test_size as needed
    best_model = SVC(**param_grid1, probability = True)
    best_model.fit(X_train, y_train)
    from sklearn.metrics import accuracy_score
    y_pred_proba= best_model.predict_proba(X_test)[:,1]

    roc_auc_score = roc_auc_score(y_test, y_pred_proba)



    print(f"ROC AUC SCORE:, {roc_auc_score}")
```

We imported roc_auc_score from sklearn.metrics library since we have a binary classification problem and we needed to measure the new model performance after tuning it. A score of 1 shows a perfect model while a score of 0.5 shows the model is doing random guessing hence not suitable for prediction. A higher roc_auc_score of 90% and above signifies the model performs well for binary classification tasks.



```
[ ]

    print(f"ROC AUC SCORE:, {roc_auc_score}")
```

```
ROC AUC SCORE:, 0.9885677993013655
```

The model after hyperparameter tuning has a score 0.9886 which is approximately 99%. Hence, showing that using the SVM model for this binary classification task will yield very accurate prediction values for the diagnosis problem.

References

1. http:// www.imaginis.com/breasthealth/breast_cancer.asp, Last Accessed August 2007.

2. West, D., Mangiameli, P., Rampal, R., & West, V. (2005). Ensemble strategies for a medical diagnosis decision support system: A breast cancer diagnosis application. European Journal of Operational Research (162), 532–551

3. https://www.mayoclinic.org/diseases-conditions/breast-cancer/symptoms-causes/syc□20352470

4. Pavlopoulos, S.A.; Delopoulos, A.N. Designing and implementing the transition to a fully digital hospital. IEEE Trans. Inf. Technol. Biomed. 1999, 3, 6–19.

5. Barracliffe, L.; Arandjelović, O.; Humphris, G. A pilot study of breast cancer patients: Can machine learning predicts healthcare professionals' responses to patient emotions? In Proceedings of the International Conference on Bioinformatics and Computational Biology, Honolulu, HI, USA, 20–22 March 2017; pp. 101–106.

6. Birkett, C.; Arandjelović, O.; Humphris, G. Towards objective and reproducible study of patient-doctor interaction: Automatic text analysis based VR-CoDES annotation of consultation transcripts. In Proceedings of the IEEE Engineering in Medicine and Biology Society Conference, Jeju Island, Korea, 11–15 July 2017; pp. 2638–2641