AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY (AUST)
141 & 142, Love Road, Tejgaon Industrial Area, Dhaka-1208.

Department of Computer Science and Engineering
Program: Bachelor of Science in Computer Science and Engineering

Course No: CSE-3208
Course Title: Introduction to Artificial Intelligence lab

Date of Submission: 10.03.2025

Submitted to,
Ms. Nawrin Tabassum
Ms. Sumaiya Nuha Mustafina

Submitted by,
Name: Nasrin Akther Jerin
Student ID: 20210204002
Group: A1

Report on Machine Learning Models for Binary Classification

Introduction

This report focuses on building and evaluating two machine learning models, Decision Tree and K-Nearest Neighbors (KNN), to solve a binary classification problem in healthcare. The dataset chosen for this study is the **Survey Lung Cancer** dataset. The objective is to predict the likelihood of lung cancer based on various patient attributes.

Why This Dataset?

The **Survey Lung Cancer** dataset was chosen because it contains relevant attributes that can help in predicting lung cancer cases. Early detection of lung cancer can significantly improve patient outcomes. This dataset includes various risk factors such as smoking, alcohol consumption, age, and gender, making it an ideal choice for this study.

Data Preprocessing

Importing necessary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

pandas: Used for data manipulation, loading, and cleaning.
numpy: Used for numerical operations and handling missing values efficiently.
matplotlib.pyplot: Used for generating plots and visualizing insights.
seaborn: Used for advanced statistical visualizations and feature analysis.

.

Importing machine learning libraries

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, classification_report
```

Train_test_split: Splits the dataset into training and testing sets.
StandardScaler: Standardizes features to improve KNN model performance.
SimpleImputer: Handles missing values by replacing them with mean, median, or most frequent values.
DecisionTreeClassifier: Implements a decision tree for classification tasks.

KNeighborsClassifier: Implements the K-Nearest Neighbors algorithm for classification.
Metrics from sklearn.metrics: Used to evaluate model performance based on accuracy, precision, recall, and F1-score

Load dataset

```
df = pd.read_csv('/survey lung cancer.csv')
```

Examine dataset structure

- `df.head()`

Displays the first five rows of the dataset.

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH	SWALLOWING DIFFICULTY	CHEST PAIN	LUNG_CANCER
0	M	69	1	2	2	1	1	2	1	2	2	2	2	2	2	YES
1	M	74	2	1	1	1	2	2	2	1	1	1	2	2	2	YES
2	F	59	1	1	1	2	1	2	1	2	1	2	2	1	2	NO
3	M	63	2	2	2	1	1	1	1	1	2	1	1	2	2	NO
4	F	63	1	2	1	1	1	1	1	2	1	2	2	1	1	NO

- `print(df.shape)`

Shows the number of rows and columns.

```
(309, 16)
```

- `print(df.info())`

Displays dataset information including missing values and data types.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 309 entries, 0 to 308
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   GENDER                 309 non-null    object
 1   AGE                    309 non-null    int64
 2   SMOKING                309 non-null    int64
 3   YELLOW_FINGERS         309 non-null    int64
 4   ANXIETY                309 non-null    int64
 5   PEER_PRESSURE          309 non-null    int64
 6   CHRONIC DISEASE        309 non-null    int64
 7   FATIGUE                309 non-null    int64
 8   ALLERGY                309 non-null    int64
 9   WHEEZING               309 non-null    int64
 10  ALCOHOL CONSUMING      309 non-null    int64
 11  COUGHING               309 non-null    int64
 12  SHORTNESS OF BREATH    309 non-null    int64
 13  SWALLOWING DIFFICULTY  309 non-null    int64
 14  CHEST PAIN             309 non-null    int64
 15  LUNG_CANCER            309 non-null    object
dtypes: int64(14), object(2)
memory usage: 38.8+ KB
None
```

