



**INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA**

**KULLIYAH OF INFORMATION AND COMMUNICATION TECHNOLOGY  
SEMESTER 2, 2024/2025**

**CSCI 2304**

**INTELLIGENT SYSTEM**

**SECTION 4**

**LECTURER'S NAME: ASST. PROF. DR. AMIR AATIEFF AMIR HUSSIN**

**TITLE: MACHINE LEARNING TECHNIQUES ON CANCER CLASSIFICATION**

**GROUP: MUHYIDDIN YASSIN**

<b>GROUP MEMBERS</b>	<b>MATRIC NUMBERS</b>
AMMAR QASIM FOOTEN BIN JOHN ANTHONY FOOTEN	2217441
MAHAMAT ALI ADAM	2129455
UNGKU QISTINA BINTI UNGKU MOHD FARIS	2215442
SYASYA BINTI SYAERILL	2213690

**DUE DATE:**

16th June 2024

## Machine Learning Techniques on Cancer Classification

<b>1.0 Introduction</b>	<p>Cancer remains one of the leading causes of morbidity and mortality worldwide, with millions of new cases diagnosed each year. As a complex and multifaceted disease, cancer research requires powerful data analysis techniques to detect trends. Our data analysis program focuses on leveraging these advancements to conduct comprehensive analyses of cancer-related data by using machine learning techniques which is k-NN, Decision Tree Classifier, Support Vector Machine (SVM) and Naive-Bayes</p>																						
<b>2.0 Problem Statement</b>	<p>The project seeks to enhance the accuracy and efficiency of cancer classification by applying and comparing various machine learning techniques, such as k-NN, Decision Tree Classifier, SVM, and Naive-Bayes, on a dataset of breast cancer cases from UCI's repository. The goal is to identify the most effective algorithm for predicting cancer malignancy, thereby aiding in early detection and treatment planning.</p>																						
<b>2.0.1 Objectives</b>	<p>The primary objectives of the Machine Learning Techniques on Cancer Classification System are:</p> <ul style="list-style-type: none"><li>• Improve the accuracy of cancer classification using machine learning techniques.</li><li>• Compare the performance of different machine learning models.</li><li>• Select the best-performing model for future predictions.</li></ul>																						
<b>2.1 Data Set:</b>  This data set is sourced from UCI's Breast Cancer Wisconsin.	<table><thead><tr><th>No. Attributes</th><th>Definition</th></tr></thead><tbody><tr><td>1. Radius</td><td>(Distances from center to points on the perimeter)</td></tr><tr><td>2. Texture</td><td>(Standard deviation of gray-scale values)</td></tr><tr><td>3. Perimeter</td><td>(Total length of its boundary)</td></tr><tr><td>4. Area</td><td>(Surface measurement)</td></tr><tr><td>5. Smoothness</td><td>(Local variation in radius lengths)</td></tr><tr><td>6. Compactness</td><td>(Density (perimeter<sup>2</sup> / area))</td></tr><tr><td>7. Concavity</td><td>(Severity on contour of cell)</td></tr><tr><td>8. Concave points</td><td>(No. of concave portions of contour)</td></tr><tr><td>9. Symmetry</td><td>(Balanced similarity of cell)</td></tr><tr><td>10. Fractal Dimension</td><td>(Coastline approximation)</td></tr></tbody></table>	No. Attributes	Definition	1. Radius	(Distances from center to points on the perimeter)	2. Texture	(Standard deviation of gray-scale values)	3. Perimeter	(Total length of its boundary)	4. Area	(Surface measurement)	5. Smoothness	(Local variation in radius lengths)	6. Compactness	(Density (perimeter <sup>2</sup> / area))	7. Concavity	(Severity on contour of cell)	8. Concave points	(No. of concave portions of contour)	9. Symmetry	(Balanced similarity of cell)	10. Fractal Dimension	(Coastline approximation)
No. Attributes	Definition																						
1. Radius	(Distances from center to points on the perimeter)																						
2. Texture	(Standard deviation of gray-scale values)																						
3. Perimeter	(Total length of its boundary)																						
4. Area	(Surface measurement)																						
5. Smoothness	(Local variation in radius lengths)																						
6. Compactness	(Density (perimeter <sup>2</sup> / area))																						
7. Concavity	(Severity on contour of cell)																						
8. Concave points	(No. of concave portions of contour)																						
9. Symmetry	(Balanced similarity of cell)																						
10. Fractal Dimension	(Coastline approximation)																						

