SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE NAME: Machine Learning Laboratory **COURSE CODE:** DJS22L602

CLASS: Third Year B.Tech **SEM:** VI

NAME: DIPTI AGARWAL **SAP: 60003220189**

EXPERIMENT NO. 4

CO Measured:

CO3 - Apply various machine learning techniques.

TITLE: Evaluation of Classification techniques.

AIM / OBJECTIVE:

To implement classification techniques and evaluate the performance and analyses results using confusion matrix.

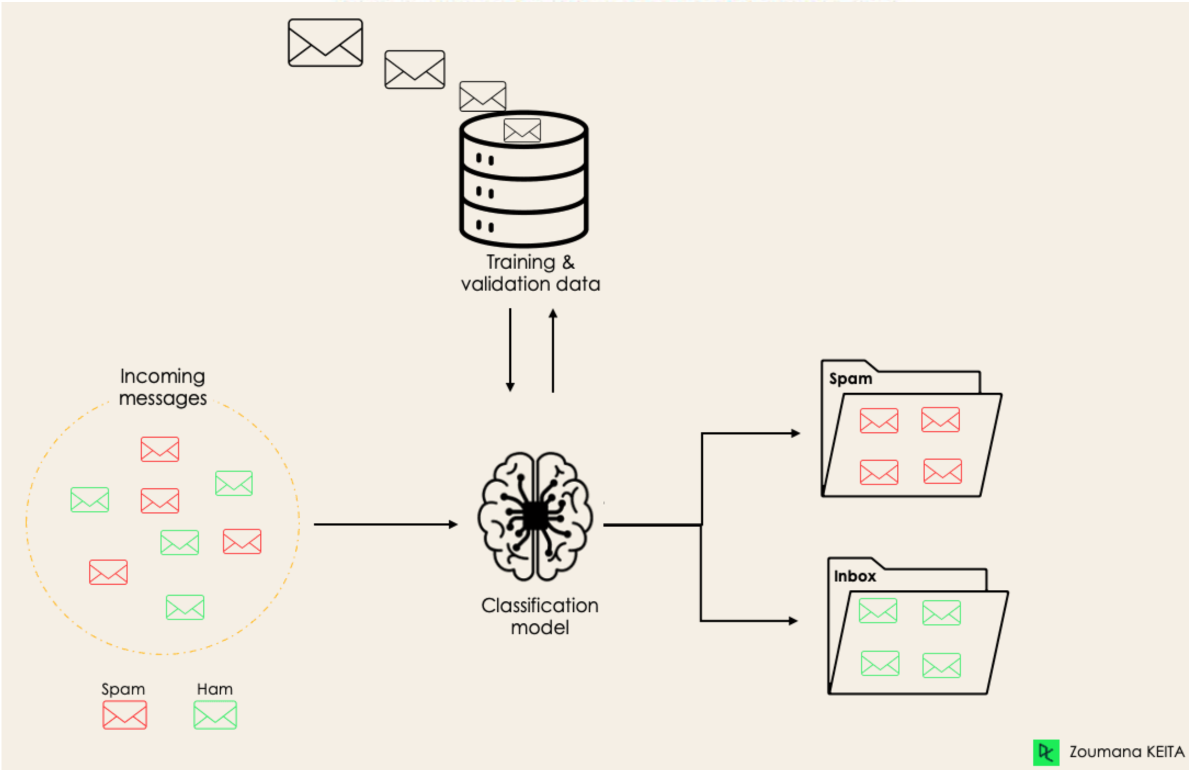
DESCRIPTION OF EXPERIMENT:

Classification

Classification is a supervised machine learning method where the model tries to predict the correct label of a given input data. In classification, the model is fully trained using the training data, and then it is evaluated on test data before being used to perform prediction on new unseen data.

For instance, an algorithm can learn to predict whether a given email is spam or ham (no spam), as illustrated below.

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)



Lazy Learners Vs. Eager Learners

There are two types of learners in machine learning classification: lazy and eager learners.

Eager learners are machine learning algorithms that first build a model from the training dataset before making any prediction on future datasets. They spend more time during the training process because of their eagerness to have a better generalization during the training from learning the weights, but they require less time to make predictions.