- `print(df.describe())`

Summarizes statistical insights such as mean, min, max values of numeric columns.

```
              AGE     SMOKING  YELLOW_FINGERS     ANXIETY  PEER_PRESSURE  \
count  309.000000  309.000000      309.000000  309.000000     309.000000
mean    62.673139    1.563107        1.569579    1.498382       1.501618
std      8.210301    0.496806        0.495938    0.500808       0.500808
min     21.000000    1.000000        1.000000    1.000000       1.000000
25%     57.000000    1.000000        1.000000    1.000000       1.000000
50%     62.000000    2.000000        2.000000    1.000000       2.000000
75%     69.000000    2.000000        2.000000    2.000000       2.000000
max     87.000000    2.000000        2.000000    2.000000       2.000000

       CHRONIC DISEASE     FATIGUE     ALLERGY     WHEEZING  ALCOHOL CONSUMING  \
count       309.000000  309.000000  309.000000  309.000000         309.000000
mean          1.504854    1.673139    1.556634    1.556634           1.556634
std           0.500787    0.469827    0.497588    0.497588           0.497588
min           1.000000    1.000000    1.000000    1.000000           1.000000
25%           1.000000    1.000000    1.000000    1.000000           1.000000
50%           2.000000    2.000000    2.000000    2.000000           2.000000
75%           2.000000    2.000000    2.000000    2.000000           2.000000
max           2.000000    2.000000    2.000000    2.000000           2.000000

         COUGHING  SHORTNESS OF BREATH  SWALLOWING DIFFICULTY  CHEST PAIN
count  309.000000           309.000000             309.000000  309.000000
mean     1.579288             1.640777               1.469256    1.556634
std      0.494474             0.480551               0.499863    0.497588
min      1.000000             1.000000               1.000000    1.000000
25%      1.000000             1.000000               1.000000    1.000000
50%      2.000000             2.000000               1.000000    2.000000
75%      2.000000             2.000000               2.000000    2.000000
max      2.000000             2.000000               2.000000    2.000000
```

Handling Categorical Data

- Convert categorial 1 for Lung Cancer 0 for no lung cancer

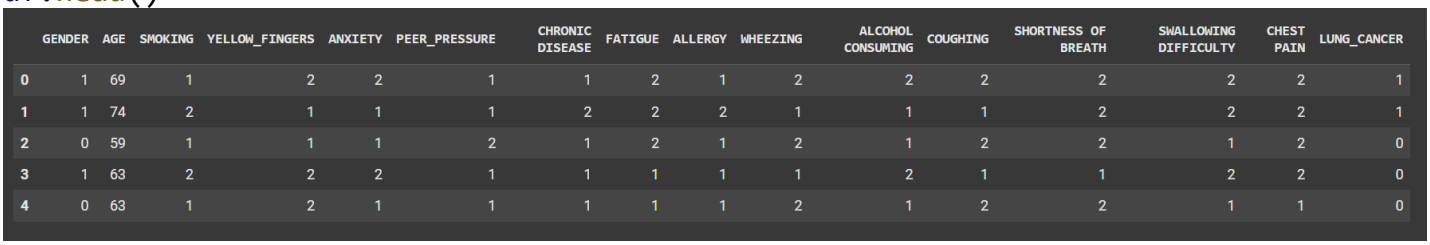
```
df['LUNG_CANCER'] = df['LUNG_CANCER'].apply(lambda x: 1 if x == 'YES' else 0)
```

- Convert categorial 1 for Male(M) 0 for Female(F)

```
df['GENDER']=df['GENDER'].apply(lambda x: 1 if x == 'M' else 0)
```

Check the transformed dataset

```
df.head()
```



	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH	SWALLOWING DIFFICULTY	CHEST PAIN	LUNG_CANCER
0	1	69	1	2	2	1	1	2	1	2	2	2	2	2	2	1
1	1	74	2	1	1	1	2	2	2	1	1	1	2	2	2	1
2	0	59	1	1	1	2	1	2	1	2	1	2	2	1	2	0
3	1	63	2	2	2	1	1	1	1	1	2	1	1	2	2	0
4	0	63	1	2	1	1	1	1	1	2	1	2	2	1	1	0

Handling Missing Data

- Replace missing values with mean

```
df['ALCOHOL CONSUMING'].fillna(df['ALCOHOL CONSUMING'].mean(),
inplace=True)
```