<b>3.0 The Algorithm</b>	<p>Decision Tree Classifier: A decision tree is a flowchart-like tree structure where an internal node represents a feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome.</p> <p>Support Vector Machine (SVM): SVM is a supervised machine learning algorithm which can be used for classification or regression problems. It uses a technique called the kernel trick to transform the data and then based on these transformations it finds an optimal boundary between the possible outputs.</p> <p>k-Nearest Neighbors (k-NN): This algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.</p> <p>Naive Bayes: Naive Bayes classifiers are a family of simple “probabilistic classifiers” based on applying Bayes’ theorem with strong (naïve) independence assumptions between the features.</p> <p>Classification and Clustering method</p> <ul style="list-style-type: none"> <li>- Classification is used to predict the label (benign_0__mal_1) for new cancer cases based on the features in the dataset.it will train the classification model using features such as radius and texture to predict whether the new cases are benign or malignant.</li> <li>- Clustering is an unsupervised learning technique used to group similar data points into clusters based on their inherent characteristics and features.</li> </ul>
<b>3.1 Pseudocode</b>	<p>1. Import necessary libraries for data manipulation, visualization, and machine learning.</p>

### 1) Setting up Libraries

```
[ ] 1 # @title
    2 import pandas as pd
    3 import numpy as np
    4 import seaborn as sns
    5 from sklearn.model_selection import train_test_split
    6 from sklearn.preprocessing import StandardScaler
    7 from sklearn.tree import DecisionTreeClassifier
    8 from sklearn.svm import SVC
    9 from sklearn.neighbors import KNeighborsClassifier
   10 from sklearn.naive_bayes import GaussianNB
   11 from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
   12 import matplotlib.pyplot as plt
```

2. Mount Google Drive to access the dataset stored there.

### 2) Mount google drive

```
▶ 1 # @title
   2 from google.colab import drive
   3 drive.mount("/content/drive")
```

↔ Mounted at /content/drive

3. Load the cancer classification dataset from Google Drive and display the first few rows.

```
▶ 1 # @title
   2 cancer = pd.read_csv('/content/drive/MyDrive/cancer_classifications.csv')
```

4. Print the first few rows and information about the dataset to understand its structure and contents.

```

1 # @title
2 print(cancer.head())
3 print(cancer.info())

```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	\
0	17.99	10.38	122.80	1001.0	0.11840	
1	20.57	17.77	132.90	1326.0	0.08474	
2	19.69	21.25	130.00	1203.0	0.10960	
3	11.42	20.38	77.58	386.1	0.14250	
4	20.29	14.34	135.10	1297.0	0.10030	

	mean compactness	mean concavity	mean concave points	mean symmetry	\
0	0.27760	0.3001	0.14710	0.2419	
1	0.07864	0.0869	0.07017	0.1812	
2	0.15990	0.1974	0.12790	0.2069	
3	0.28390	0.2414	0.10520	0.2597	
4	0.13280	0.1980	0.10430	0.1809	

	mean fractal dimension	benign_0_mal_1
0	0.07871	0
1	0.05667	0
2	0.05999	0
3	0.09744	0
4	0.05883	0

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 11 columns):
#   Column                                Non-Null Count  Dtype
---  ---                                -
0   mean radius                          569 non-null    float64
1   mean texture                         569 non-null    float64
2   mean perimeter                       569 non-null    float64
3   mean area                           569 non-null    float64
4   mean smoothness                     569 non-null    float64
5   mean compactness                    569 non-null    float64
6   mean concavity                      569 non-null    float64
7   mean concave points                 569 non-null    float64
8   mean symmetry                       569 non-null    float64
9   mean fractal dimension               569 non-null    float64
10  benign_0_mal_1                      569 non-null    int64
dtypes: float64(10), int64(1)
memory usage: 49.0 KB
None

```

## 5. Preprocess the Data:

- Separate the dataset into features (X) and target variable (y).
- Split the data into training and testing sets.
- Standardize the numerical features using StandardScaler.