Most machine learning algorithms are eager learners, and below are some examples:

- Logistic Regression.
- Support Vector Machine.
- Decision Trees.
- Artificial Neural Networks.

Lazy learners or instance-based learners, on the other hand, do not create any model immediately from the training data, and this is where the lazy aspect comes from. They just memorize the training data, and each time there is a need to make a prediction, they search for the nearest neighbor from the whole training data, which makes them very slow during prediction. Some examples of this kind are:

- K-Nearest Neighbor.
- Case-based reasoning.

.

Binary classification algorithms

Most of the binary classification algorithms can be also used for multi-class classification. These algorithms include but are not limited to:

- Random Forest
- Naive Bayes
- K-Nearest Neighbors
- Gradient Boosting
- SVM
- Logistic Regression.

Multi-label classification

It is not possible to use multi-class or binary classification models to perform multi-label classification. However, most algorithms used for those standard classification tasks have their specialized versions for multi-label classification. We can cite:

- *Multi-label Decision Trees*
- *Multi-label Gradient Boosting*
- *Multi-label Random Forests*

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

Metrics to Evaluate Machine Learning Classification Algorithms

Now that we have an idea of the different types of classification models, it is crucial to choose the right evaluation metrics for those models. In this section, we will cover the most commonly used metrics: accuracy, precision, recall, F1 score, and area under the ROC (Receiver Operating Characteristic) curve and AUC (Area Under the Curve).

A bit of context

Imagine you are a healthcare startup, and want an AI assistant able to predict whether a given patient has a heart disease or not based on its health record. This is a binary classification problem where the model will predict

- 1, True or Yes if the patient has heart disease
- 0, False or No otherwise

1 Confusion matrix

A 2X2 matrix that nicely summarizes the number of correct predictions of the model. It also helps in computing different other performance metrics.

Predicti / Reality	Yes	No
Yes	True Positives (TP)	False Negatives (FN)
No	False Positives(FP)	True Negatives (TN)

Type I Error

Type II Error

Type I & II Errors can be used interchangeably when referring to False Positives and False negatives respectively

2 Accuracy

We get *accuracy* by answering this question: **"out of the predictions made by the model, what percentage is correct?"**

$$Accuracy = \frac{TP + TN}{\text{Total number observation}}$$

Zoumana KEITA

3 Precision

We get precision by answering this question: **"out of all the YES predictions, how many of them were correct?"**

$$\textit{Precision} = \frac{TP}{TP + FP}$$

4 Recall / Sensitivity

It aims to answer this question: **"how good was the model at predicting real Yes events?"**, which can be considered as the flip of the precision.

$$\textit{Sensitivity} = \frac{TP}{TP + FN}$$

5 Recall / Specificity

It aims to answer this question: **"how good was the model at predicting real No events?".**

$$\textit{Specificity} = \frac{TN}{TN + FP}$$

6 F1 Score

Sometimes used when dealing with imbalanced data set, meaning that there are more of one class/label than there are of the other. It corresponds to the harmonic mean of the precision and recall.

$$\textit{F1 Score} = 2 \cdot \frac{\textit{Precision} \times \textit{Recall}}{\textit{Precision} + \textit{Recall}}$$

Zoumana KEITA

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

7 AUC – ROC Curve

AUC– ROC generates probability values instead of binary 0/1 values. It should be used when your data set is roughly balanced.

Using ROC for imbalanced data sets lead to incorrect interpretation.

ROC curves provide good overview of trade-off between the TP rate and FP rate for binary classifier using different probability thresholds.

- A value below 0.5 indicates a poor classifier
- A value of 0.5 means random classifier
- Value over 0.7 corresponds to a good classifier
- 0.8 indicates a strong classifier
- We have 1 when the classifier perfectly predicts everything.

Strategies to choose the right metric

Choose accuracy
- The cost of FP and FN are roughly equal.
- The benefit of TP and TN are roughly equal.

Choose Precision
- The cost of FP is much higher than a FN.
- The benefit of a TP is much higher than a TN.

