Module Title: Artificial Intelligence and Machine Learning
Module Code: CSMAI21
Lecturer Responsible: Dr Yevgeniya Kovalchuk
Type of Assignment: Individual
Student Name: Saanjanna Yuvaraj
Student Number: 28812368
Actual hours spent for this assignment: 20 hours

TABLE OF CONTENT

1. INTRODUCTION

The “Artificial Intelligence and Machine Learning” coursework details the data Science processes with Python as the primary tool for developing a comprehensive data Science workflow, and the results are assessed and studied for various models. This coursework specifically focuses on AI and ML. Artificial Intelligence (AI) is a technique for building systems that mimic human behaviour or decision-making. Machine Learning(ML) is a subset of AI that uses data to solve tasks. These solvers are trained models of data that learn based on their information. In. this coursework, the following items are submitted. A single zip archive containing;

a) Report (PDF File)
b) Dataset(s) (CSV file(s))
c) Python script(s) (PY file(s) [not IPYNB])

2. ABSTRACT

Machine Learning is to learn machines based on various training and testing data and determine the results in every condition without explicit programmed. The dataset explained in the coursework represents a machine learning model for Breast Cancer in classification. The machine learning model is compared with the support vector machine and decision tree and the Artificial neural network deep learning model. One of the machine learning models (Artificial neural network)is based on deep learning architecture and implemented using Keras python library. The dataset is pre-processed, such as data cleaning, Exploratory Data Analysis (EDA), and implemented using evaluation and tuning the model. The hybrid machine learning model for the breast cancer dataset reported better results using the Support vector machine with an accuracy of 97.07% After Tuning the Accuracy is 98.62%compared to the decision tree model and deep learning model.

Keywords
Breast Cancer; Machine Learning; Classification, Decision tree Support Vector Machine; and Artificial Neural Network

3. BACKGROUND

Machine Learning is a branch of Artificial Intelligence, a scientific discipline concerned with the design and development of algorithms based on data. As a result, it is more feasible to automatically assess the available data to determine breast cancer survival rate and risk-specific factors. Fig.1 shows generalized machine learning architecture for breast cancer, data set collection, select attributes, and feature extraction[4].
The coursework has been taken with 70% of the training dataset and 30% of the Testing dataset.

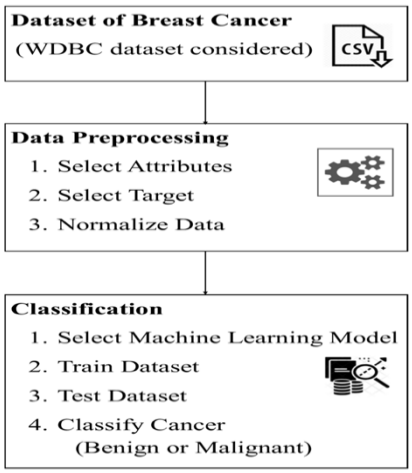


Fig. 1. The ML Architecture of Breast Cancer

4. LITERATURE REVIEW

Telsang VA Et al. [4] proposed a machine-learning algorithm to detect and predict breast cancer. The support vector machine studies the performance of their work, and compared with the other machine learning model. The Accuracy of the Support vector machine is stated as superior to the other models like Naive Bias, Random Forest, Logistic Regression, K- Nearest Neighbour. The obtained test results were compared with the studies in the previous literature. Further research was proposed to carry out detailed research of ML algorithms to increase the mathematical models for predicting breast cancer. Zhang J Et al. [5] This paper presents an instance-based learning approach in typical instances. First, this paper approaches the issues of measuring the typicality of an instance then shows some concepts to have graded structures. Then finally, it uses instance-based learning and classification algorithm that uses the typicality of instances. It has been tested on artificial and practical domains and compared with three different IBL approaches. The results show that the approach has been recorded with lower storage requirements and higher classification accuracy than previous instance-based algorithms on several domains.

5. DATASET DESCRIPTION

This dataset describes the Wisconsin Breast Cancer Database (classification) obtained from the University of Wisconsin. The source was from Dr William H. Wolberg. The one-time usage of the dataset has two attributes through ten that have been used to represent instances with two possible classes called benign or malignant. The most common disease and the leading cause of death to most women across the globe are Breast Cancer (BC). Although many individuals with breast cancer have no family history, women who have blood relatives suffering from the same disease are at higher risk [4]. This dataset is used to predict breast cancer early using different machine learning models. This coursework uses various machine learning and deep learning models like decision trees, Support Vector Machine, and Artificial Neural Network. The author in this dataset, Wolberg, W .H. has the size of the dataset only 369 instances. Two pairs of simultaneous hyperplanes were found to be consistent with 50% of the data Accuracy on the remaining 50% of the dataset: 93.5% and Three pairs of simultaneous hyperplanes were found to be consistent with 67% of data Accuracy on the remaining 33% of dataset: 95.9% and the other author is Zhang.J has the size of the data set is 369 instances applied with four instance-based learning algorithms and Collected classification results averaged over ten trial. The Best accuracy result:1-nearest neighbour: 93.7% trained on 200 instances, tested on the other 169. Later on, the instances were 699, with ten plus with class attributes. The dataset has a missing value of 16. Sixteen instances in Groups 1 to 6 contain a single missing (i.e., unavailable) attribute value, now denoted by "?". The class distribution was benign 458 (66.5%) and Malignant 241(34.5%)

