

# **HIA302: HEALTH DATA ANALYTICS**

## **Group Project**

### **Data Analysis and Diagnosis Prediction of Breast Cancer Dataset**

Group members:

1. Lee Zi Sheng
2. Tee Kar Men
3. Nuraini Muhammad Naim

## PROJECT BACKGROUND

This project, which is part of the requirement for the HIA303: Health Data Analytics module, aims to perform pre-processing and data analysis using Python and to find the best supervised learning technique to model this dataset.

We have chosen three machine learning models and have distributed for the focus on the project as the following:


- Tee Kar Men: K-Nearest Neighbour
- Nuraini Muhammad Naim: Logistic Regression
- Lee Zi Sheng: Support Vector Machine

However, for the interest of the narrative of the report writing, we decided to not discuss different machine learning models separately in this report.

The compilation of files related to this project can be accessed at <https://github.com/Karmen-Tee/Breast-Cancer>.

## DATA PREPARATION PROCESS

The dataset used for this project was the Breast Cancer Wisconsin (Original) Data Set that is available at <https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28original%29>.




**UCI** Machine Learning Repository  
Center for Machine Learning and Intelligent Systems

Check out the [beta version](#) of the new UCI Machine Learning Repository we are currently testing! [Contact us](#) if you have any feedback.

### Breast Cancer Wisconsin (Original) Data Set

Download: [Data Folder](#), [Data Set Description](#)

**Abstract:** Original Wisconsin Breast Cancer Database



<b>Data Set Characteristics:</b>	Multivariate	<b>Number of Instances:</b>	699	<b>Area:</b>	Life
<b>Attribute Characteristics:</b>	Integer	<b>Number of Attributes:</b>	10	<b>Date Donated</b>	1992-07-15
<b>Associated Tasks:</b>	Classification	<b>Missing Values?</b>	Yes	<b>Number of Web Hits:</b>	765553

## Overview of Dataset

Firstly, we assessed the dataset by checking the available data points and the data types to ensure that the data is ready to be used for statistical analysis. The dataset contains 699 instances and 11 attributes, including 10 input variables and one output variable (Figure 1). The output variable is known as the 'Class' attribute which is categorised into 2 if benign tumour and 4 if malignant tumour. All attributes are integer values except 'Bare\_nuclei' attribute. However, machine learning algorithms expect numeric input thus conversion to numeric data types was performed (Figure 1). For our interest, the 'class' attribute will be addressed as target variable, while the remaining attributes will be the features which help predict our target variable.

<pre>df.info()  &lt;class 'pandas.core.frame.DataFrame'&gt; RangeIndex: 699 entries, 0 to 698 Data columns (total 11 columns): #   Column                Non-Null Count  Dtype ---  --- 0   ID                     699 non-null    int64 1   Clump_thickness        699 non-null    int64 2   Uniformity_size        699 non-null    int64 3   Uniformity_shape       699 non-null    int64 4   Marginal_adhesion      699 non-null    int64 5   Single_epithelial_cell_size 699 non-null    int64 6   Bare_nuclei            699 non-null    object 7   Bland_chromatin        699 non-null    int64 8   Normal_nucleoli        699 non-null    int64 9   mitoses                699 non-null    int64 10  class                  699 non-null    int64 dtypes: int64(10), object(1) memory usage: 60.2+ KB</pre>	<pre>df['Bare_nuclei'] = pd.to_numeric(df['Bare_nuclei'],errors='coerce') df.info()  &lt;class 'pandas.core.frame.DataFrame'&gt; RangeIndex: 699 entries, 0 to 698 Data columns (total 11 columns): #   Column                Non-Null Count  Dtype ---  --- 0   ID                     699 non-null    int64 1   Clump_thickness        699 non-null    int64 2   Uniformity_size        699 non-null    int64 3   Uniformity_shape       699 non-null    int64 4   Marginal_adhesion      699 non-null    int64 5   Single_epithelial_cell_size 699 non-null    int64 6   Bare_nuclei            683 non-null    float64 7   Bland_chromatin        699 non-null    int64 8   Normal_nucleoli        699 non-null    int64 9   mitoses                699 non-null    int64 10  class                  699 non-null    int64 dtypes: float64(1), int64(10) memory usage: 60.2 KB</pre>
--	--

**Figure 1:** Summary of the original dataset (*left*) and after 'Bare\_nuclei' was converted from object to numeric (*right*).

## Basic Exploratory Data Analysis

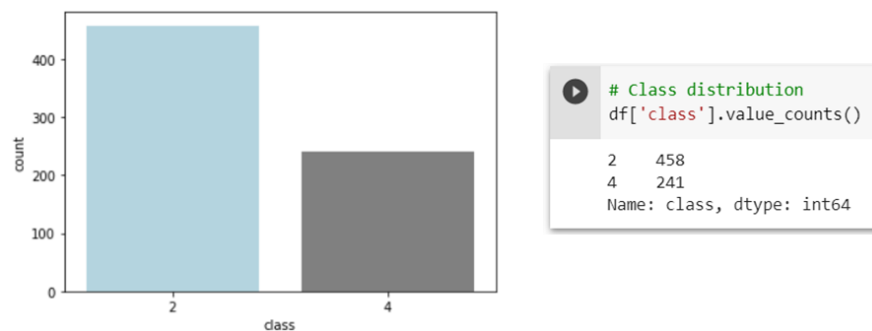
We performed a statistical analysis to view the statistical information of the dataset by using the 'describe' function (Figure 2). As the domain of each numeric feature (except ID number) ranges from 1 – 10, the dataset should not include any zero value and any zero values will be defined as missing values. From the minimum values of each feature, we confirmed that there are no zero values in the dataset (Fig. 2).

```
# Compute and view summary statistics for numerical values.
# Transpose for better view.
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
ID	699.0	1.071704e+06	617095.729819	61634.0	870688.5	1171710.0	1238298.0	13454352.0
Clump_thickness	699.0	4.417740e+00	2.815741	1.0	2.0	4.0	6.0	10.0
Uniformity_size	699.0	3.134478e+00	3.051459	1.0	1.0	1.0	5.0	10.0
Uniformity_shape	699.0	3.207439e+00	2.971913	1.0	1.0	1.0	5.0	10.0
Marginal_adhesion	699.0	2.806867e+00	2.855379	1.0	1.0	1.0	4.0	10.0
Single_epithelial_cell_size	699.0	3.216023e+00	2.214300	1.0	2.0	2.0	4.0	10.0
Bare_nuclei	683.0	3.544656e+00	3.643857	1.0	1.0	1.0	6.0	10.0
Bland_chromatin	699.0	3.437768e+00	2.438364	1.0	2.0	3.0	5.0	10.0
Normal_nucleoli	699.0	2.866953e+00	3.053634	1.0	1.0	1.0	4.0	10.0
mitoses	699.0	1.589413e+00	1.715078	1.0	1.0	1.0	1.0	10.0
class	699.0	2.689557e+00	0.951273	2.0	2.0	2.0	4.0	4.0

**Figure 2:** Summary statistics of dataset.

The target variable has two labels, where 458 (65.5%) instances for Benign and 241 (34.5%) instances for Malignant (Figure 3). We calculated the mean of each feature for each label and found that mean of each feature for benign tumours are greater than malignant tumours (Figure 4).



**Figure 3:** Class distribution.