Choose recall
- The cost of FN is much higher than a FP.
- The cost of a TN is much higher than a TP.

ROC AUC & Precision – Recall curves
- Use ROC when the dealing with balanced data sets.
- Use precision-recall for imbalanced data sets.

Zoumana KEITA

PROCEDURE:

1. Use any 3 classification algorithm for a problem statement and evaluate the performance of the system using below mentioned metrics.
   Confusion matrix, accuracy, precision recall, sensitivity, specificity, PPV, NPV, ROC Curve, Lift curve and Gain curve.

CODE:

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

```
import pandas as pd
dataset = pd.read_csv('mushroom_cleaned.csv')
print(dataset.tail())
```

```
       cap-diameter  cap-shape  gill-attachment  gill-color  stem-height  \
54030            73          5                3           2     0.887740
54031            82          2                3           2     1.186164
54032            82          5                3           2     0.915593
54033            79          2                3           2     1.034963
54034            72          5                3           2     1.158311

       stem-width  stem-color    season  class
54030         569          12  0.943195      1
54031         490          12  0.943195      1
54032         584          12  0.888450      1
54033         491          12  0.888450      1
54034         492          12  0.888450      1
```

[3]
```
dataset.shape
```

```
(54035, 9)
```

[4]
```
X = dataset.drop('class', axis=1)
y = dataset['class']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
X.shape
```

```
(54035, 8)
```

[13]
```
y.shape
```

```
(54035,)
```

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

lr_model = LogisticRegression()
rf_model = RandomForestClassifier()
dt_model = DecisionTreeClassifier()

lr_model.fit(X_train, y_train)
rf_model.fit(X_train, y_train)
dt_model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
DecisionTreeClassifier
DecisionTreeClassifier()
```

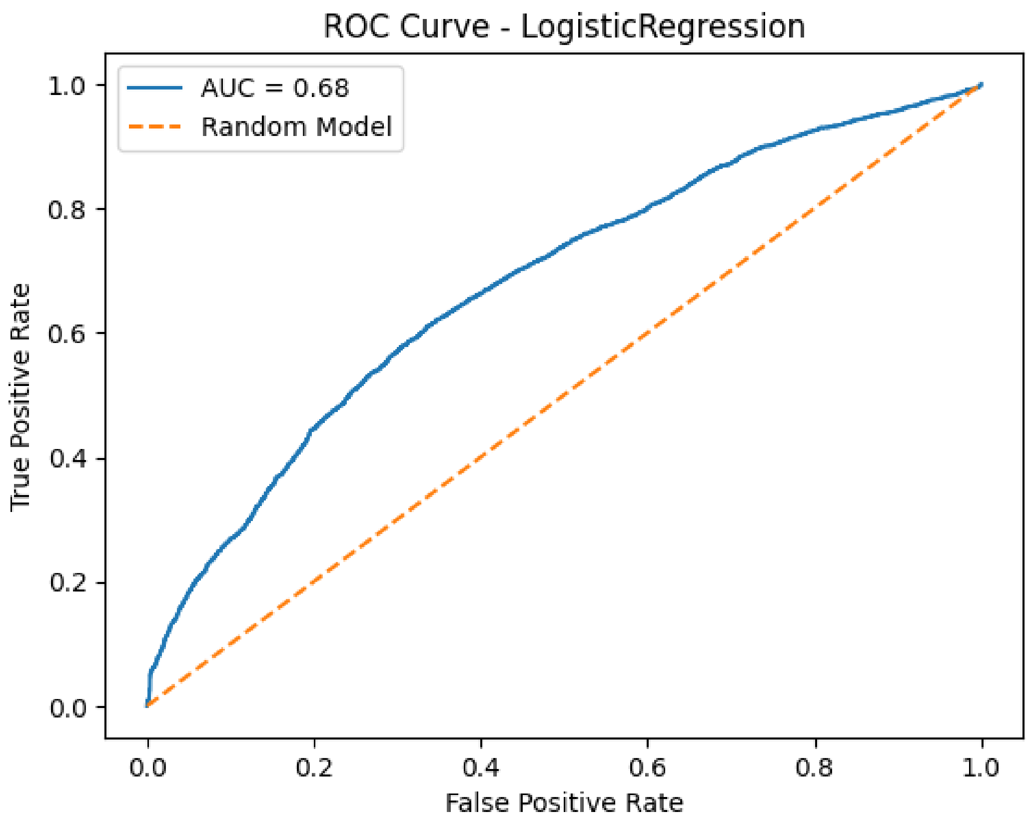
```
def evaluate_model(model, X_test, y_test):
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]
if len(y_prob) != len(y_test):
    print(f"Shape mismatch: y_prob has {len(y_prob)} elements and y_test has {len(y_test)} elements.")
    return
if np.any(np.isnan(y_prob)) or np.any(np.isinf(y_prob)):
    print("Warning: y_prob contains NaN or infinite values!")
    return
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
precision = precision_score(y_test, y_pred)
print(f"Precision (PPV): {precision:.2f}")
```