6. MACHINE LEARNING MODEL, AND SUMMARY OF THE APPROACH

The “Artificial Intelligence and Machine Learning ” coursework details the Data Science process with Python as the primary tool for developing a comprehensive Data Science workflow[1]. The assessment focuses on implementing four different models with breast cancer datasets from the raw data gathering, pre-processing such as data cleaning, Exploratory Data Analysis (EDA), Building and implementing a network model, and evaluating the model. In addition, a deep learning neural network has been implemented using Keras python library, whereas Keras has the critical features of modularity, Minimalistic, Extensibility. Neural networks, used in coursework, are a particular type of machine learning (ML) algorithm. So, like every machine learning algorithm, it follows the usual Machine learning workflow of data pre-processing, model building and model evaluation[2]. The below figure shows the workflow of machine learning models.

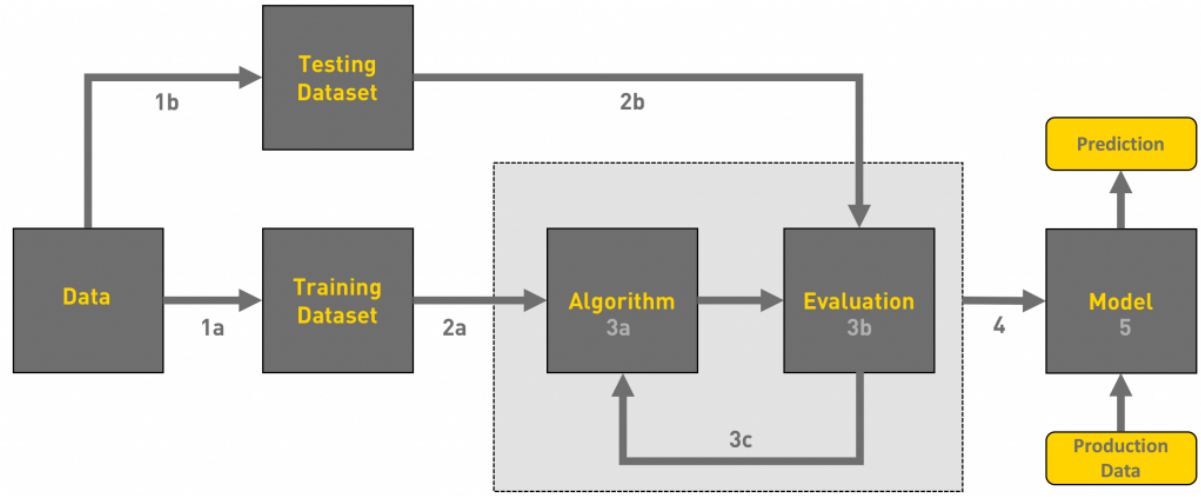


Fig. 2. The Workflow of Machine Learning Model

7. DATA VISUALISATION, PRE-PROCESSING AND FEATURE SELECTION

In today's world, the majority of women who has breast cancer have no family history, but women who have blood relatives suffering from the same disease are at higher risk; therefore, this dataset is chosen to predict breast cancer at the early stage using various machine and deep learning models. The snippet below captures Python code covering data visualisation, breast cancer data set pre-processing, and feature selection.

The below is built in python libraries.

```
[ ] ######################################Decision Tree Model#################################################
```

The Dataset described in this course work is Wisconsin Breast cancer Database.(Classification) The model used in this is Decision tree , support vector Machine Deep Learning(Neural Network)

```
######################################Built-in Python Libraries / Dependencies#######################################
import pandas as pd
import numpy as np
from pandas import read_csv
from google.colab import files
from sklearn.naive_bayes import MultinomialNB
import io
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import KFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as lda
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.model_selection import cross_val_score
import pandas as pd
```

```
[1] from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
np.random.seed(123)  # for reproducibility
from matplotlib import pyplot as plt
%matplotlib inline
```

The below code snippet explains the data pre-processing like cleaning the data, removing duplicates, missing values, and null values.

```
plt.style.use('classic')
%matplotlib inline
uploaded = files.upload()
```

Choose files breastcancer.csv

- **breastcancer.csv**(text/csv) - 19889 bytes, last modified: 24/02/2022 - 100% done

```
Saving breastcancer.csv to breastcancer.csv
```

```
print (uploaded['breastcancer.csv'][:200].decode('utf-8') + '...')
```

```
1000025,5,1,1,1,2,1,3,1,1,2
1002945,5,4,4,5,7,10,3,2,1,2
1015425,3,1,1,1,2,2,3,1,1,2
1016277,6,8,8,1,3,4,3,7,1,2
1017023,4,1,1,3,2,1,3,1,1,2
1017122,8,10,10,8,7,10,9,7,1,4
1018099,1,1,1,1,2,10,3,1,1,2...
```

```
#To read the csv file
df = pd.read_csv(io.StringIO(uploaded['breastcancer.csv'].decode('utf-8')))
df.columns = ['Sample code number','Clump Thickness','Uniformity of Cell Size','Uniformity of Cell Shape','Marginal Adhesion','Single Epithelial Cell Size',
df
```