```

1 # @title
2 # Step 2: Preprocess the Data
3 # Split the data into features and target variable
4 X = cancer.drop('benign_0_mal_1', axis=1)
5 y = cancer['benign_0_mal_1']

```

```

[ ] 1 # @title
2 # Split the data into training and testing sets
3 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```

```

[ ] 1 # @title
2 # Standardize numerical features
3 scaler = StandardScaler()
4 X_train = scaler.fit_transform(X_train)
5 X_test = scaler.transform(X_test)

```

## 6. Select and Train Classifiers:

- Define a dictionary of classifiers with their corresponding model objects.
- Train each classifier on the training data.
- Predict on the testing data using each trained classifier.
- Calculate accuracy, classification report, and confusion matrix for each classifier.

e. Print out the performance metrics for each classifier.

```
1 # @title
2 # Step 3: Select and Train Classifiers
3 classifiers = {}
4 'Decision Tree': DecisionTreeClassifier(),
5 'SVM': SVC(),
6 'k-NN': KNeighborsClassifier(),
7 'Naive Bayes': GaussianNB()
8 }

[ ] 1 # @title
2 # Train and evaluate each classifier
3 results = {}
4 for name, clf in classifiers.items():
5     clf.fit(X_train, y_train)
6     y_pred = clf.predict(X_test)
7     accuracy = accuracy_score(y_test, y_pred)
8     classification_report_str = classification_report(y_test, y_pred)
9     confusion_matrix_array = confusion_matrix(y_test, y_pred)
10    results[name] = {
11        'accuracy': accuracy,
12        'classification_report': classification_report_str,
13        'confusion_matrix': confusion_matrix_array
14    }
15    print(f"{name} Classifier:")
16    print("Accuracy:", accuracy)
17    print("Classification Report:\n", classification_report_str)
18    print("Confusion Matrix:\n", confusion_matrix_array)
19    print("\n")
```

7. Evaluate and Compare Classifiers:

- Extract accuracy for each classifier from the results dictionary.
- Plot a bar chart to compare classifier performance based on accuracy.

```
1 # @title
2 # Step 5: Evaluate and Compare Classifiers
3 # Plotting accuracy for each classifier
4 accuracies = {name: result['accuracy'] for name, result in results.items()}
5 plt.bar(accuracies.keys(), accuracies.values())
6 plt.xlabel('Classifiers')
7 plt.ylabel('Accuracy')
8 plt.title('Classifier Performance Comparison')
9 plt.xticks(rotation=45)
10 plt.show()
```

8. Predict on new data:

- Define new data points as an array.
- Preprocess the new data using the same scaler used for training data.
- Use the best-performing classifier to predict on the new scaled data.
- Convert numerical predictions to labels ('Malignant' or 'Benign').
- Print out predictions for new data points.

	<pre> 1 # @title 2 new_data = np.array([ 3     [13.45, 14.36, 87.46, 566.3, 0.09779, 0.08129, 0.06664, 0.04781, 0.1885, 0.05766], 4     [17.99, 10.38, 122.8, 1001.0, 0.1184, 0.2776, 0.3001, 0.1471, 0.2419, 0.07871] 5 ])  1 # @title 2 # Preprocess the new data using the same scaler 3 new_data_scaled = scaler.transform(new_data)  [ ] 1 # @title 2 # Predict using the best-performing classifier (e.g., SVM) 3 best_classifier = classifiers['SVM'] 4 prediction = best_classifier.predict(new_data_scaled)  [ ] 1 # @title 2 # Output the predictions 3 prediction_labels = ['Malignant' if pred == 1 else 'Benign' for pred in prediction] 4 print(f'The predictions for the new data points are: {prediction_labels}') </pre> <p>9. Visualize Data:</p> <ol style="list-style-type: none"> <li>Create a scatter plot with mean radius vs mean concavity, colored by malignancy.</li> <li>Add title and labels to the plot.</li> <li>Display the plot.</li> </ol> <pre> 1 # @title 2 # Create the scatter plot with color based on species 3 sns.scatterplot( 4     data=cancer, 5     x="mean radius", 6     y="mean concavity", 7     hue="benign_0__mal_1", 8     palette="BuPu", # Choose a color palette from seaborn 9 ) 10 11 # Add title and labels 12 plt.title("Mean Radius vs Mean Concavity by Malignancy") 13 plt.xlabel("Mean Radius (mm)") 14 plt.ylabel("Mean Concavity (mm)") 15 16 # Show the plot 17 plt.show() </pre>
<p><b>4.0 Analysis of Result</b></p>	<p>Model Training and Evaluation: Four different classifiers (Decision Tree, SVM, k-NN, Naive Bayes) are trained on the cancer dataset. Each model's performance is evaluated using accuracy, classification report, and confusion matrix. The accuracy scores indicate how well each model predicts the test data.</p>