```
    recall = recall_score(y_test, y_pred)
    print(f"Recall (Sensitivity): {recall:.2f}")
    tn, fp, fn, tp = cm.ravel()
    specificity = tn / (tn + fp)
    print(f"Specificity: {specificity:.2f}")
    npv = tn / (tn + fn)
    print(f"NPV: {npv:.2f}")
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    if len(fpr) != len(tpr):
        print(f"ROC Curve: fpr and tpr have mismatched lengths. fpr: {len(fpr)}, tpr: {len(tpr)}")
        return
    roc_auc = auc(fpr, tpr)
    print(f"ROC AUC: {roc_auc:.2f}")
    plt.plot(fpr, tpr, label=f'AUC = {roc_auc:.2f}')
    plt.plot([0, 1], [0, 1], linestyle='--', label='Random Model')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'ROC Curve - {model.__class__.__name__}')
    plt.legend()
    plt.show()
    plot_lift_gain_curve(model, X_test, y_test)
    return accuracy, precision, recall, specificity, npv, roc_auc
def plot_lift_gain_curve(model, X_test, y_test):
    y_prob = model.predict_proba(X_test)[:, 1]
    precision, recall, thresholds = precision_recall_curve(y_test, y_prob)
    gain = recall
    baseline = np.linspace(0, 1, len(gain))
```

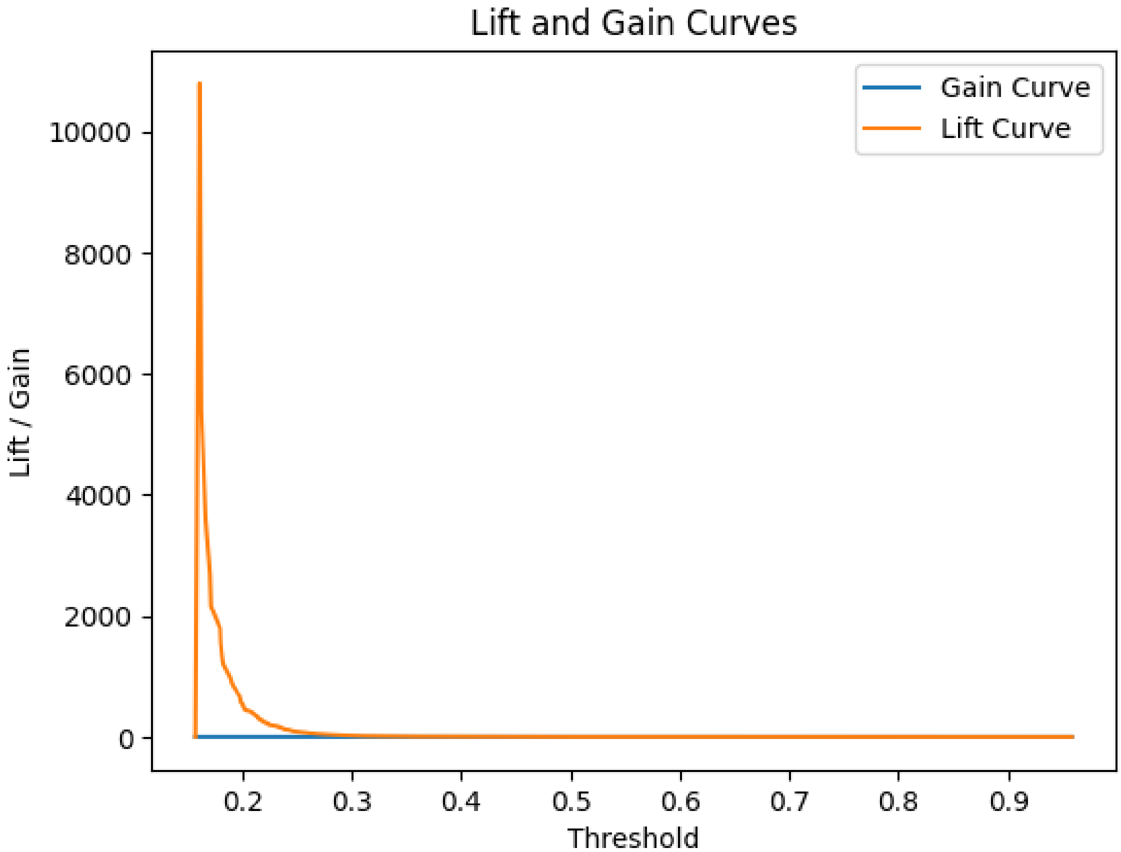
```
lift = np.divide(gain, baseline, out=np.zeros_like(gain), where=baseline != 0)