```
df.groupby('class').mean()
```

	ID	Clump_thickness	Uniformity_size	Uniformity_shape	Marginal_adhesion	Single_epithelial_cell_size	Bare_nuclei	Bland_chromatin	Normal_nucleoli	mitoses
class										
2	1.107591e+06	2.956332	1.325328	1.443231	1.364629	2.120087	1.346847	2.100437	1.290393	1.063319
4	1.003505e+06	7.195021	6.572614	6.560166	5.547718	5.298755	7.627615	5.979253	5.863071	2.589212

**Figure 4:** Mean of each attribute after grouping based on class attribute.

The relationship between features (including target variable) was explored and visualised using the 'pairplot' function available from the Seaborn library in Fig. 5. There was no clear pattern observed from the scatter plots. The distribution patterns in diagonal illustrated that each label of target variable (benign or malignant) corresponding to each feature have relatively distinctive separations. There was minimal overlapping between the distribution of two labels of target variables for all features.



**Figure 5:** Pair plot of features based on label classes, with *blue* representing benign tumour and *red* representing malignant tumour.

## Correlation Matrix

The correlation between features was summarised (Figure 6) and visualised using a correlation heatmap with the degree of correlation highlighted by colour palette (Figure 7). We studied the strength of the correlation between target variable and each feature, where the highest correlation coefficient is between target variable and Bare Nuclei, followed by Uniformity of Cell Shape, Uniformity of Cell Size, Bland Chromatin, Clump thickness, Normal Nucleoli, Marginal Adhesion, Single Epithelial Cell Size, and lowest between Mitoses (Figure 8).

```
# Correlations between features.
correlations=df.corr()
correlations
```

	ID	Clump_thickness	Uniformity_size	Uniformity_shape	Marginal_adhesion	Single_epithelial_cell_size	Bare_nuclei	Bland_chromatin	Normal_nucleoli	mitoses	class
ID	1.000000	-0.055308	-0.041603	-0.041576	-0.064878	-0.045528	-0.099248	-0.060051	-0.052072	-0.034901	-0.080226
Clump_thickness	-0.055308	1.000000	0.644913	0.654589	0.486356	0.521816	0.593091	0.558428	0.535835	0.350034	0.716001
Uniformity_size	-0.041603	0.644913	1.000000	0.906882	0.705582	0.751799	0.691709	0.755721	0.722865	0.458693	0.817904
Uniformity_shape	-0.041576	0.654589	0.906882	1.000000	0.683079	0.719668	0.713878	0.735948	0.719446	0.438911	0.818934
Marginal_adhesion	-0.064878	0.486356	0.705582	0.683079	1.000000	0.599599	0.670648	0.666715	0.603352	0.417633	0.696800
Single_epithelial_cell_size	-0.045528	0.521816	0.751799	0.719668	0.599599	1.000000	0.585716	0.616102	0.628881	0.479101	0.682785
Bare_nuclei	-0.099248	0.593091	0.691709	0.713878	0.670648	0.585716	1.000000	0.680615	0.584280	0.339210	0.822696
Bland_chromatin	-0.060051	0.558428	0.755721	0.735948	0.666715	0.616102	0.680615	1.000000	0.665878	0.344169	0.756616
Normal_nucleoli	-0.052072	0.535835	0.722865	0.719446	0.603352	0.628881	0.584280	0.665878	1.000000	0.428336	0.712244
mitoses	-0.034901	0.350034	0.458693	0.438911	0.417633	0.479101	0.339210	0.344169	0.428336	1.000000	0.423170
class	-0.080226	0.716001	0.817904	0.818934	0.696800	0.682785	0.822696	0.756616	0.712244	0.423170	1.000000

Figure 6: Correlation matrix with calculated coefficients between features.

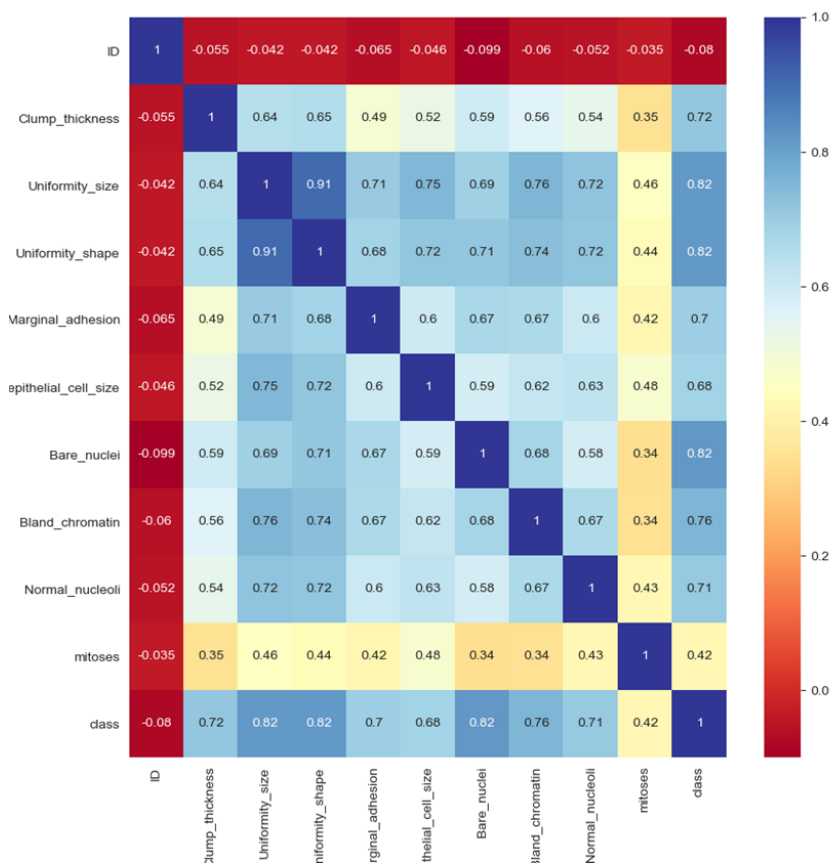


Figure 7: Correlation heatmap visualisation.