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
0	1002945	5	4	4	5	7	10	3	2	1	2
1	1015425	3	1	1	1	2	2	3	1	1	2
2	1016277	6	8	8	1	3	4	3	7	1	2
3	1017023	4	1	1	3	2	1	3	1	1	2
4	1017122	8	10	10	8	7	10	9	7	1	4
...	...	...	...	...	...	...	...	...	...	...	...
693	776715	3	1	1	1	3	2	1	1	1	2
694	841769	2	1	1	1	2	1	1	1	1	2
695	888820	5	10	10	3	7	3	8	10	2	4
696	897471	4	8	6	4	3	4	10	6	1	4
697	897471	4	8	8	5	4	5	10	4	1	4

698 rows × 11 columns

```
#First five rows of the data
df.head()
```

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
0	1002945	5	4	4	5	7	10	3	2	1	2
1	1015425	3	1	1	1	2	2	3	1	1	2
2	1016277	6	8	8	1	3	4	3	7	1	2
3	1017023	4	1	1	3	2	1	3	1	1	2
4	1017122	8	10	10	8	7	10	9	7	1	4

Pre-processing the Dataset

+ Code + Text

```
# Checking the data type
df.dtypes
```

```
Sample code number             int64
Clump Thickness                int64
Uniformity of Cell Size        int64
Uniformity of Cell Shape       int64
Marginal Adhesion              int64
Single Epithelial Cell Size    int64
Bare Nuclei                   object
Bland Chromatin                int64
Normal Nucleoli                int64
Mitoses                        int64
Class                          int64
dtype: object
```

```
# Checking the missing values
df.isnull().sum()
```

```
Sample code number            0
Clump Thickness               0
Uniformity of Cell Size       0
```

[7]
```
Uniformity of Cell Shape       0
Marginal Adhesion              0
Single Epithelial Cell Size    0
Bare Nuclei                    0
Bland Chromatin                0
Normal Nucleoli                0
Mitoses                        0
Class                          0
dtype: int64
```

```
# Dropping the duplicates
df = df.drop_duplicates()
df
```

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
0	1002945	5	4	4	5	7	10	3	2	1	2
1	1015425	3	1	1	1	2	2	3	1	1	2
2	1016277	6	8	8	1	3	4	3	7	1	2
3	1017023	4	1	1	3	2	1	3	1	1	2
4	1017122	8	10	10	8	7	10	9	7	1	4
...	...	...	...	...	...	...	...	...	...	...	...
693	776715	3	1	1	1	3	2	1	1	1	2
694	841769	2	1	1	1	2	1	1	1	1	2
695	888820	5	10	10	3	7	3	8	10	2	4
696	897471	4	8	6	4	3	4	10	6	1	4
697	897471	4	8	8	5	4	5	10	4	1	4

690 rows × 11 columns

[9]
```
df.isna().any().sum()
```

```
0
```

[10]
```
#Dropping the missing values
df.drop(df.loc[df['Bare Nuclei'] == "?"].index, inplace=True)
df
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4913: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  errors=errors,
```

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
0	1002945	5	4	4	5	7	10	3	2	1	2
1	1015425	3	1	1	1	2	2	3	1	1	2
2	1016277	6	8	8	1	3	4	3	7	1	2
3	1017023	4	1	1	3	2	1	3	1	1	2
4	1017122	8	10	10	8	7	10	9	7	1	4
...	...	...	...	...	...	...	...	...	...	...	...
693	776715	3	1	1	1	3	2	1	1	1	2
694	841769	2	1	1	1	2	1	1	1	1	2
695	888820	5	10	10	3	7	3	8	10	2	4
696	897471	4	8	6	4	3	4	10	6	1	4
697	897471	4	8	8	5	4	5	10	4	1	4

674 rows × 11 columns

[11]
```
#Total number of rows and columns
df.shape
```

```
(674, 11)
```

[12]
```
#Statistics of the data dataset
df.info
```

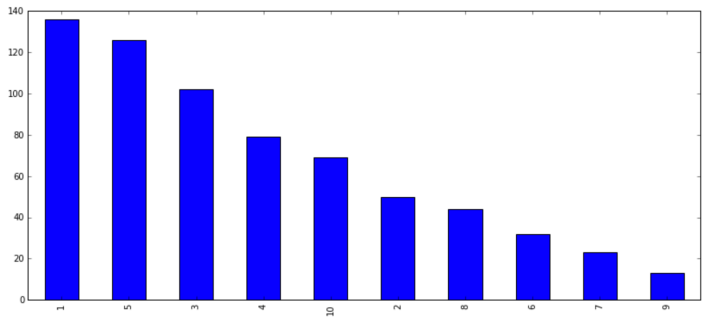
```
<bound method DataFrame.info of      Sample code number  Clump Thickness  Uniformity of Cell Size  \
0               1002945                5                        4
1               1015425                3                        1
2               1016277                6                        8
3               1017023                4                        1
4               1017122                8                       10
```

```
[12] ..                  ...                ...                   ...
     693              776715                  3                     1
     694              841769                  2                     1
     695              888820                  5                    10
     696              897471                  4                     8
     697              897471                  4                     8

          Uniformity of Cell Shape  Marginal Adhesion  Single Epithelial Cell Size  \
     0                           4                  5                            7
     1                           1                  1                            2
     2                           8                  1                            3
     3                           1                  3                            2
     4                          10                  8                            7
     ..                        ...                ...                          ...
     693                         1                  1                            3
     694                         1                  1                            2
     695                        10                  3                            7
     696                         6                  4                            3
     697                         8                  5                            4

         Bare Nuclei  Bland Chromatin  Normal Nucleoli  Mitoses  Class
     0            10                3                2        1      2
     1             2                3                1        1      2
     2             4                3                7        1      2
     3             1                3                1        1      2
     4            10                9                7        1      4
     ..          ...              ...              ...      ...    ...
     693           2                1                1        1      2
     694           1                1                1        1      2
     695           3                8               10        2      4
     696           4               10                6        1      4
     697           5               10                4        1      4

     [674 rows x 11 columns]>
```