Decision Tree Classifier:  
Accuracy: 0.9385964912280702

Classification Report:

	precision	recall	f1-score	support
0	0.91	0.93	0.92	43
1	0.96	0.94	0.95	71
accuracy			0.94	114
macro avg	0.93	0.94	0.93	114
weighted avg	0.94	0.94	0.94	114

Confusion Matrix:

```
[[40  3]
 [ 4 67]]
```

SVM Classifier:  
Accuracy: 0.9736842105263158

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.93	0.96	43
1	0.96	1.00	0.98	71
accuracy			0.97	114
macro avg	0.98	0.97	0.97	114
weighted avg	0.97	0.97	0.97	114

Confusion Matrix:

```
[[40  3]
 [ 0 71]]
```

k-MN Classifier:  
Accuracy: 0.9473684210526315

Classification Report:

	precision	recall	f1-score	support
0	0.93	0.93	0.93	43
1	0.96	0.96	0.96	71
accuracy			0.95	114
macro avg	0.94	0.94	0.94	114
weighted avg	0.95	0.95	0.95	114

Confusion Matrix:

```
[[40  3]
 [ 3 68]]
```

Naive Bayes Classifier:  
Accuracy: 0.9385964912280702

Classification Report:

	precision	recall	f1-score	support
0	0.95	0.88	0.92	43
1	0.93	0.97	0.95	71
accuracy			0.94	114
macro avg	0.94	0.93	0.93	114
weighted avg	0.94	0.94	0.94	114

Confusion Matrix:

```
[[38  5]
 [ 2 69]]
```

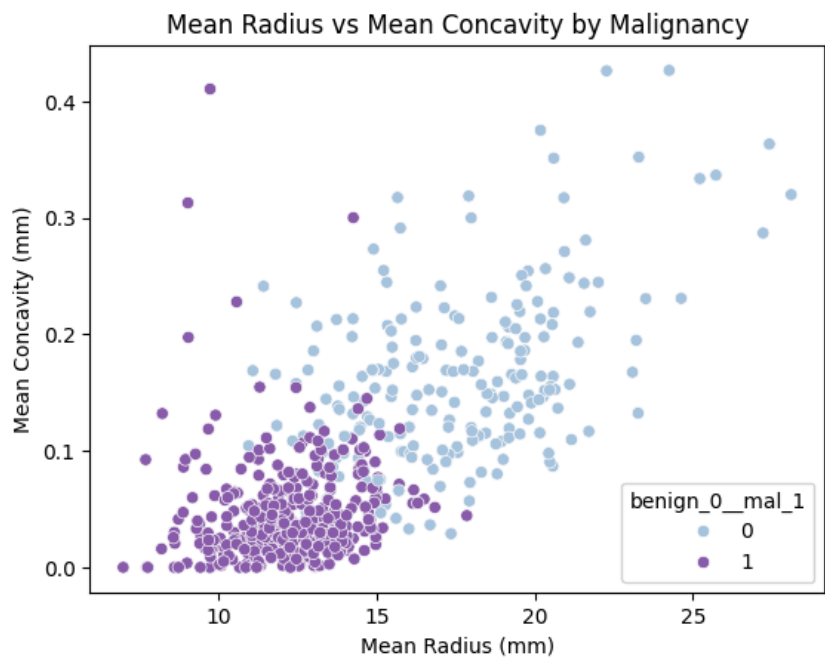


Best Classifier: The SVM classifier is chosen as the best-performing model based on accuracy. It is then used to predict new data points after scaling them with the same scaler used for training data.