```

corr=df.corr()
corr['class'].sort_values(ascending = False)

class            1.000000
Bare_nuclei       0.822696
Uniformity_shape  0.818934
Uniformity_size   0.817904
Bland_chromatin   0.756616
Clump_thickness   0.716001
Normal_nucleoli   0.712244
Marginal_adhesion 0.696800
Single_epithelial_cell_size 0.682785
mitoses           0.423170
ID               -0.080226
Name: class, dtype: float64

```

**Figure 8.** Correlation coefficients between target variable and features where values sorted in descending manner.

### Data Cleaning and Transformation

Firstly, we removed the ID number. There are 16 instances with missing values in feature 'Bare\_nuclei' and all missing values are imputed using k-nearest neighbours (k-NN) algorithm by identifying neighbouring points with the Euclidean distance matrix (Figure 8). More information on the methodology of k-NN will be discussed later in the Machine Learning Models section.

```

# Print missing values.
df.isnull().sum()

Clump_thickness      0
Uniformity_size      0
Uniformity_shape     0
Marginal_adhesion    0
Single_epithelial_cell_size 0
Bare_nuclei          16
Bland_chromatin      0
Normal_nucleoli      0
mitoses              0
class                0
dtype: int64

```

```

# Check missing values after imputation.
df.isnull().sum()

Clump_thickness      0
Uniformity_size      0
Uniformity_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

```

**Figure 8.** List of missing values before imputation (*left*) and after imputation (*right*).

## Data Encoding and Transformation

The target 'class' variable is a categorical data containing label values of 2 for benign and 4 for malignant tumours. Even though 'class' is a numeric variable, we performed data encoding to convert each category into digital signals where 0 is Benign and 1 is Malignant using the 'map' function (Figure 9).

```
# Mapping Benign as 0 and Maglinant as 1.
df['class']=df['class'].map({2:0,4:1}).astype(int)
print(df['class'].values)

[0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 1 0 1 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 1
 0 1 1 0 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 0 0 1 0 1
 1 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 0 1 0
 1 1 1 0 0 0 1 0 0 0 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0
 0 1 0 1 1 0 0 1 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1 1 1 0 1 0 1 0 0 0 1 1
 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 1 1 1 0 0 1 0 0 1
 1 1 1 0 1 1 0 1 1 1 0 1 0 0 1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 1 0 1 1 1 0 0 0
 0 1 1 1 1 1 0 1 1 1 0 1 0 1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 1 1 0 0 1 1 0 1
 0 0 0 1 1 0 1 0 1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 0 0 1 0 0 1 0 0 1 1 1 1 0 0
 1 1 0 1 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0 1 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0
 0 0 0 0 0 1 0 1 0 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0
 0 0 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0
 0 1 1 0 0 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0
 0 1 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 0 1 0 1 1
 1 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 0 1 0 0 1 0 0 0 0 0 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 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1]
```

**Figure 9:** Data encoding of target variable, where 0 if benign and 1 if malignant.

## Train-test Split

The dataset was split into training and test sets to evaluate the performance of machine learning algorithms for predictive modelling. It is a technique commonly used in supervised learning algorithms. The training set is used to train the selected machine algorithm with known inputs and outputs while the test set is used to evaluate and estimate error rate of the trained machine learning model.

We first stored the input and out elements in separate variables, then split the dataset in 4:1 ratio, where training set contains 559 (80%) instances and test set with 140 (20%) instances (Figure 10)

```
# Split input and output elements.
# Input elements.
X = df[['Clump_thickness', 'Uniformity_size', 'Uniformity_shape', 'Marginal_adhesion',
        'Single_epithelial_cell_size', 'Bare_nuclei', 'Bland_chromatin', 'Normal_nucleoli', 'mitoses']]
# Output element.
y = df['class']

# Split dataset into 2 parts: training set and test set
# test_size sets as 0.2 means 80% for training set and 20% for test set.
# Use the random_state parameter for reproducible results.
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)
print (X_train.shape, X_test.shape)

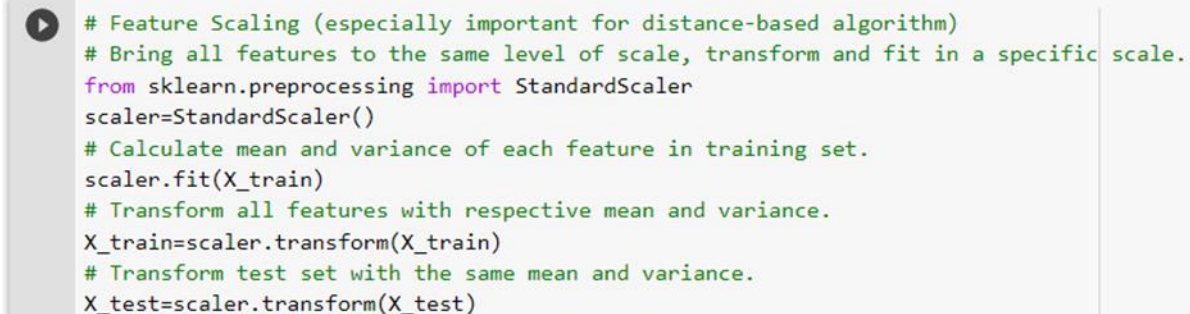
(559, 9) (140, 9)
```

**Figure 10:** Features and target variable were separated and the whole dataset was split into training and test sets.



## Feature Scaling

As the dataset may contain features varying degrees of magnitude, range and units, it is important to scale all features to the same level prior to modelling. We utilised the StandardScaler from Scikit-learn library to standardise both training and test sets (Figure 11).

A screenshot of a code editor showing Python code for feature scaling. The code is as follows:

```
# Feature Scaling (especially important for distance-based algorithm)
# Bring all features to the same level of scale, transform and fit in a specific scale.
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
# Calculate mean and variance of each feature in training set.
scaler.fit(X_train)
# Transform all features with respective mean and variance.
X_train=scaler.transform(X_train)
# Transform test set with the same mean and variance.
X_test=scaler.transform(X_test)
```

**Figure 11.** Python codes for feature scaling.

## MACHINE LEARNING TECHNIQUES

We have chosen to focus on the following three machine learning techniques.

### 1. K-Nearest Neighbours Algorithm

The K-Nearest Neighbours (KNN) is one of the earliest and simplest algorithms used for classification. It can be used to solve both classification and regression problems. KNN is a non-parametric algorithm with no assumption about the statistical distribution of data. Furthermore, KNN is referred as a lazy algorithm where it does not learn much during the training process, rather perform learning when unknown object is introduced for prediction. For classification, KNN classifies unknown object based on the majority vote of nearest neighbouring points using a distance matrix. Therefore, the nearest neighbouring points and distance matrix are the most important considerations for KNN classifier. The number of neighbouring points is defined as the k-value. The predictive accuracy of the classifier is highly dependent on the choice of k-value and the selection of best k-value can be done in several methods. A simple and direct way is to run the algorithm with different k-values to determine and select which achieved the best performance. Although there are different choices of distance metrics available, most KNN classifiers use the standard Euclidean distance metric. As KNN is a distance-based algorithm, the Euclidean distance is used to measure the similarity between neighbouring points and unknown object, where higher similarity indicates higher likelihood to be the same class.

*Reason for selection: it is a simple and efficient form of machine learning that is suitable for the prediction of diagnosis.*

### 2. Logistic regression technique

Regression algorithm technique is one of the most commonly used algorithms to predict an outcome or to find a cause-and-effect relationship between dependent variable and independent variable in a dataset by determining the best fit line that assess through all the



data points with minimal distance. There are several types of regression modelling but the most prominent techniques are the linear regression and logistic regression.

Logistic regression may further be classified into the following categories:

- Binomial: There are only two possibilities for the dependent variable e.g. yes and no.
- Multinomial: Dependent variable may have three or more possibilities but with no quantitative significance e.g. "Type 1", "Type 2", "Type 3".
- Ordinal: Dependent variable may have three or more possibilities with quantitative significance e.g. "Poor", "Good", "Very Good", "Excellent".

*Reason for selection: Logistic regression is commonly used for predictive analysis, hence an appropriate selection for the objective of this project i.e. diagnosis prediction of breast cancer.*

### 3. Support Vector Machine

Support Vector Machines (SVM) are supervised machine learning for regression and classification purposes. SVM can separate the data by identifying the hyper-plane or line. The hyper-plane maximises the distance or margin between the nearest data point for better classification. SVM is able to solve the problem for non-linear datasets, by applying the kernel trick it is able to transform 2D datasets into 3D datasets for better segregation. Lastly SVM ignores the outlier of the variables. Several features are important while applying the SVM model which is a kernel, C and gamma. The function of the kernel is helping to solve complex datasets with high dimensions. Several types of Kernel function such as linear, polynomial and RBF. Next, C is a hypermeter in the model to control the error, low C means low error and vice versa. However, low C does not necessarily provide better results, the value of C depends on the features of the datasets. Finally, there is Gamma, which only applies to the RBF kernel. Gamma is used to decide the curvature for the decision boundary to classify the data.

*Reason for selection: This model is preferred by many because it is able to produce significant accuracy with less computation power.*

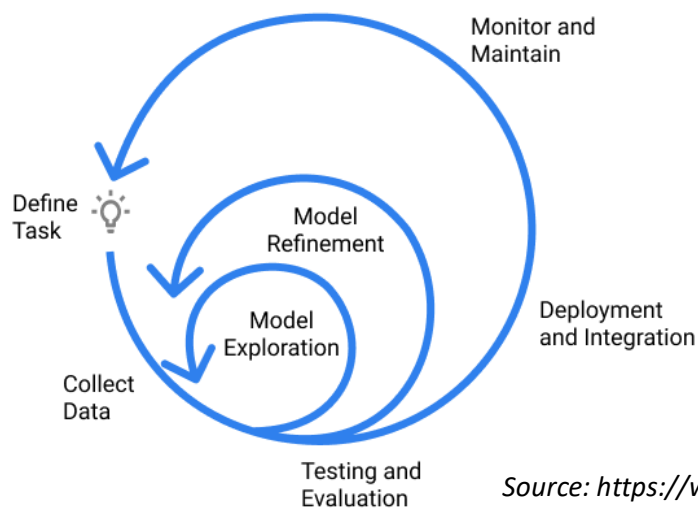
We have used the following codes in applying the machine learning model to the dataset:

Objectives	Codes																																	
