Breast Cancer Prediction

Data Collection

data.shape

(569, 33)

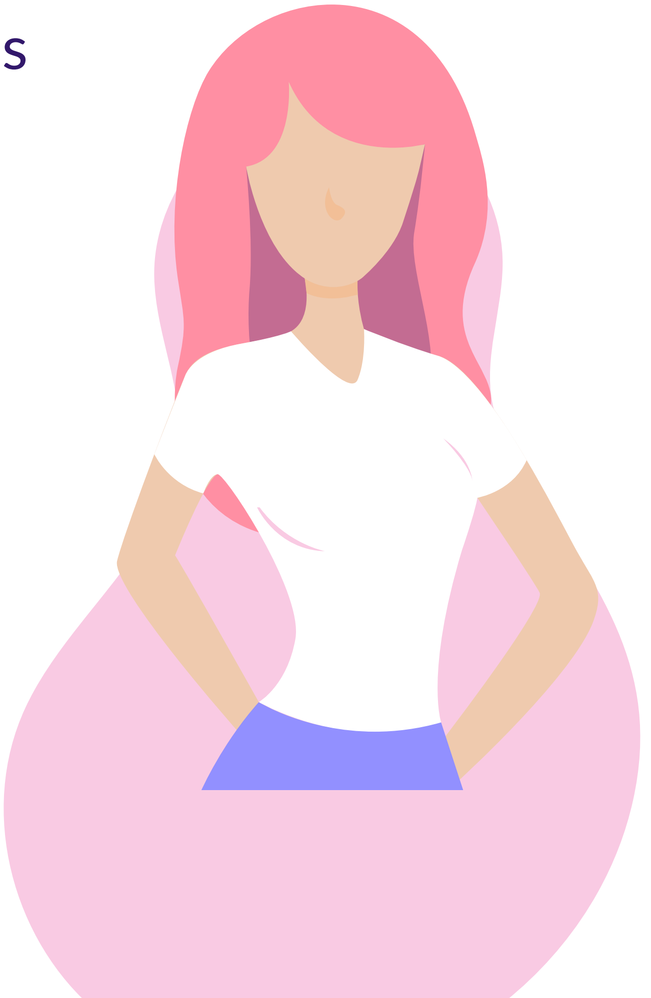
Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

M = malignant, B = benign

Exploring Data Analysis

data.info()

0 id
1 diagnosis
2 radius_mean
3 texture_mean
4 perimeter_mean
5 area_mean
6 smoothness_mean
7 compactness_mean
8 concavity_mean
9 concave points_
10 symmetry_mean
11 fractal_dimension_mean
12 radius_se
13 texture_se
14 perimeter_se
15 area_se
16 smoothness_se
17 compactness_se
18 concavity_se
19 concave points_se
20 symmetry_se
21 fractal_dimension_se
22 radius_worst
23 texture_worst
24 perimeter_worst
25 area_worst
26 smoothness_worst
27 compactness_worst
28 concavity_worst
29 concave
30 symmetry_worst
31 fractal_dimension_worst
32 Unnamed: 32



Only one single feature is categorical and it's values are B and M :

diagnosis

data.describe(include="O")

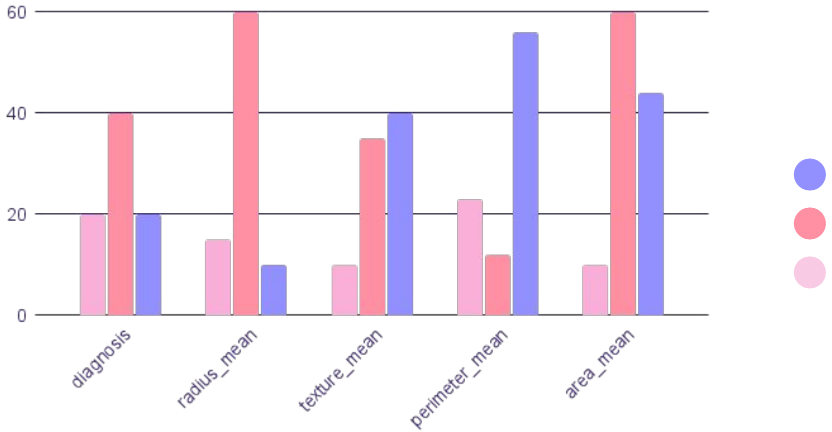
	diagnosis
count	569
unique	2
top	B
freq	357

Using this method we see number of unique values in categorical type of feature

```
data.diagnosis.value_counts()
```

```
B   357
M   212
```

Data Visualization




```
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
```

Data Filtering

We have one categorical feature, so we need to convert it into numeric values using LabelEncoder :