plt.plot(thresholds, gain[:-1], label="Gain Curve")

plt.plot(thresholds, lift[:-1], label="Lift Curve")

plt.xlabel('Threshold')

plt.ylabel('Lift / Gain')

plt.title('Lift and Gain Curves')

plt.legend()
```

```
plt.show()
```

```
Evaluating LogisticRegression model...
Confusion Matrix:
[[2854 2055]
 [1895 4003]]
Accuracy: 0.63
Precision (PPV): 0.66
Recall (Sensitivity): 0.68
Specificity: 0.58
NPV: 0.60
ROC AUC: 0.68
```

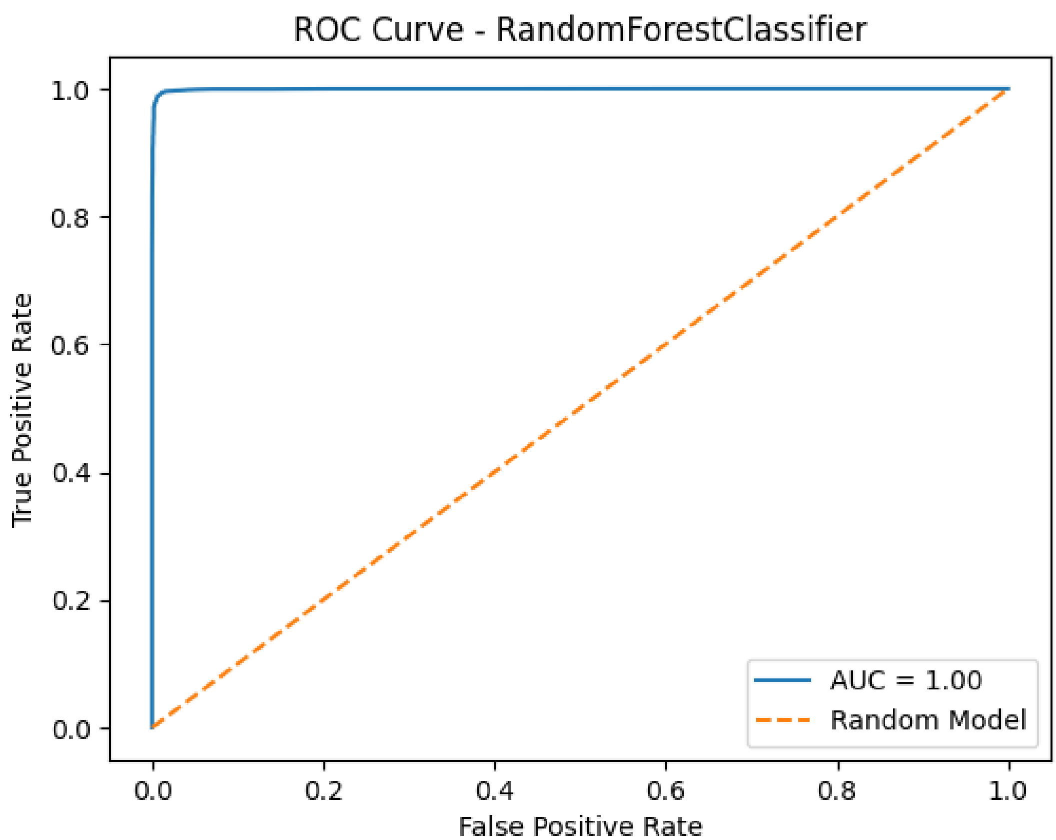


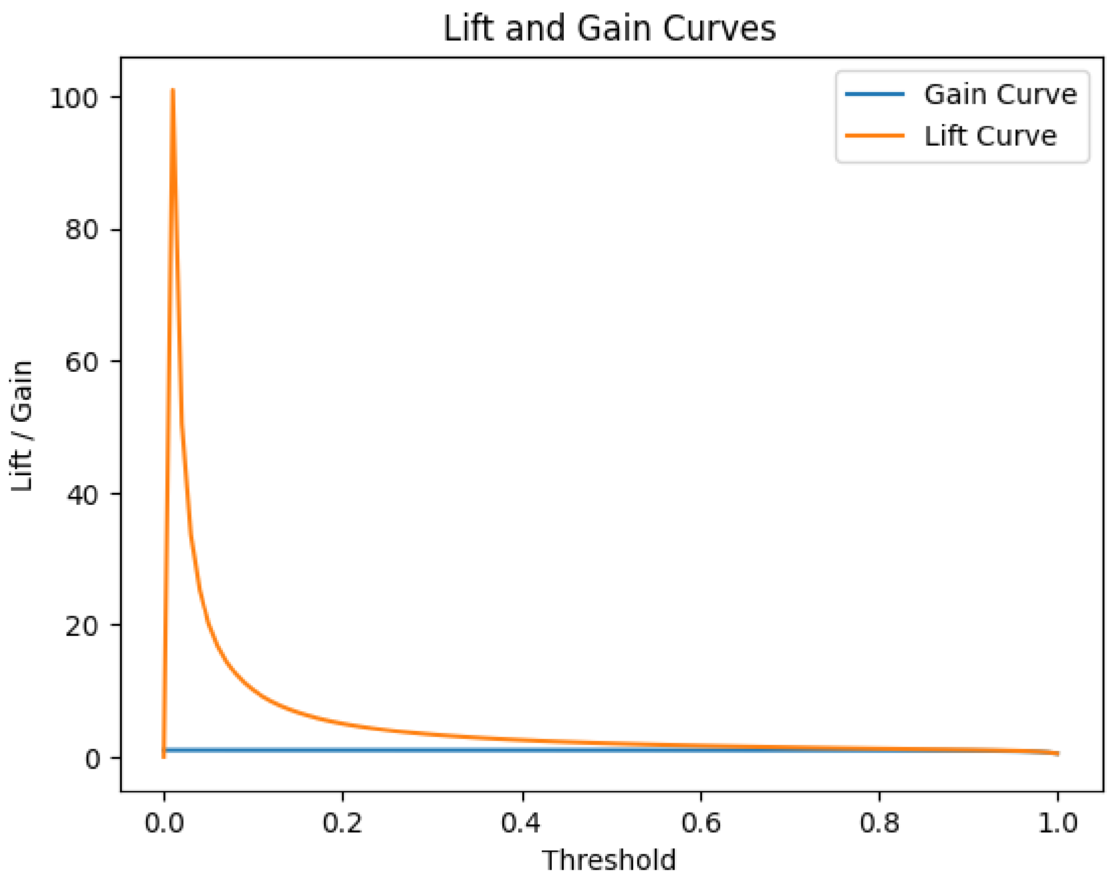
SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)



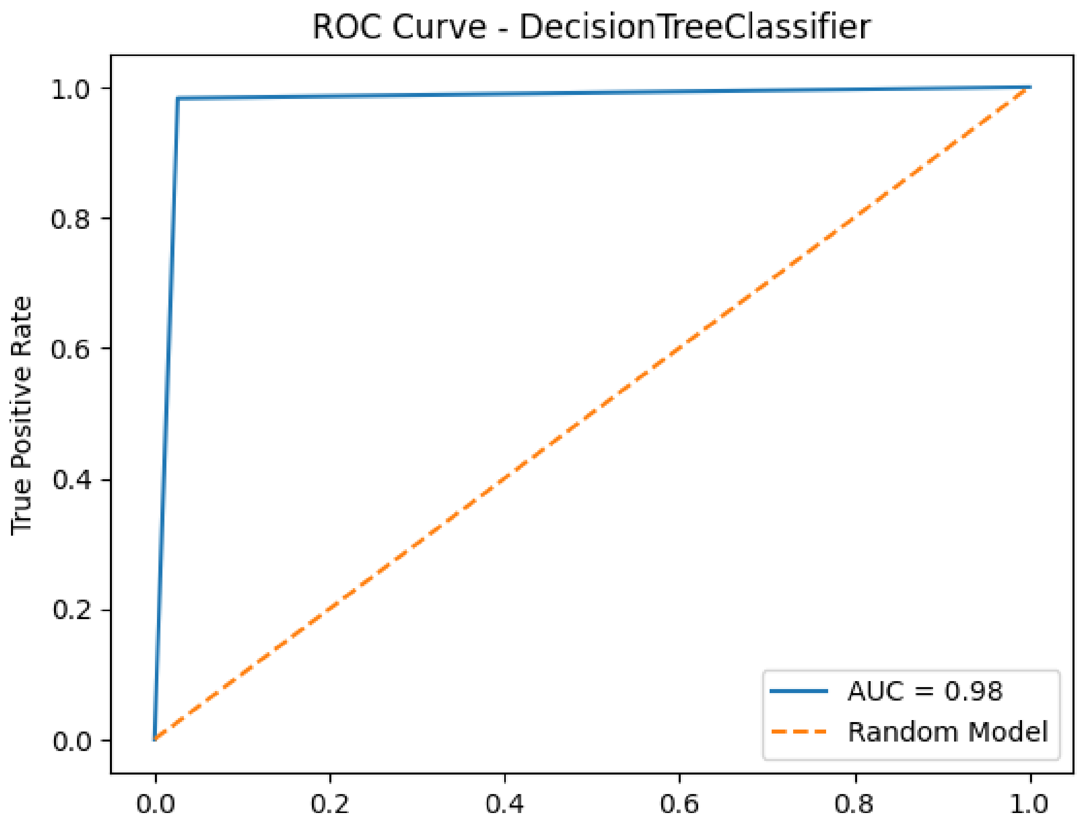
SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