The Bar Chart Representative of visualisation of breast cancer dataset

Visualisation

```
[13] # Plotting a Histogram
     ax = df['Clump Thickness'].value_counts().plot(kind='bar',figsize=(14,6))
```

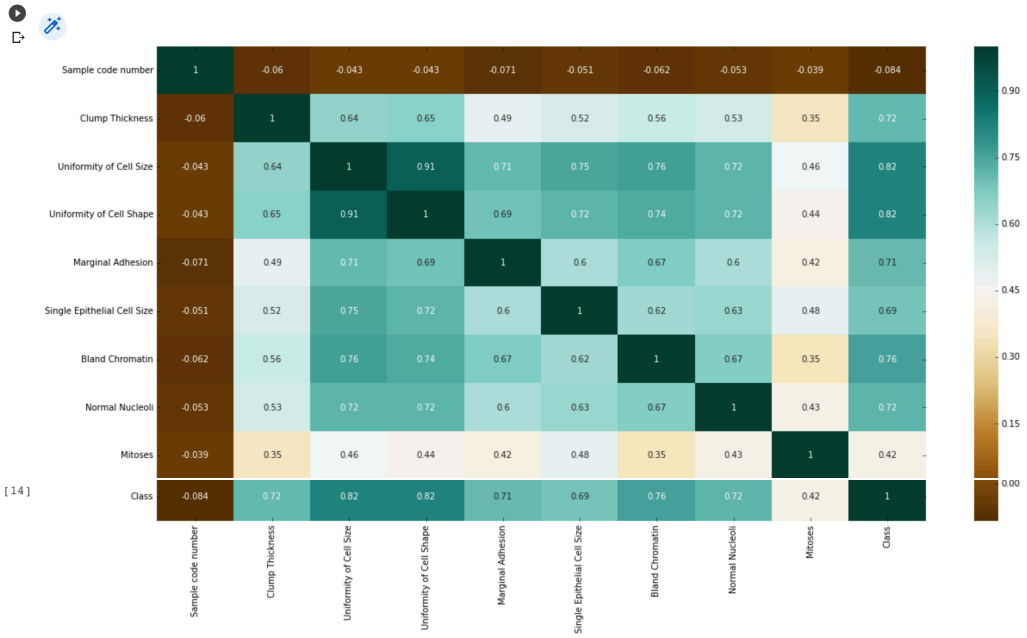


The above graph plots the clump thickness of the breast cancer and shows the various size of cells.

```
[14] # Finding the relations between the variables
     plt.figure(figsize=(20,10))
     c= df.corr()
     sns.heatmap(c,cmap="BrBG",annot=True)
```