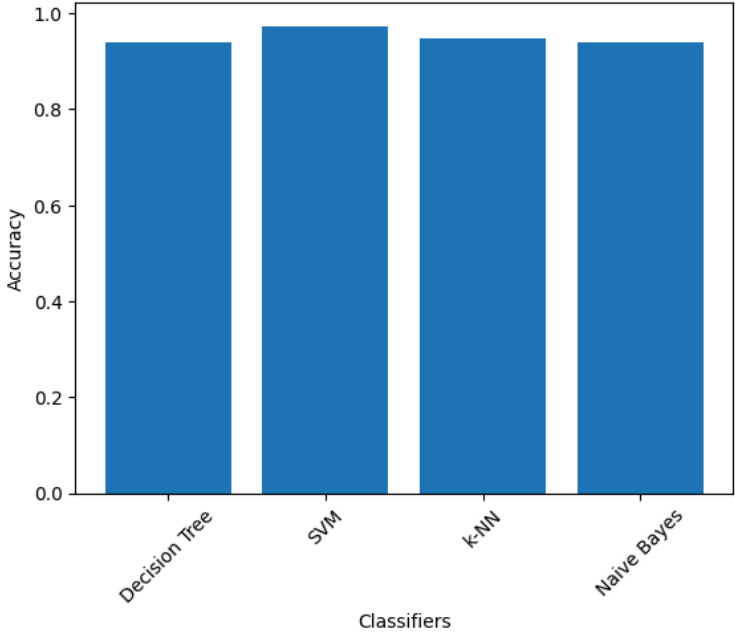
Prediction Results: The predictions for new data points are outputted as 'Malignant' or 'Benign', which shows the model's capability to classify unseen data.

```
The predictions for the new data points are: ['Malignant', 'Benign']
```

Data Visualization: A scatter plot visualizes the relationship between 'mean radius' and 'mean concavity' with color coding to distinguish between benign and malignant cases. This can help in understanding how these features relate to cancer classification.



Performance Comparison: A bar plot compares the accuracy of each classifier, providing a visual representation of which models performed better on this dataset.

	<div><p>Classifier Performance Comparison</p><table border="1"><thead><tr><th>Classifier</th><th>Accuracy</th></tr></thead><tbody><tr><td>Decision Tree</td><td>0.94</td></tr><tr><td>SVM</td><td>0.97</td></tr><tr><td>k-NN</td><td>0.95</td></tr><tr><td>Naive Bayes</td><td>0.94</td></tr></tbody></table></div> <p><b>Classification:</b></p> <p>The bar chart shows the accuracy of Decision Tree, SVM (Support Vector Machine), k-NN (k-Nearest Neighbors), and Naive Bayes—in a cancer classification. All classifiers exhibit high accuracy, with Decision Tree and SVM performing the best, nearly reaching 1.0 (or 100% accuracy), while Naive Bayes has the lowest accuracy among them, albeit still close to 0.9. This highlights the strong performance of these models in improving diagnostic accuracy for the cancer classification.</p>	Classifier	Accuracy	Decision Tree	0.94	SVM	0.97	k-NN	0.95	Naive Bayes	0.94
Classifier	Accuracy										
Decision Tree	0.94										
SVM	0.97										
k-NN	0.95										
Naive Bayes	0.94										
<b>5.0 Conclusion</b>	<p>In conclusion, our exploration of machine learning techniques for cancer classification demonstrated the significant potential of these models to improve diagnostic accuracy and patient outcomes. By comparing various algorithms, including Decision Tree, SVM, k-NN, and Naive Bayes, we highlighted the capability of machine learning to offer precise, data-driven insights that can complement and enhance traditional medical diagnostics. Future work will focus on integrating more advanced models, utilizing larger datasets, and ensuring clinical applicability to further revolutionize cancer detection and treatment.</p>										
<b>6.0 Code and Data Set</b>	<p><a href="#">Link for the code</a></p> <p><a href="#">Link for data set</a> (needs to be saved into google drive first for the codes to work)</p>										