ML models are supposed to deal only with numerical values

- M
- B

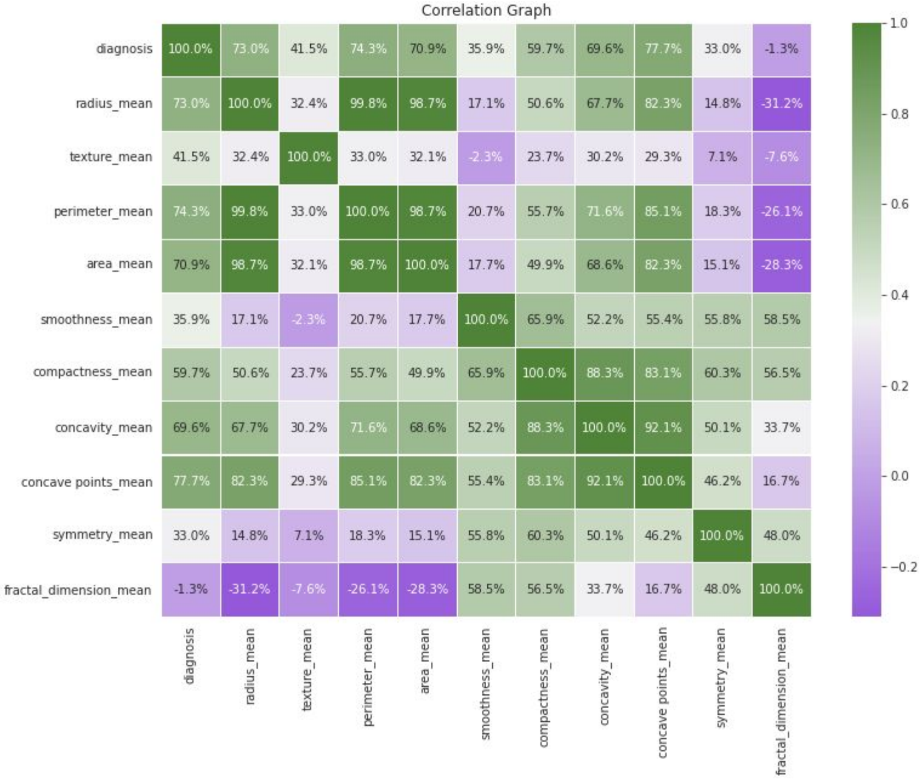
- 1
- 0

0	357
1	212

Find the correlation between other features

How features correspond with the output

data[cols].corr()

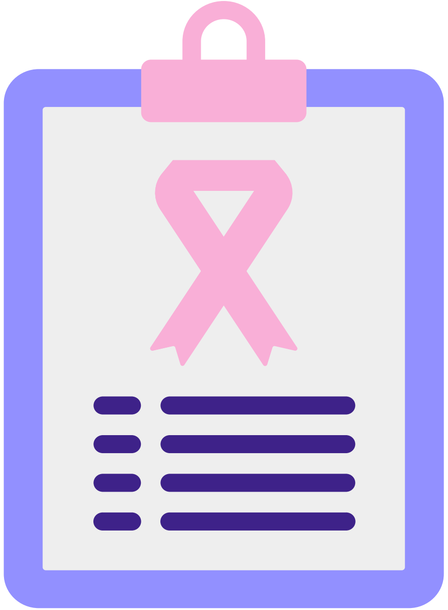


Model Implementation

Preprocessing and model selection

```
from sklearn.model_selection import
train_test_split

from sklearn.preprocessing import
StandardScaler
```



Compare the relative value of different statistical models and determine which one is the best fit for the observed data .

Machine Learning Models

01

LogisticRegression

02

RandomForestClassifier

03

DecisionTreeClassifier

04

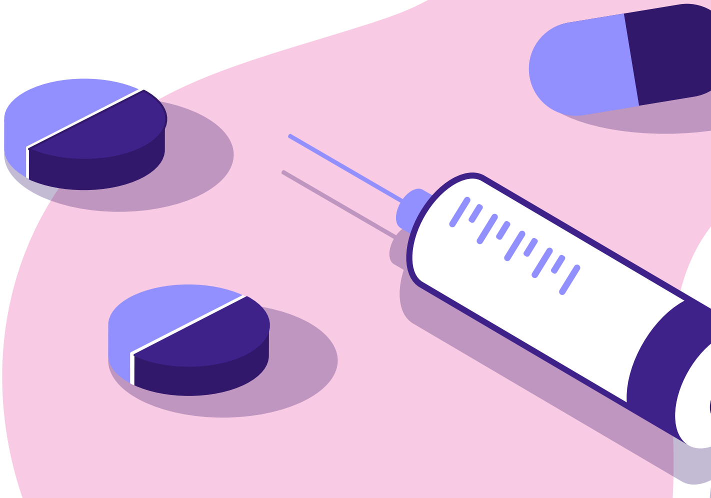
SVC

05

GaussianNB

06

KNeighborsClassifier



To check the Model Accuracy, Errors and it's Validations i’ve used :

```
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score

from sklearn.metrics import classification_report

from sklearn.model_selection import KFold

from sklearn.model_selection import cross_validate, cross_val_score

from sklearn.svm import SVC

from sklearn import metrics
```

Feature Selection

```
prediction_feature = [ ’radius_mean’,
'perimeter_mean', 'area_mean',
'symmetry_mean', 'compactness_mean',
'concave points_mean']

targeted_feature = 'diagnosis'
```