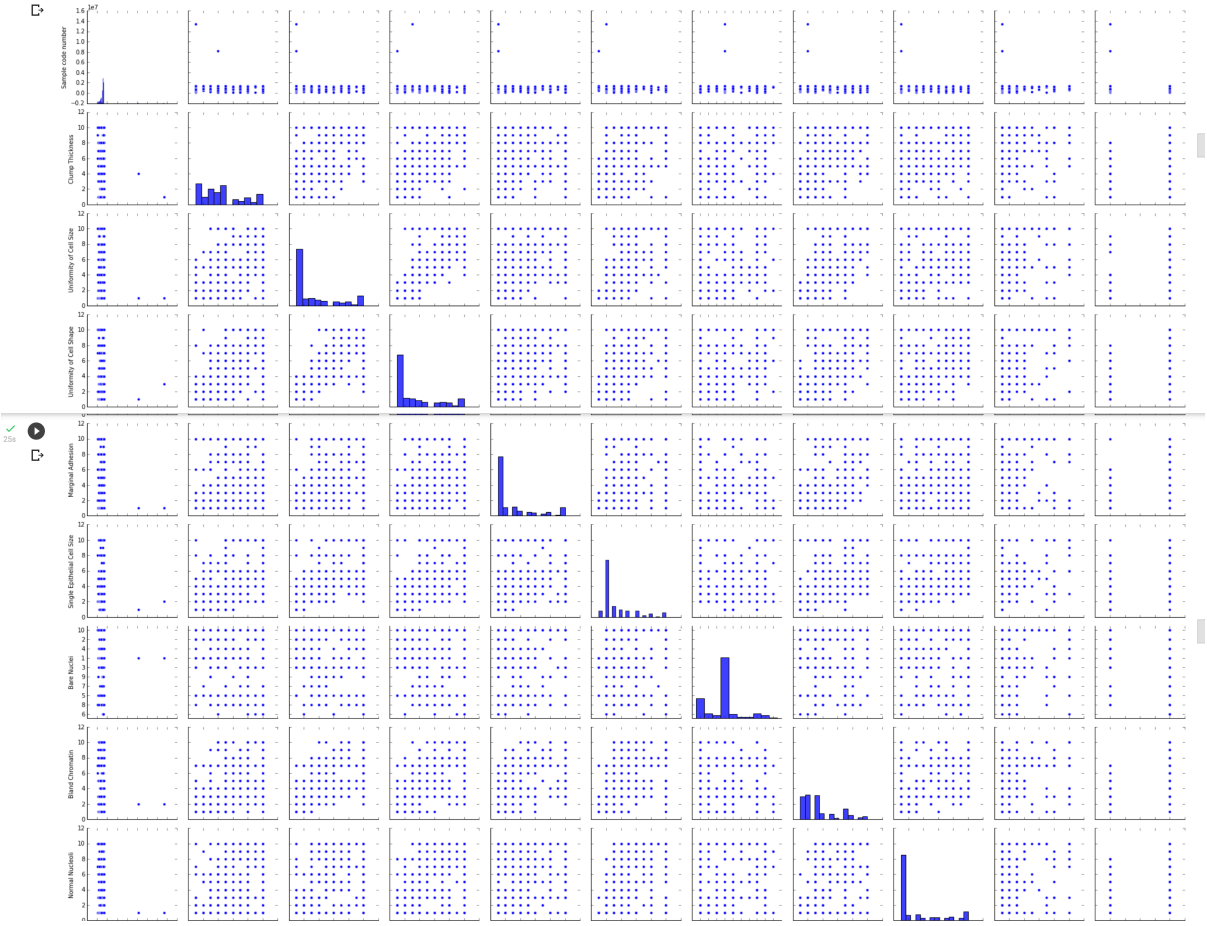
	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bland Chromatin	Normal Nucleoli	Mitoses	Class
Sample code number	1.000000	-0.060298	-0.043237	-0.042992	-0.071245	-0.050956	-0.061607	-0.052835	-0.039203	-0.084159
Clump Thickness	-0.060298	1.000000	0.641303	0.653117	0.489771	0.519467	0.556485	0.534234	0.351413	0.716232
Uniformity of Cell Size	-0.043237	0.641303	1.000000	0.905683	0.714460	0.748694	0.759526	0.723538	0.462032	0.820527
Uniformity of Cell Shape	-0.042992	0.653117	0.905683	1.000000	0.693817	0.717031	0.737994	0.723062	0.442850	0.820543
Marginal Adhesion	-0.071245	0.489771	0.714460	0.693817	1.000000	0.603272	0.671794	0.601954	0.418226	0.710786
Single Epithelial Cell Size	-0.050956	0.519467	0.748694	0.717031	0.603272	1.000000	0.622658	0.633944	0.484079	0.689372
Bland Chromatin	-0.061607	0.556485	0.759526	0.737994	0.671794	0.622658	1.000000	0.668860	0.345627	0.758376
Normal Nucleoli	-0.052835	0.534234	0.723538	0.723062	0.601954	0.633944	0.668860	1.000000	0.432881	0.721841
Mitoses	-0.039203	0.351413	0.462032	0.442850	0.418226	0.484079	0.345627	0.432881	1.000000	0.424227
Class	-0.084159	0.716232	0.820527	0.820543	0.710786	0.689372	0.758376	0.721841	0.424227	1.000000

[14]

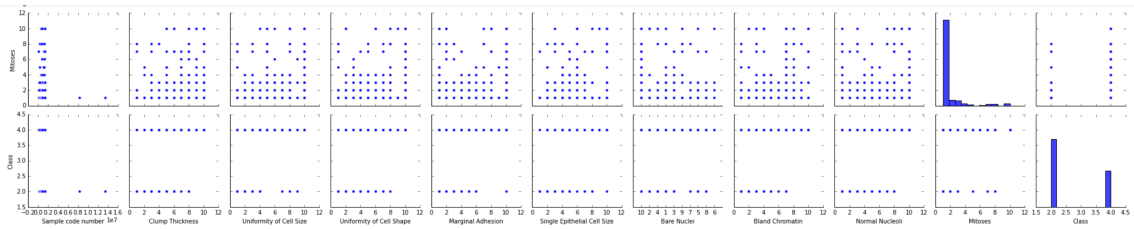


The above graph sns heat map used to find the relation of the attributes. The "sample code number" with all the attributes are not correlated, whereas it does not impact much on the dataset. The "Uniformity of Cell Size" is highly correlated with 91% wheras the "Mitoses" has the correlation of 35% to 45%

```
#plot the pair plot.
sns.pairplot(df, vars=['Sample code number','Clump Thickness','Uniformity of Cell Size','Uniformity of Cell Shape','Marginal Adhesion','Single Epithelial Ce
plt.show()
```



[15]



Feature Selection:

The dataset is having less number of attributes therefore the feature selection is not done.

7.1 FEATURE SELECTION

The Dataset is having a smaller number of Attributes of 10 Columns therefore feature selection is not implemented for the dataset.