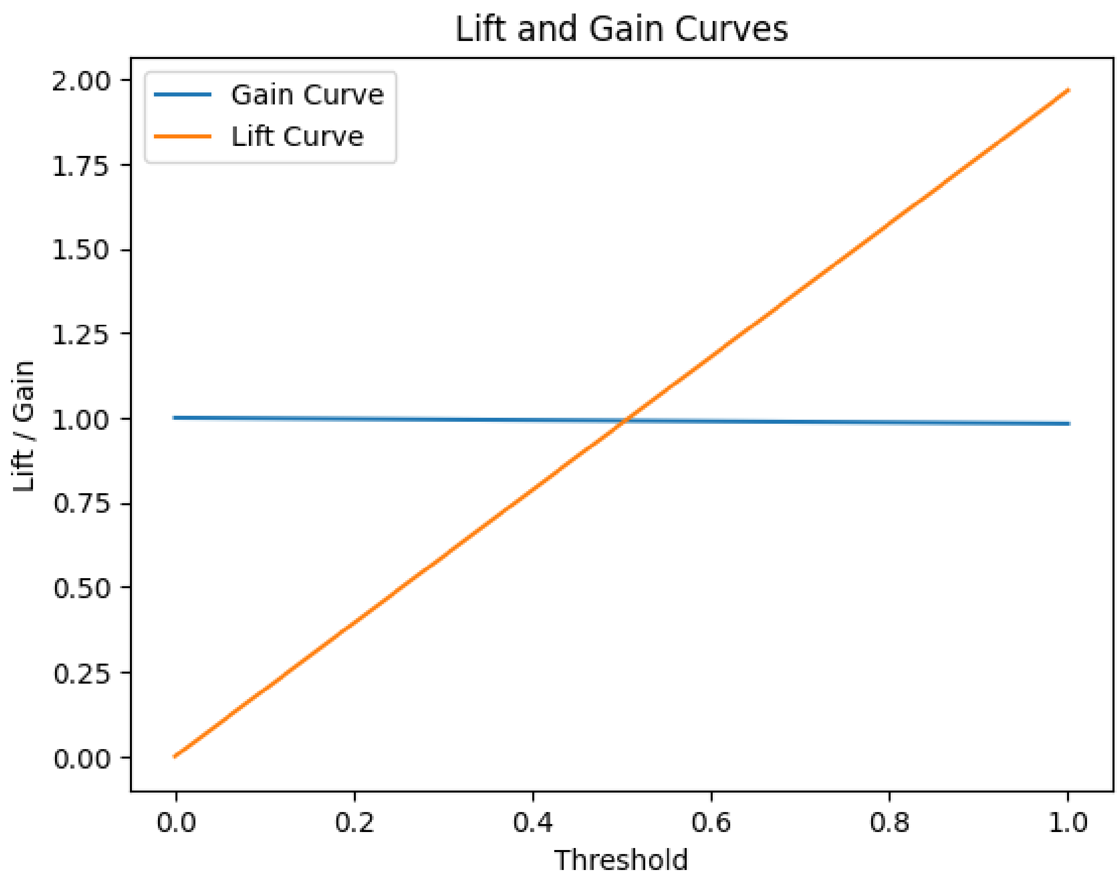
```
Evaluating RandomForestClassifier model...
Confusion Matrix:
[[4854   55]
 [  41 5857]]
Accuracy: 0.99
Precision (PPV): 0.99
Recall (Sensitivity): 0.99
Specificity: 0.99
NPV: 0.99
ROC AUC: 1.00
```




Lift and Gain Curves
Gain Curve
Lift Curve
Lift / Gain
100
80
60
40
20
0
0.0
0.2
0.4
0.6
0.8
1.0
Threshold