<p>Select the best K value for KNN Model using Receiver Operating Characteristic (ROC).</p> <p>Interpret through ROC curves and area under the curve (AUC)</p> <p>Calculating accuracy for each K ranging from 1 to 100</p> <p>Create a pandas.dataframe (table) using the dictionary.</p>	<pre>[ ] # Check for the best K value by getting Receiver Operating Characteristic (ROC) # Interpret through ROC curves and area under the curve (AUC) # Calculating accuracy for each K ranging from 1 to 100 tt = {} il = [] ac=[]  for i in range(1,100):     knn = KNeighborsClassifier(n_neighbors=i)     knn.fit(X_train,y_train)     y_pred = knn.predict(X_test)      # Create a list of K values range 1 to 100     il.append(i)     # Create a list of ROC AUC for each K values     ac.append(sklearn.metrics.roc_auc_score(y_test,y_pred))      # Define as dictionary (key-value pairs)     tt.update({'K':il})     tt.update({'ROC_AUC':ac})</pre> <pre>[ ] #Select best K value for KNN Model # Create a pandas.dataframe (table) of attributes stored in dictionary. vv = pd.DataFrame(tt) vv.sort_values('ROC_AUC',ascending=False,inplace=True,ignore_index=True) vv.head(10) # Selecting K value with the best ROC accuracy</pre> <table><thead><tr><th></th><th>K</th><th>ROC_AUC</th></tr></thead><tbody><tr><td>0</td><td>21</td><td>0.979144</td></tr><tr><td>1</td><td>12</td><td>0.979144</td></tr><tr><td>2</td><td>20</td><td>0.979144</td></tr><tr><td>3</td><td>19</td><td>0.979144</td></tr><tr><td>4</td><td>18</td><td>0.979144</td></tr><tr><td>5</td><td>17</td><td>0.979144</td></tr><tr><td>6</td><td>16</td><td>0.979144</td></tr><tr><td>7</td><td>15</td><td>0.979144</td></tr><tr><td>8</td><td>14</td><td>0.979144</td></tr><tr><td>9</td><td>11</td><td>0.979144</td></tr></tbody></table>		K	ROC_AUC	0	21	0.979144	1	12	0.979144	2	20	0.979144	3	19	0.979144	4	18	0.979144	5	17	0.979144	6	16	0.979144	7	15	0.979144	8	14	0.979144	9	11	0.979144
	K	ROC_AUC																																
0	21	0.979144																																
1	12	0.979144																																
2	20	0.979144																																
3	19	0.979144																																
4	18	0.979144																																
5	17	0.979144																																
6	16	0.979144																																
7	15	0.979144																																
8	14	0.979144																																
9	11	0.979144																																
<p>Select the best condition of the model for SVM Model</p>	<pre>[ ] #Select the best condition of the model for SVM Model from sklearn.model_selection import GridSearchCV param_grid = {'C':[0.1,1,10,100,1000], 'gamma':[1,0.1,0.01,0.001,0.0001], 'kernel':['rbf']} grid = GridSearchCV(SVC(),param_grid,verbose = 4) grid.fit(X_train,y_train) grid.best_params_ grid.best_estimator_ grid_predictions = grid.predict(X_test) cmG = confusion_matrix(y_test,grid_predictions) sns.heatmap(cmG, annot=True) print(classification_report(y_test,grid_predictions))</pre> <pre>[ ] grid.best_params_</pre>																																	

Objectives	Codes
Run the ML algorithm (KNN, LR and SVM)	<pre>[ ] #KNN from sklearn.neighbors import KNeighborsClassifier classifier=KNeighborsClassifier(n_neighbors=21) classifier.fit(X_train,y_train) y_pred_K=classifier.predict(X_test)  #LR from sklearn.linear_model import LogisticRegression LR_model = LogisticRegression() LR_model.fit(X_train,y_train) y_pred_L =LR_model.predict(X_test)  #SVM from sklearn.svm import SVC svm_model = SVC(C=1, gamma=0.01, kernel='rbf',probability=True) svm_model.fit(X_train, y_train) y_pred_S=svm_model.predict(X_test)  #Prediction probability pred_probk = classifier.predict_proba(X_test) pred_probl = LR_model.predict_proba(X_test) pred_probs = svm_model.predict_proba(X_test)</pre>

Following the application of these models, we then proceeded with the evaluation of each model to judge its performance in terms of their accuracy and reliability. The evaluation process is also useful in a machine learning development lifecycle as the outcome can be used to explore and further refine the model.

## Machine Learning Development Lifecycle



Source: <https://www.jeremyjordan.me/ml-projects-guide/>

We have chosen the following evaluation methods to compare the performance of the three different algorithms:

- Confusion matrix
- Classification report (Precision, Recall, F1 Score)
- Accuracy score
- Receiver Operating Characteristics (ROC) curve

### Confusion matrix

Confusion matrix is also known as the error matrix that visualises the predicted results versus the ground-truth labels.

		Actual Class	
		Cat	Non-Cat
Predicted Class	Cat	90	60
	Non-Cat	10	940

**Figure 12:** Example of a confusion matrix. (Source: <https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce>)

There are four possible outcomes based on this matrix:

- True positive: Predicted to be true and found to be true in reality
- True negative: Predicted to be false and found to be false in reality
- False positive: Predicted to be true but found to be false in reality. Also known as Type I error.
- False negative: Predicted to be false but found to be true in reality. Also known as Type II error.

### Classification report (Precision, Recall, F1 score)

Precision is the number of correct positive results divided by the number of positive results predicted by the classifier.

$$\text{Precision} = \text{True positive} / (\text{True Positive} + \text{False Positive})$$

Recall is the number of correct positive results divided by the total samples that should have been identified as positive.

$$\text{Recall} = \text{True positive} / (\text{True Positive} + \text{False Negative})$$

F1 score is combination of precision and recall. A high F1 indicates high precision and high recall, presenting a good balance between precision and recall. However, with low F1, it is unclear whether there is low precision or low recall.

$$\text{F1} = 2 / [ (1/\text{precision}) + (1/\text{recall}) ]$$

or

$$\text{F1} = 2 * ( (\text{precision} * \text{recall}) / (\text{precision} + \text{recall}) )$$

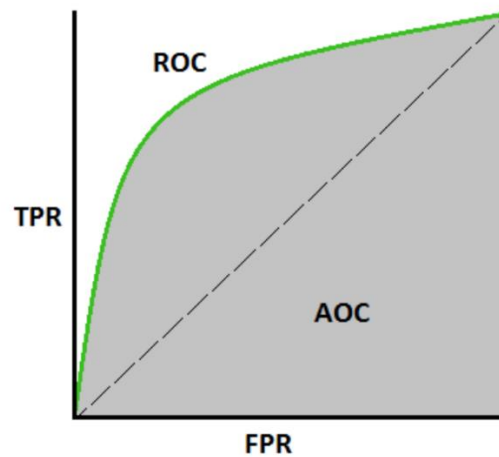