- Drop unnecessary columns if they exist

```
df = df.drop(columns=['Unnamed: 0', 'ID'], errors='ignore'
```

Splitting Dataset into Training and Testing Sets: Splits the dataset into training and testing sets, where the training set is used to train the model and the testing set is used for evaluation.

```
x=df.drop(columns=['LUNG_CANCER'])
y=df['LUNG_CANCER']
```

- calculate test size based on the last 3 digits of ID

```
test_size=(2 % 40) / 100
```

The size of the test set is dynamically determined. Here 98% is training set & 2% is test set.

```
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=test_size, random_state=42)

print(f"Training set size: {x_train.shape}, Test set size:
{x_test.shape}")
```

- **Feature Scaling for KNN**: Since KNN is distance-based, feature scaling is necessary to ensure all variables contribute equally.

```
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

Model Training: Decision tree and a KNN models are initialized using **Scikit-learn** and trained on training dataset.

- Initialize models

```
dt_model = DecisionTreeClassifier(random_state=42)
knn_model = KNeighborsClassifier(n_neighbors=5)
```

- Train models

```
dt_model.fit(x_train, y_train)
knn_model.fit(x_train, y_train)
```

Model Evaluation: Accuracy, Precision, Recall, and F1-score are printed for both models.

- calculate matrices

```
def evaluate_model(y_true, y_pred, model_name):
    print(f"Evaluation Metrics for {model_name}:")
    print("Accuracy:", accuracy_score(y_true, y_pred))
    print("Precision:", precision_score(y_true, y_pred))
    print("Recall:", recall_score(y_true, y_pred))
    print("F1-score:", f1_score(y_true, y_pred))
    print(classification_report(y_true, y_pred))
    print("\n")
```

- Predictions

```
y_pred_dt = dt_model.predict(x_test)
y_pred_knn = knn_model.predict(x_test)
```

- Evaluate

```
evaluate_model(y_test, y_pred_dt, "Decision Tree")
evaluate_model(y_test, y_pred_knn, "KNN")
```

```
Evaluation Metrics for Decision Tree:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1-score: 1.0
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         7

    accuracy                           1.00         7
   macro avg       1.00      1.00      1.00         7
weighted avg       1.00      1.00      1.00         7

Evaluation Metrics for KNN:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
```

```
F1-score: 1.0
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         7