8. MODEL TRAINING EVALUATION

Splitting the dataset in Target and feature

```
[16] y= df.iloc[:,10]
     y       #dftarget
```

```
0      2
1      2
2      2
3      2
4      4
      ..
693    2
694    2
695    4
696    4
697    4
Name: Class, Length: 674, dtype: int64
```

```
x=df.iloc[:,0:10]
x      #dffeature
```

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses
0	1002945	5	4	4	5	7	10	3	2	1
1	1015425	3	1	1	1	2	2	3	1	1
2	1016277	6	8	8	1	3	4	3	7	1
3	1017023	4	1	1	3	2	1	3	1	1
4	1017122	8	10	10	8	7	10	9	7	1
...	...	...	...	...	...	...	...	...	...	...
693	776715	3	1	1	1	3	2	1	1	1
694	841769	2	1	1	1	2	1	1	1	1
695	888820	5	10	10	3	7	3	8	10	2
696	897471	4	8	6	4	3	4	10	6	1
697	897471	4	8	8	5	4	5	10	4	1

674 rows × 10 columns

Split dataset into training set and test set

+ Code + Text

```
[18] #Creating the train and validation set
     x_train,x_valid,y_train, y_valid = train_test_split(x,y,random_state= 1,test_size=0.3)
```

```
[19] #Distribution in Training set
     x_train.value_counts(normalize=True)
```

```
Sample code number  Clump Thickness  Uniformity of Cell Size  Uniformity of Cell Shape  Marginal Adhesion  Single Epithelial Cell Size  Bare Nuclei  Bland Ch
63375               9                1                        2                         6                  4                            10           7
```

```
[19] 1214556          3          1          1          1          2          1          2
     1220330          1          1          1          1          2          1          3
     1219859          8         10          8          8          4          8          7
     1219525          8         10         10         10          5         10          8

     1072179         10          7          7          3          8          5          7
     1070935          3          1          1          1          1          1          2
                      1          1          3          1          2          1          1
     1070522          3          1          1          1          1          1          2
     13454352         1          1          3          1          2          1          2
     Length: 471, dtype: float64
```

```
#Distribution in Validation set
y_valid.value_counts(normalize=True)
```

```
2    0.640394
4    0.359606
Name: Class, dtype: float64
```

```
[21] #Shape of training set
     x_train.shape, y_train.shape

     ((471, 10), (471,))
```

```
[22] #Shape of Validation set
     x_valid.shape, y_valid.shape

     ((203, 10), (203,))
```

```
[23] #Creating the decision tree function
     classifier = DecisionTreeClassifier()
```

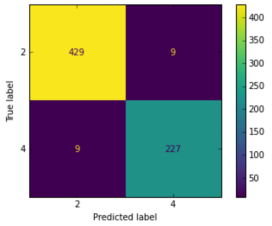
```
[24] classifier=classifier.fit(x_train,y_train)
```

```
[25] #Checking the validation score
     classifier.score=(x_valid,y_valid)
```

```
[26] #Predictions on validation set
     x_predict = classifier.predict(x_valid)
     metrics.accuracy_score(y_valid,x_predict)

     0.9113300492610837
```

The Above codes explains the splitting dataset in training, testing and accuracy score of 91.13% for decision tree model

```
[27] # plotting the confusion matrix
     plot_confusion_matrix(classifier,x, y)
     plt.show()

     /usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_confusion
       warnings.warn(msg, category=FutureWarning)
```



A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model.
For a binary classification problem, we would have a 2 x 2 matrix as shown below with 4 values:
True Positive (TP)
The predicted value matches the actual value
The actual value was positive, and the model predicted a positive value
True Negative (TN)
The predicted value matches the actual value
The actual value was negative, and the model predicted a negative value
False Positive (FP) – Type 1 error
The predicted value was falsely predicted
The actual value was negative, but the model predicted a positive value
Also known as the Type 1 error
False Negative (FN) – Type 2 error
The predicted value was falsely predicted
The actual value was positive, but the model predicted a negative value

Also known as the Type 2 error
The different values of the Confusion matrix would be as follows:

- True Positive (TP) = 429; meaning 429 positive class data points were correctly classified by the model
- True Negative (TN) = 227; meaning 227 negative class data points were correctly classified by the model
- False Positive (FP) = 09; meaning 09 negative class data points were incorrectly classified as belonging to the positive class by the model
- False Negative (FN) = 09; meaning 09 positive class data points were incorrectly classified as belonging to the negative class by the model

*Cross validation *

```
[28] model = DecisionTreeClassifier()
     # evaluate pipeline
     cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
     scores = cross_val_score(model, x, y, scoring='roc_auc', cv=cv, n_jobs=-1)
```

```
[29] scores
```

```
array([0.95643939, 0.92613636, 0.95643939, 0.97916667, 0.9555336 ,
       0.84683794, 0.97826087, 0.87796443, 0.93507752, 0.93507752,
       0.91666667, 0.95643939, 0.90530303, 0.9469697 , 0.98863636,
       0.97727273, 0.98863636, 0.87895257, 0.96511628, 0.94670543,
       0.95643939, 0.89583333, 0.92424242, 0.95833333, 0.91205534,
       0.94416996, 0.9451581 , 0.94416996, 0.96753876, 0.95833333])
```