### Accuracy score

Accuracy score calculates the ratio between the number of correct predictions against all predictions made. It measures the probability of a prediction being made correctly.

$$\text{Accuracy} = (\text{True positive} + \text{True negative}) / \text{total prediction}$$

### Receiver Operating Characteristics (ROC) curve

The ROC curve shows the true positive rate against false positive rate for various threshold values.



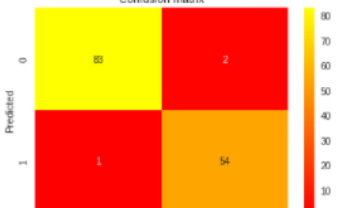
**Figure 13:** Relation between True Positive Rate (TPR), False Positive Rate (FPR), Receiver Operating Characteristics (ROC) curve and Area Under ROC Curve (AUC) (grey area). (Source: <https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce>)

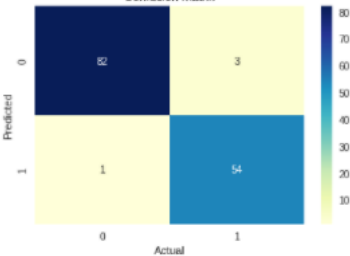
Confusion Matrix		Target			
		Positive	Negative		
Model	Positive	a	b	Positive Predictive Value	$a/(a+b)$
	Negative	c	d	Negative Predictive Value	$d/(c+d)$
		Sensitivity	Specificity	Accuracy = $(a+d)/(a+b+c+d)$	
		$a/(a+c)$	$d/(b+d)$		

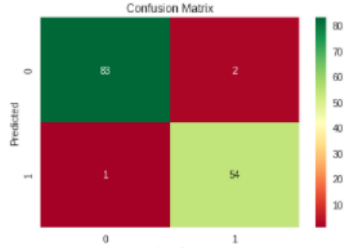
**Figure 14:** Summary of performance evaluations. (Source: <https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/>)

## EVALUATION RESULTS

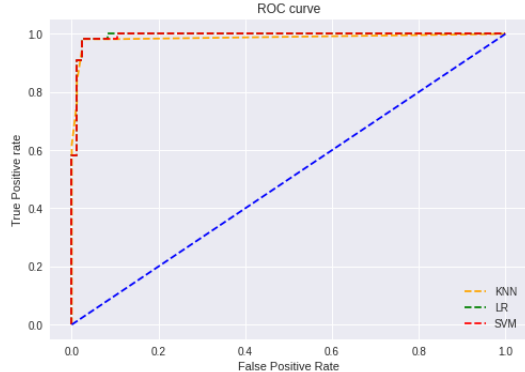
### Confusion matrix, classification report & accuracy score

K-NEAREST NEIGHBOUR																															
INPUT	OUTPUT																														
<pre>[ ] #KNN from sklearn.metrics import classification_report, confusion_matrix, accuracy_score  results=confusion_matrix(y_test,y_pred_K) print('Confusion matrix') print(results)  results1=classification_report(y_test,y_pred_K) print('Classification report') print(results1)  results2=accuracy_score(y_test,y_pred_K) print('Accuracy score') print(results2)  svm = sns.heatmap(pd.DataFrame(results), annot=True, cmap='autumn',fmt='d') plt.title('Confusion Matrix',y=1.1) plt.xlabel('Actual') plt.ylabel('Predicted')</pre>	<div><div><div>Confusion matrix</div><div><div>[[83  2]</div><div>[ 1 54]]</div></div></div><div><div>Classification report</div><div><table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>0.99</td><td>0.98</td><td>0.98</td><td>85</td></tr><tr><td>1</td><td>0.96</td><td>0.98</td><td>0.97</td><td>55</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.98</td><td>140</td></tr><tr><td>macro avg</td><td>0.98</td><td>0.98</td><td>0.98</td><td>140</td></tr><tr><td>weighted avg</td><td>0.98</td><td>0.98</td><td>0.98</td><td>140</td></tr></tbody></table></div></div></div> <div><div>Accuracy score</div><div>0.9785714285714285</div><div>Text(33.0, 0.5, 'Predicted')</div></div> <div><div>Confusion Matrix</div><div></div></div>		precision	recall	f1-score	support	0	0.99	0.98	0.98	85	1	0.96	0.98	0.97	55	accuracy			0.98	140	macro avg	0.98	0.98	0.98	140	weighted avg	0.98	0.98	0.98	140
	precision	recall	f1-score	support																											
0	0.99	0.98	0.98	85																											
1	0.96	0.98	0.97	55																											
accuracy			0.98	140																											
macro avg	0.98	0.98	0.98	140																											
weighted avg	0.98	0.98	0.98	140																											

LOGISTIC REGRESSION																															
INPUT	OUTPUT																														
<pre>[ ] #LR results10=confusion_matrix(y_test,y_pred_L) print('Confusion Matrix') print(results10)  results11=accuracy_score(y_test,y_pred_L) print('Accuracy Score') print(results11)  results12=classification_report(y_test, y_pred_L) print('Classification Report') print(results12)  svm = sns.heatmap(pd.DataFrame(results10), annot=True, cmap='YlGnBu',fmt='d') plt.title('Confusion Matrix',y=1.1) plt.xlabel('Actual') plt.ylabel('Predicted')</pre>	<div><div><div>Confusion Matrix</div><div>[[82 3] [ 1 54]]</div><div>Accuracy Score 0.9714285714285714</div><div>Classification Report</div><div><table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>0.99</td><td>0.96</td><td>0.98</td><td>85</td></tr><tr><td>1</td><td>0.95</td><td>0.98</td><td>0.96</td><td>55</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.97</td><td>140</td></tr><tr><td>macro avg</td><td>0.97</td><td>0.97</td><td>0.97</td><td>140</td></tr><tr><td>weighted avg</td><td>0.97</td><td>0.97</td><td>0.97</td><td>140</td></tr></tbody></table></div></div><div><div>Text(33.0, 0.5, 'Predicted')</div><div><div>Confusion Matrix</div></div></div></div>		precision	recall	f1-score	support	0	0.99	0.96	0.98	85	1	0.95	0.98	0.96	55	accuracy			0.97	140	macro avg	0.97	0.97	0.97	140	weighted avg	0.97	0.97	0.97	140
	precision	recall	f1-score	support																											
0	0.99	0.96	0.98	85																											
1	0.95	0.98	0.96	55																											
accuracy			0.97	140																											
macro avg	0.97	0.97	0.97	140																											
weighted avg	0.97	0.97	0.97	140																											

SUPPORT VECTOR MACHINE																															
INPUT	OUTPUT																														