```
Evaluating DecisionTreeClassifier model...
Confusion Matrix:
[[4778  131]
 [ 101 5797]]
Accuracy: 0.98
Precision (PPV): 0.98
Recall (Sensitivity): 0.98
Specificity: 0.97
NPV: 0.98
ROC AUC: 0.98
```






```
models = [lr_model, rf_model, dt_model]

for model in models:

   print(f"\nEvaluating {model.__class__.__name__} model...")

   evaluate_model(model, X_test, y_test)
```

CONCLUSION:

Base all conclusions on your actual results; describe the meaning of the experiment and the implications of your results.

REFERENCES:

(List the references as per format given below and citations to be included the document)

1. Ethem Alpaydın, “Introduction to Machine Learning”, 4th Edition, The MIT Press, 2020.

2. Peter Harrington, “Machine Learning in Action”, 1st Edition, Dreamtech Press, 2012.

3. Tom Mitchell, “Machine Learning”, 1st Edition, McGraw Hill, 2017.

4. Andreas C, Müller and Sarah Guido, “Introduction to Machine Learning with Python: A Guide for Data Scientists”, 1st Edition, O'reilly, 2016.

5. Kevin P. Murphy, “Machine Learning: A Probabilistic Perspective”, 1st Edition, MIT Press, 2012.

Website References:

[1]https://in.mathworks.com/campaigns/offers/next/choosing-the-best-machine-learning-classification-model-and-avoiding-overfitting.html

[2] https://developers.google.com/machine-learning/crash-course/classification

[3] https://builtin.com/data-science/supervised-machine-learning-classification

[4] https://www.sciencedirect.com/topics/computer-science/classification-machine-learning