Split the dataset into Training Set 🩺 and Testing Set 🔬 by 33% and set the 15 fixed records

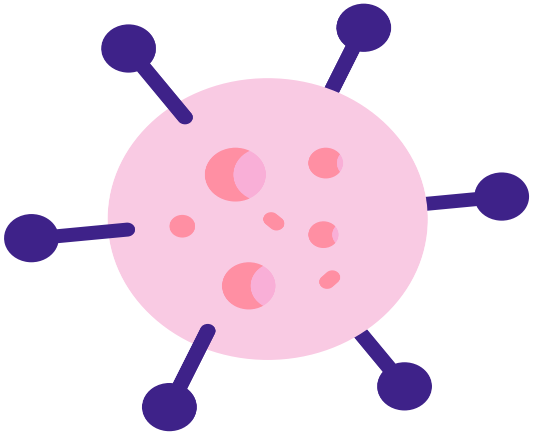
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=15)
```

Perform Feature Standard Scaling

Standardize features by removing the mean and scaling to unit variance

Scale the data to keep all the values in the same magnitude

StandardScaler()



Model Building

```
def model_building(model, X_train, X_test,
y_train, y_test):

    model.fit(X_train, y_train)
    score = model.score(X_train, y_train)
    predictions = model.predict(X_test)
    accuracy = accuracy_score(predictions,
y_test)

    return (score, accuracy, predictions)
```

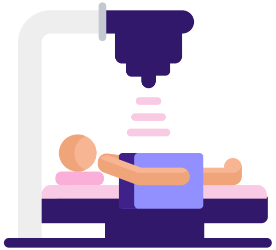
A dictionary for multiple models for bulk predictions

```
"LogisticRegression"
"RandomForestClassifier"
"DecisionTreeClassifier"
"SVC"
```

While Predict we can store model's score & prediction values to new generated dataframe

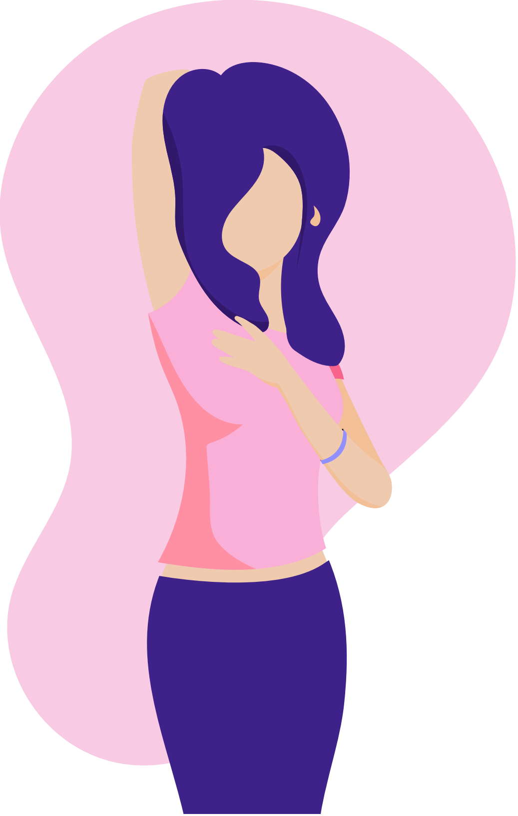
	model_name	score	accuracy_score	accuracy_percentage
0	LogisticRegression	0.916010	0.909574	90.96%
1	RandomForestClassifier	0.992126	0.925532	92.55%
2	DecisionTreeClassifier	1.000000	0.909574	90.96%
3	SVC	0.923885	0.914894	91.49%

Cross-validation is primarily used in applied machine learning **to estimate the skill of a machine learning model on unseen data**. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.



Let's define a function for cross validation scoring for multiple ML models

Call the function to know the cross validation function by mean for our select model prediction



```
Full-Data Accuracy: 1.0
Cross Validation Score of'RandomForestClassifier '

Score: 0.99
Score: 0.99
Score: 0.99
Score: 1.0
Score: 1.0

Full-Data Accuracy: 1.0
Cross Validation Score of'DecisionTreeClassifier '

Score: 1.0
Score: 1.0
Score: 1.0
Score: 1.0
Score: 1.0
```

Some of the model are giving prefect scoring . It means sometimes overfitting occurs .

HyperTuning the ML Model

For HyperTunning I used **GridSearchCV** to know the best performing parameters.

Hyperparameters can have a big impact on model training as it relates to **training time**, **infrastructure resource requirements** , **model convergence** and **model accuracy** .

DecisionTreeClassifier : Max_features / min_samples_split / min_samples_leaf

KNeighborsClassifier : N_neighbors / leaf_size / weights

SVC : C / kernel / gamma

RandomForestClassifier : bootstrap / max_depth / max_features / min_samples_leaf / min_samples_split / n_estimators

Deploy Model

Finally, we are done so far.
The last step is to deploy our model in production map.
So we need to export our model and bind with web application API.

Pickle is the standard way of serializing objects in Python

I've used streamlit to build the Web App

Presented By Wiem BELHADJ

Thanks

Demo