<pre>[ ] #SVM results3=confusion_matrix(y_test,y_pred_5) print('Confusion Matrix') print(results3)  results4=accuracy_score(y_test,y_pred_5) print('Accuracy Score') print(results4)  results5=classification_report(y_test, y_pred_5) print('Classification Report') print(results5)  svm = sns.heatmap(pd.DataFrame(results3), annot=True, cmap='RdYlGn',fmt='d') plt.title('Confusion Matrix',y=1.1) plt.xlabel('Actual') plt.ylabel('Predicted')</pre>	<p>Confusion Matrix</p> <pre>[[83  2]  [ 1 54]]</pre> <p>Accuracy Score 0.9785714285714285</p> <p>Classification Report</p> <table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>0.99</td><td>0.98</td><td>0.98</td><td>85</td></tr><tr><td>1</td><td>0.96</td><td>0.98</td><td>0.97</td><td>55</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.98</td><td>140</td></tr><tr><td>macro avg</td><td>0.98</td><td>0.98</td><td>0.98</td><td>140</td></tr><tr><td>weighted avg</td><td>0.98</td><td>0.98</td><td>0.98</td><td>140</td></tr></tbody></table> <p>Text(33.0, 0.5, 'Predicted')</p> 		precision	recall	f1-score	support	0	0.99	0.98	0.98	85	1	0.96	0.98	0.97	55	accuracy			0.98	140	macro avg	0.98	0.98	0.98	140	weighted avg	0.98	0.98	0.98	140
	precision	recall	f1-score	support																											
0	0.99	0.98	0.98	85																											
1	0.96	0.98	0.97	55																											
accuracy			0.98	140																											
macro avg	0.98	0.98	0.98	140																											
weighted avg	0.98	0.98	0.98	140																											

### Receiver Operating Characteristics (ROC) curve & Area Under Curve (AUC)

INPUT	OUTPUT
<pre>[ ] from sklearn.metrics import roc_curve  # roc curve for models fpr1, tpr1, thresh1 = roc_curve(y_test, pred_probK[:,1], pos_label=1) fpr2, tpr2, thresh2 = roc_curve(y_test, pred_probL[:,1], pos_label=1) fpr3, tpr3, thresh3 = roc_curve(y_test, pred_probS[:,1], pos_label=1)  # roc curve for tpr = fpr random_probs = [0 for i in range(len(y_test))] p_fpr, p_tpr, _ = roc_curve(y_test, random_probs, pos_label=1)  [ ] from sklearn.metrics import roc_auc_score  # auc scores auc_score1 = roc_auc_score(y_test, pred_probK[:,1]) auc_score2 = roc_auc_score(y_test, pred_probL[:,1]) auc_score3 = roc_auc_score(y_test, pred_probS[:,1])  print('KNN AUC (%)',auc_score1) print('LR AUC (%)',auc_score2) print('SVM AUC (%)',auc_score3)  [ ] # matplotlib import matplotlib.pyplot as plt plt.style.use('seaborn')  # plot roc curves plt.plot(fpr1, tpr1, linestyle='--',color='orange', label='KNN') plt.plot(fpr2, tpr2, linestyle='--',color='green', label='LR') plt.plot(fpr3, tpr3, linestyle='--',color='red', label='SVM') plt.plot(p_fpr, p_tpr, linestyle='--', color='blue') # title plt.title('ROC curve') # x label plt.xlabel('False Positive Rate') # y label plt.ylabel('True Positive rate')  plt.legend(loc='best') plt.savefig('ROC',dpi=300) plt.show();</pre>	<p>KNN AUC (%): 0.9855614973262032</p> <p>LR AUC (%): 0.9929411764705882</p> <p>SVM AUC (%): 0.9925133689839573</p> 



## **DISCUSSION**

After cleaning the data, the dataset is split into input (X) and output (Y). The input and output are further classified into train datasets and test datasets. The percentage of train and test datasets are 80% and 20% respectively from the actual data. Three different machine models were applied in this study which are KNN, LR and SVM. The reason for applying these models is that these datasets are suitable for data classification. After applying the Machine Learning model, the model is evaluated by confusion matrix, accuracy score, classification report with Precision, Sensitivity, F1 Score and AUC.

The confusion matrix compares the actual target value with the values predicted by the machine learning model and made up of a two-dimensional table based on the results from the predicted and actual data. It is able to evaluate the performance of classification problems of two or more types of classes by declaring 4 categories of results: True Positive, False Positive, True Negative and False Negative.

Therefore, the confusion matrix is an appropriate solution to assess the ability of the machine learning model to give accurate results. Table 1 shows the comparison of confusion matrices of each machine learning model used in this project. Both K-NN and SVM models predicted a total of 137 correct values (True Positive and True Negative), 2 False Negative and 1 False Positive. On the other hand, LR resulted in 136 correct values, 3 false negatives and 1 false-positive value.

	Malignant	Benign	
K-NN	83	2	Malignant
	1	54	Benign
LR	82	3	Malignant
	1	54	Benign
SVM	83	2	Malignant
	1	54	Benign

**Table 1:** Confusion matrix of K-NN, LR and SVM

Certain terms can be found in the classification reports. Accuracy score defines the number of correct predictions made from the ML model. The accuracy percentage of the model on analysis in Breast-Cancer datasets are shown in Table 2. All three models have an accuracy score of 97%. K-NN and SVM have an accuracy score of 97.6 and LR have a slightly lower accuracy score with 97.1%.

Model	Accuracy Score
K-NN	97.6%
LR	97.1%
SVM	97.6%

**Table 2:** Accuracy score for breast cancer datasets.

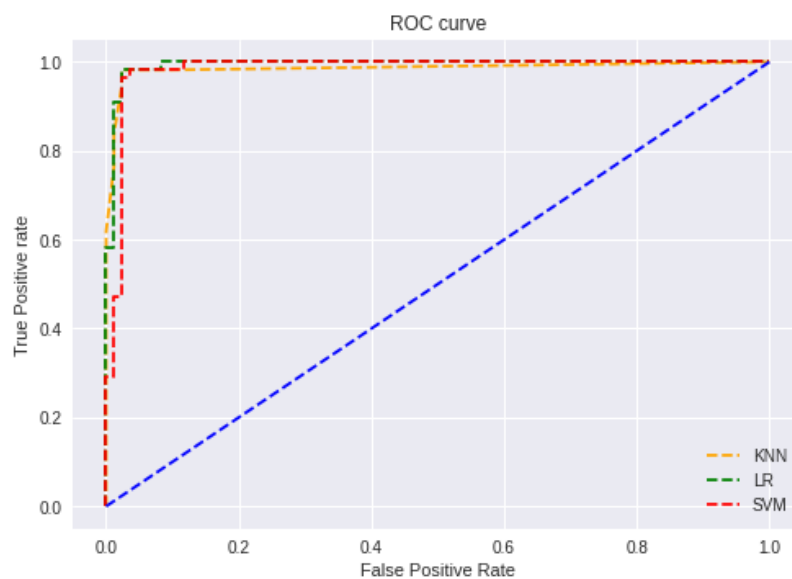
Precision is defined by the ratio of true positive to the sum of true and false positive while recall is defined by the ratio of true positive to the sum of true positive and false negative. On the other hand, F1 measures the average of precision and sensitivity. The value of F1-score which is closer to 1.0 indicates a better performance. Lastly, the area under the curve (AUC) is a way to measure the ability of the model to classify the datasets with a higher AUC indicating a better model to distinguish between the classes.

Table 3 shows the comparison of the three machine learning models in terms of precision, recall and F1-score. With the precision of 0.99, recall of 0.98 and F1-score of 0.98 in identifying benign cancer, both K-NN and SVM models performed better than LR which scored 0.99 for precision, 0.96 for recall and 0.98 for F1-score. None of the models outperform another and all three models are able to classify benign and malignant cancer based on the Breast Cancer Wisconsin Datasets.