Grid search k value for SMOTE oversampling classification

```
[30] # grid search k value for SMOTE oversampling for imbalanced classification
     from numpy import mean
     from sklearn.datasets import make_classification
     from sklearn.model_selection import cross_val_score
     from sklearn.model_selection import RepeatedStratifiedKFold
     from sklearn.tree import DecisionTreeClassifier
     from imblearn.pipeline import Pipeline
     from imblearn.over_sampling import SMOTE
     from imblearn.under_sampling import RandomUnderSampler
```

```
[31] #define dataset
     x, y = make_classification(n_samples=10000, n_features=2, n_redundant=0,
       n_clusters_per_class=1, weights=[0.99], flip_y=0, random_state=1)
```

```
[32] # values to evaluate
     k_values = [1, 2, 3, 4, 5, 6, 7]
     for k in k_values:
       # define pipeline
       model = DecisionTreeClassifier()
       over = SMOTE(sampling_strategy=0.1, k_neighbors=k)
       under = RandomUnderSampler(sampling_strategy=0.5)
       steps = [('over', over), ('under', under), ('model', model)]
       pipeline = Pipeline(steps=steps)
       # evaluate pipeline
       cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
       scores = cross_val_score(pipeline, x, y, scoring='roc_auc', cv=cv, n_jobs=-1)
       score = mean(scores)
       print('> k=%d, Mean ROC AUC: %.3f' % (k, score))
```

```
> k=1, Mean ROC AUC: 0.837
> k=2, Mean ROC AUC: 0.839
> k=3, Mean ROC AUC: 0.839
> k=4, Mean ROC AUC: 0.841
> k=5, Mean ROC AUC: 0.854
> k=6, Mean ROC AUC: 0.856
> k=7, Mean ROC AUC: 0.839
```

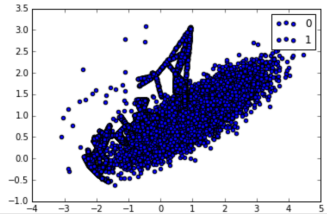
```
# borderline-SMOTE for imbalanced dataset
from collections import Counter
from sklearn.datasets import make_classification
from imblearn.over_sampling import BorderlineSMOTE
from matplotlib import pyplot
from numpy import where
```

```
[34] # define dataset
     x, y = make_classification(n_samples=10000, n_features=2, n_redundant=0,
       n_clusters_per_class=1, weights=[0.99], flip_y=0, random_state=1)
```

```
[35] # summarize class distribution
     counter = Counter(y)
     print(counter)
```

```
Counter({0: 9900, 1: 100})
```

```
[36] # transform the dataset
     oversample = BorderlineSMOTE()
     x, y = oversample.fit_resample(x, y)

[37] # summarize the new class distribution
     counter = Counter(y)
     print(counter)

     Counter({0: 9900, 1: 9900})

     # scatter plot of examples by class label
     for label, _ in counter.items():
       row_ix = where(y == label)[0]
       pyplot.scatter(x[row_ix, 0], x[row_ix, 1], label=str(label))
     pyplot.legend()
     pyplot.show()
```



The pre-processing and visualisation of the breast cancer dataset is explained above.

```
     ##############################Begining of MODEL2#############################################################
```

Support Vector machine

```
[39] # Splitting the dataset into the Training set and Test set
     from sklearn.model_selection import train_test_split
     x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 1)

[40] # Feature Scaling
     from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     x_train = sc.fit_transform(x_train)
     x_test = sc.transform(x_test)

[41] # Fitting Kernel SVM to the Training set
     from sklearn.svm import SVC
     classifier = SVC(kernel = 'linear', random_state = 0)
     classifier.fit(x_train, y_train)

     SVC(kernel='linear', random_state=0)

[42] # Predicting the Test set results
     y_pred = classifier.predict(x_test)

     # Making the Confusion Matrix
     from sklearn.metrics import confusion_matrix
     cm = confusion_matrix(y_test, y_pred)

[44] cm

     array([[2903,  109],
            [  65, 2863]])

[45] from sklearn.metrics import accuracy_score
     accuracy=accuracy_score(y_test,y_pred)

[46] accuracy

     0.9707070707070707

[47] # Applying Grid Search to find the best model and the best parameters
     from sklearn.model_selection import GridSearchCV
     parameters = [{'C': [1, 10, 100, 1000], 'kernel': ['linear']},
                   {'C': [1, 10, 100, 1000], 'kernel': ['rbf'], 'gamma': [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]}]
     grid_search = GridSearchCV(estimator = classifier,
                                param_grid = parameters,
                                scoring = 'accuracy',
                                cv = 10,
                                n_jobs = -1)
     grid_search = grid_search.fit(x_train, y_train)
```

```
[48] accuracy = grid_search.best_score_
```

```
[49] accuracy
```

```
0.9862193362193363
```

Grid search Parameter

```
[51] grid_search.best_params_
```

```
{'C': 1000, 'gamma': 0.9, 'kernel': 'rbf'}
```

```
[52] classifier = SVC(kernel = 'rbf', gamma=0.7)
     classifier.fit(x_train, y_train)
```

```
SVC(gamma=0.7)
```

```
# Predicting the Test set results
y_pred = classifier.predict(x_test)
```

```
[54] # Making the Confusion Matrix
     from sklearn.metrics import confusion_matrix
     cm = confusion_matrix(y_test, y_pred)
```

```
[55] cm
```

```
array([[2920,   92],
       [  33, 2895]])
```

```
[56] from sklearn.metrics import accuracy_score
     accuracy=accuracy_score(y_test,y_pred)
```

```
[57] accuracy
```

```
0.9789562289562289
```

```
[ ] ################################Begining of MODEL 3 ###################################################
    ################################DEEP LEARNING MODEL#####################################################
```

```
#Importing Keras Libraries#
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D, MaxPooling2D
from keras.utils import np_utils
from keras.datasets import mnist
from matplotlib import pyplot as plt
from sklearn.model_selection import GridSearchCV, StratifiedKFold
%matplotlib inline
```

```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
df['Class'] = labelencoder.fit_transform(df['Class'])
df
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
0	1002945	5	4	4	5	7	10	3	2	1	0
1	1015425	3	1	1	1	2	2	3	1	1	0
2	1016277	6	8	8	1	3	4	3	7	1	0
3	1017023	4	1	1	3	2	1	3	1	1	0
4	1017122	8	10	10	8	7	10	9	7	1	1
...	...	...	...	...	...	...	...	...	...	...	...
693	776715	3	1	1	1	3	2	1	1	1	0
694	841769	2	1	1	1	2	1	1	1	1	0
695	888820	5	10	10	3	7	3	8	10	2	1
696	897471	4	8	6	4	3	4	10	6	1	1