    accuracy                           1.00         7
   macro avg       1.00      1.00      1.00         7
weighted avg       1.00      1.00      1.00         7
```

Best Performing Model

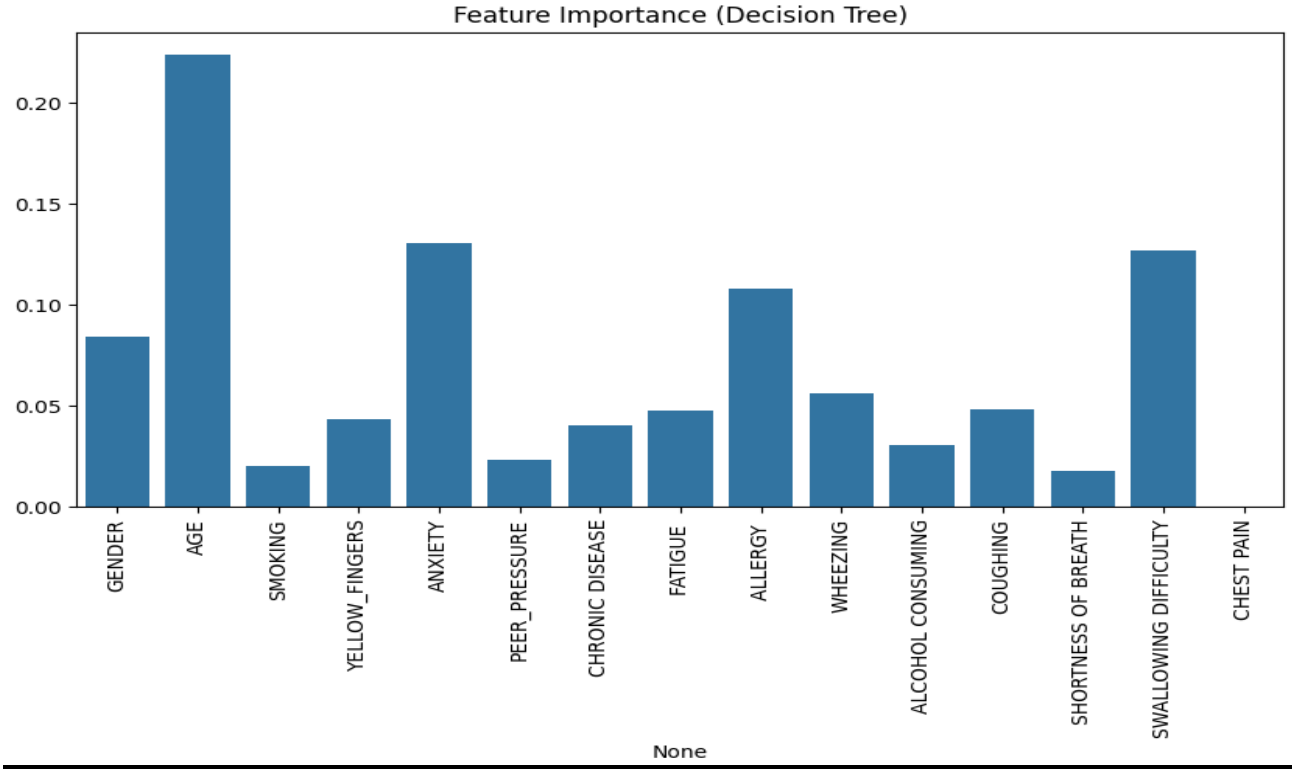
```
f1_dt = f1_score(y_test, y_pred_dt)
f1_knn = f1_score(y_test, y_pred_knn)

better_model = "Decision Tree" if f1_dt > f1_knn else "KNN"
print(f"The better model based on F1-score is: {better_model}")
```

```
The better model based on F1-score is: KNN
```

Feature Importance Visualization

```
feature_importance = dt_model.feature_importances_
plt.figure(figsize=(10,5))
sns.barplot(x=x.columns, y=feature_importance)
plt.xticks(rotation=90)
plt.title("Feature Importance (Decision Tree)")
plt.show()
```



Hyperparameter Tuning for Decision Tree and KNN

- Decision Tree Hyperparameter Tuning using GridSearchCV.

```
from sklearn.model_selection import GridSearchCV
# Define parameter grid
param_grid_dt = {
    'max_depth': [3, 5, 10, None],
    'min_samples_split': [2, 5, 10]
}
# Grid search
grid_dt = GridSearchCV(DecisionTreeClassifier(random_state=42),
param_grid_dt, cv=5, scoring='f1')
grid_dt.fit(x_train, y_train)
```

```
Best parameters for Decision Tree: {'max_depth': 10,
'min_samples_split': 10}
```

- KNN Hyperparameter Tuning using GridSearchCV.

```
param_grid_knn = {
    'n_neighbors': [3, 5, 7, 10],
    'weights': ['uniform', 'distance']
}
grid_knn = GridSearchCV(KNeighborsClassifier(), param_grid_knn, cv=5,
scoring='f1')
grid_knn.fit(x_train_scaled, y_train)

print("Best parameters for KNN:", grid_knn.best_params_)
```

```
Best parameters for KNN: {'n_neighbors': 3, 'weights': 'distance'}
```

Conclusion

The **Survey Lung Cancer** dataset was used due to its relevance in predicting lung cancer based on various health factors. A **Decision Tree** and **K-Nearest Neighbors** model were implemented and evaluated. **F1-score** was used to determine the best-performing model. **Hyperparameter tuning** was performed for both Decision Tree and KNN to improve performance. The **KNN** model was found to be the best performer. Feature importance analysis helped identify the most significant attributes. This project demonstrates the use of machine learning in **healthcare predictive analysis**, providing valuable insights into lung cancer prediction.