Algorithm	Precision	Recall	F1-score	Class
K-NN	0.99	0.98	0.98	Benign
	0.96	0.98	0.97	Malignant
LR	0.99	0.96	0.98	Benign
	0.95	0.98	0.96	Malignant
SVM	0.99	0.98	0.98	Benign
	0.96	0.98	0.97	Malignant

**Table 3:** Confusion matrix of K-NN, LR and SVM machine learning model.

The area under the ROC curve for KNN, LR and SVM are 0.985, 0.993 and 0.993 respectively. ROC curve is one of the important metrics for evaluating the performance of the ML classification models. The higher the value of the AUC %, the better the performance of the model. Based on this evaluation, the LR and SVM performed slightly better than KNN.



**Figure 15:** ROC curve for KNN, LR & SVM.

## **SUMMARY**

We analysed the Breast Cancer Wisconsin (Original) Dataset using the scikit-learn Python library in Google Colab. Three machine learning algorithms that we chose to apply were K-Nearest Neighbours (KNN), Logistics Regression (LR) and Support Vector Machine (SVM). The best k value for KNN model was 21 while the best condition for the SVM is  $C=1$ ,  $\gamma=0.01$  with kernel='rbf'. The performance of each model was evaluated using a confusion matrix, accuracy score, classification report and ROC curve followed by a comparison between the models.

KNN and SVM obtained higher efficiency in confusion matrix and classification reports with 137 correct values, precision of 0.99, recall or 0.99 and F1-score of 0.98. Both KNN and SVM also achieved higher accuracy score of 97.6%.

However, LR and SVM had the highest area under the ROC with 0.993. SVM slightly outperformed KNN and LR models, since SVM has the highest values for all evaluation models. Therefore, we concluded that the SVM model is a better model to apply in the Breast Cancer datasets. However, this study is only applicable to our selected Breast Cancer dataset. Since different machine learning models have different sets of advantages and disadvantages, this evaluation may not be applicable to other datasets. Table 4 shows the summary of the machine learning models chosen for this project.

	K-NN	Logistic regression	SVM
Can be used for	Classification, regression	Classification	Classification
Basic concept	Predicts outcome by finding the nearest neighbouring data points.	Predicts outcome by formulating the best fit line that passes all the data points with minimal distance.	Predicts outcome by constructing hyperplanes that separates the different classes.
Advantages	Easy and fast to use. Can be used for small sample size.	Easy and fast to use with good accuracy.	Effective in high dimensional spaces. Useful for data with unknown distribution.
Disadvantages	Computationally and time costly, have to compute distance of unknown object to all neighbouring points.	Requires a large sample size to be accurate. Not suitable for complex relations.	Not suitable for large datasets.

**Table 4:** Summary of comparison between the three machine learning models chosen for this project.

## **CONCLUSION**

Machine learning techniques work by building a statistical model based on training data in order to make predictions. Different techniques apply different algorithms in dealing with the data and therefore would have different sets of advantages and disadvantages. Therefore, a good understanding of the available dataset and the expected outcome is very important in deciding on which machine learning technique that should be applied. In addition to that, evaluation of a machine learning method is also very important in determining the accuracy and reliability of the model.

## **REFERENCES**

1. 14 Different Types of Learning in Machine Learning by Jason Brownlee.  
<https://machinelearningmastery.com/types-of-learning-in-machine-learning/>
2. Commonly used Machine Learning Algorithms (with Python and R Codes) by Sunil Ray.  
<https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/>
3. 20 Popular Machine Learning Metrics. Part 1: Classification & Regression Evaluation Metrics by Shervin Minaee, 28 October 2019. <https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce>
4. 11 Important Model Evaluation Metrics for Machine Learning Everyone Should Know by Tavish Srivastava, 6 August 2019. <https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/>
5. Machine Learning Basics with the K-Nearest Neighbors Algorithm by Onel Harrison.  
<https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>
6. K-Nearest Neighbor(KNN) Algorithm for Machine Learning. <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>
7. 6 Types of Regression Models in Machine Learning You Should Know About by Pavan Vadapalli, 27 July 2020. <https://www.upgrad.com/blog/types-of-regression-models-in-machine-learning/>
8. Introduction to Logistic Regression by Ayush Pant, 22 January 2019.  
<https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148>
9. Logistic Regression for Machine Learning by Jason Brownlee, 1 April 2016.  
<https://machinelearningmastery.com/logistic-regression-for-machine-learning/>
10. Support Vector Machines: A Simple Explanation by Noel Bambrick, Aylien.  
<https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html>
11. Understanding Support Vector Machine(SVM) algorithm from examples (along with code) by Sunil Ray. <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>