```
[60] 697  897471  4  8  8  5  4  5  10  4  1  1
674 rows × 11 columns
```

```
[61] x_class=df.drop(["Class"],axis=1)
     y_class=df["Class"]
```

```
[62] from sklearn.preprocessing import StandardScaler, MinMaxScaler
     #Normalise the data between 0 and 1
     norma=MinMaxScaler()
     x_class_scaled = norma.fit_transform(x_class)
     x_class_scaled
```

```
array([[0.07016441, 0.44444444, 0.33333333, ..., 0.22222222, 0.11111111,
        0.        ],
       [0.07109638, 0.22222222, 0.        , ..., 0.22222222, 0.        ,
        0.        ],
       [0.07116001, 0.55555556, 0.77777778, ..., 0.22222222, 0.66666667,
        0.        ],
       ...,
       [0.06164188, 0.44444444, 1.        , ..., 0.77777778, 1.        ,
        0.11111111],
       [0.06228791, 0.33333333, 0.77777778, ..., 1.        , 0.55555556,
        0.        ],
       [0.06228791, 0.33333333, 0.77777778, ..., 1.        , 0.33333333,
        0.        ]])
```

```
[63] #Creating the train and validation set
     x_train,x_valid,y_train, y_valid = train_test_split(x_class_scaled,y_class,random_state= 1,test_size=0.3)
```

```
[64] def classification_model():
         model = Sequential()
         model.add(Dense(10, input_dim= 10, activation='relu'))
         model.add(Dense(20, activation='relu'))
         model.add(Dropout(0.2))
         model.add(Dense(5, activation='relu'))
         model.add(Dropout(0.2))
         model.add(Dense(1,activation='sigmoid'))
         model.compile(loss='binary_crossentropy',
                   optimizer='adam',
                   metrics= ['accuracy'])
         return model
```

```
[65] from keras.wrappers.scikit_learn import KerasClassifier
     estimator = KerasClassifier(build_fn=classification_model, epochs=100, batch_size=5, verbose=0)
     kfold = StratifiedKFold(n_splits=10, shuffle=True)
     results = cross_val_score(estimator, x_train, y_train, cv=kfold)
     print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: DeprecationWarning: KerasClassifier is deprecated, use Sci-Keras (https://github.com/adriangb

Baseline: 97.23% (2.14%)
```

```
results
```

```
array([1.        , 1.        , 0.93617022, 0.97872341, 0.93617022,
       0.97872341, 0.95744681, 0.97872341, 0.97872341, 0.97872341])
```

```
[ ] ##############################END of MODEL 3 ##################################################
    ###############################################################################################
```

9. RESULTS AND DISCUSSION

The table captures the accuracy obtained from various models. The figures & Python code snippets are embedded in the above sections.

#	Name of the model	Accuracy
Model1	Decision Tree Classifier	91.13%
Model2	Support Vector Machine	97.07% After Tuning the Accuracy is 98.62%
Model3	Artificial Neural Network	97.45%

The table captures the True Positive, True Negative, False Positive and False Negative for various models. The figures & Python code snippets are embedded in the above sections.

	True Positive	True Negative	False Positive	False Negative
Decision Tree Classifier	429	227	9	9
Support Vector Machine	2903	2863	109	65

10. CONCLUSION, RECOMMENDATIONS AND FUTURE WORK

The report explained about the Wisconsin Breast Cancer Database to predict the breast cancer using various machine and deep learning algorithms. This coursework uses decision tree, support vector machine and neural network to predict the cancer. The support vector machine

gives the accuracy of 97.07% before tuning and After Tuning the Accuracy is 98.62%.Therefore the support vector machine gives the best model to predict cancer for the breast cancer dataset. Also, this turned out to be a pretty decent classifier for our dataset considering the relatively larger number of true positive and true negative values.

11. REFERENCES

[1] Dhasaradhan K, Jaichandran R, Shunmuganathan KL, Usha Kiruthika S, Rajaprakash S. Hybrid machine learning model using decision tree and support vector machine for diabetes identification. InData Engineering and Intelligent Computing 2021 (pp. 293-305). Springer, Singapore

[2]https://www.analyticsvidhya.com/blog/2016/10/tutorial-optimizing-neural-networks-using-keras-with-image-recognition-case-study

[3] https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94

[4] Telsang VA, Hegde K. Breast Cancer Prediction Analysis using Machine Learning Algorithms. In2020 International Conference on Communication, Computing and Industry 4.0 (C2I4) 2020 Dec 17 (pp. 1-5). IEEE

[5] Zhang J. Selecting typical instances in instance-based learning. In Machine learning proceedings 1992 1992 Jan 1 (pp. 470-479). Morgan Kaufmann

[6] https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/