College of Computer Science and Engineering
Department of Computer Science and Artificial Intelligence
CCAI-312: Pattern Recognition

CCAI312 Group Project

(Fall 2024)

Student Name
Jameel Rami Mahjub
Abdulrahman sami tibagi
Abdulaziz Nasrallah

For grading use only:

PLO/CLO	SO
PLO C4 (CLO 3.1): Discover and apply new knowledge as needed, using appropriate learning strategies	**SO 7:** An ability to acquire and apply new knowledge as needed, using appropriate learning strategies.

Course instructor:

DR. Mohammed AL-Dessouky

		Max Score	Student 1 score	Student 2 score	Student 3 score
PLO C4 / CLO 3.1 / SO 7	**Task**	**10**			
Total					

College of Computer Science and Engineering
Department of Computer Science and Artificial Intelligence
CCAI-312: Pattern Recognition

Contents:

College of Computer Science and Engineering
Department of Computer Science and Artificial Intelligence
CCAI-312: Pattern Recognition

Introduction:

Bankruptcy prediction is a critical financial analysis task that can help stakeholders, including investors, creditors, and policymakers, take preemptive actions to mitigate potential risks. In this project, we leverage machine learning techniques to predict whether a company is likely to go bankrupt based on its financial data. By applying machine learning to this real-world problem, we aim to develop a robust model capable of identifying key financial health indicators and predicting a company's operational status as either "alive" or "bankrupt."

Problem Definition:

The challenge lies in the identification of companies at risk of bankruptcy before they reach that stage. Many businesses fail due to a lack of early warning signs or proper financial management. This project seeks to build a binary classification model that predicts whether a company will remain operational or go bankrupt, using financial metrics and machine learning algorithms. The goal is to enhance decision-making by providing accurate and timely insights into a company's financial health.

Dataset Overview

The dataset, sourced from Kaggle's American Bankruptcy data, consists of 76,682 records and 20 financial features, such as current_asset, cost_goods_sold, depreciation_amortization, earnings_before_interest_task, inventory, net_income, total_receivables, market_value, net_sales, total_assets, total_long_term_dept, EBIT, grosse_profit, total_carrunt_liabilities, retained_earnings, total_revenue, total_liabilities, total_operating.

Training and Preprocessing:

The target variable, status_label, classifies companies as either "alive" (74,462 cases) or "failed" (5,220 cases). The data was split into training (80%) and testing (20%) sets after undergoing preprocessing steps like cleaning, transformation, and dimensionality reduction.

Algorithm Proposal:

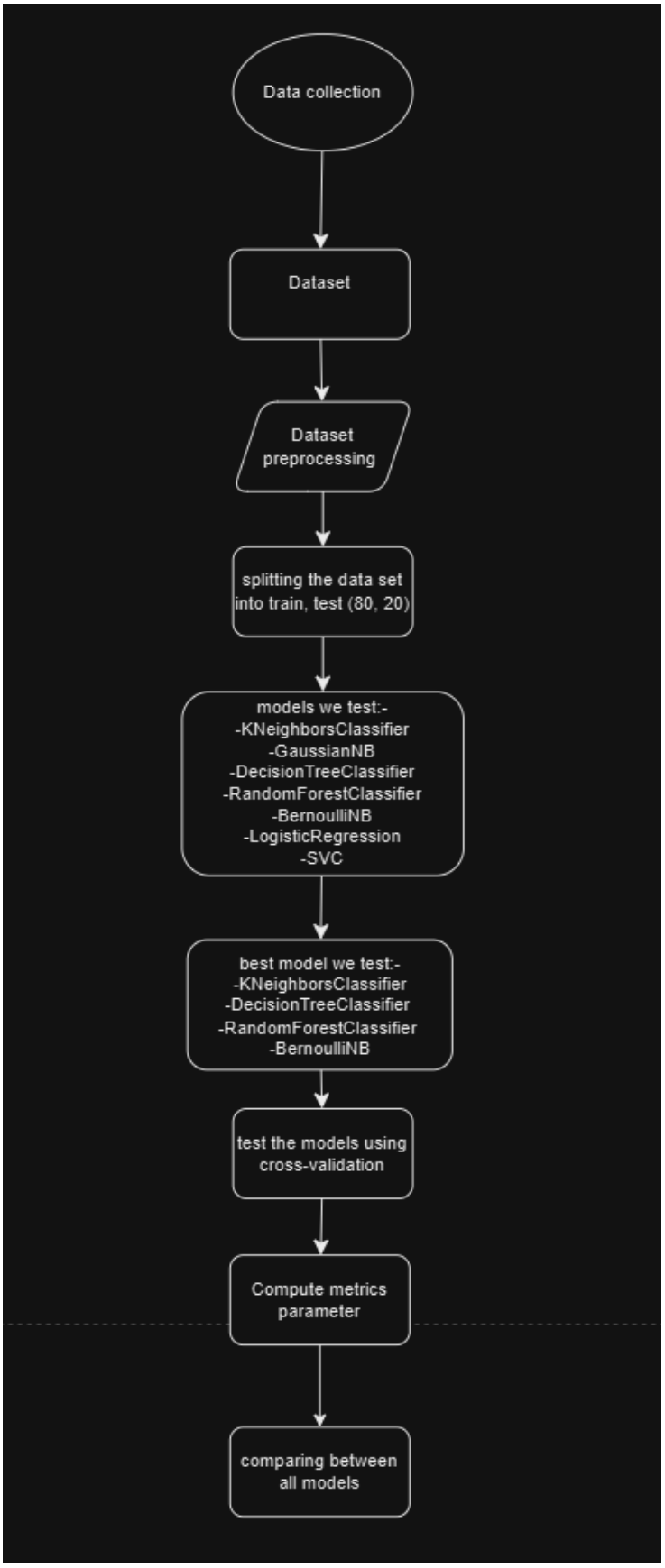
We select this algorithm Random Forest and we will compare it with 5 algorithm:

- Performs well with large and imbalanced datasets. Handles overfitting better than a single decision tree and can use class weights to address imbalance.

College of Computer Science and Engineering

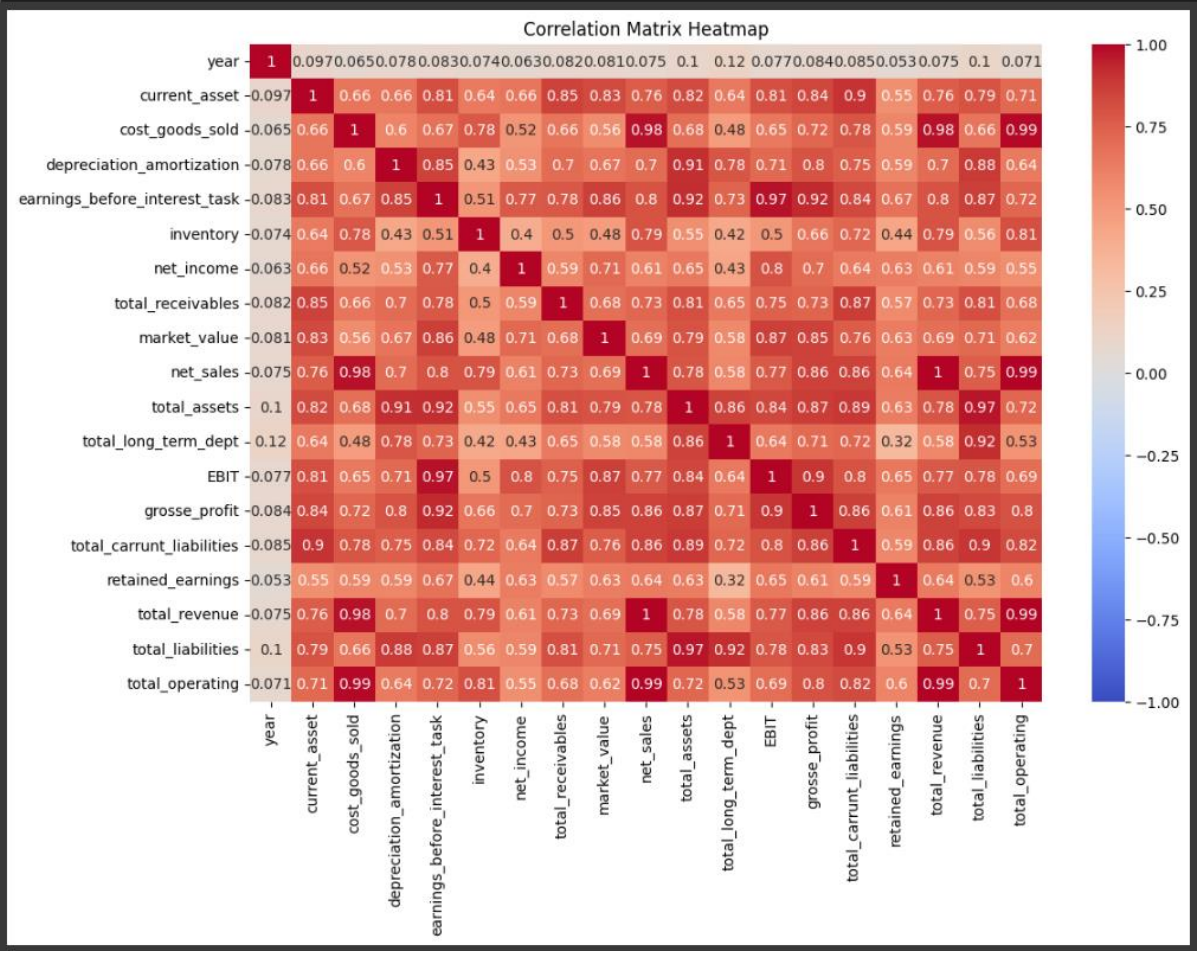
Department of Computer Science and Artificial Intelligence

CCAI-312: Pattern Recognition



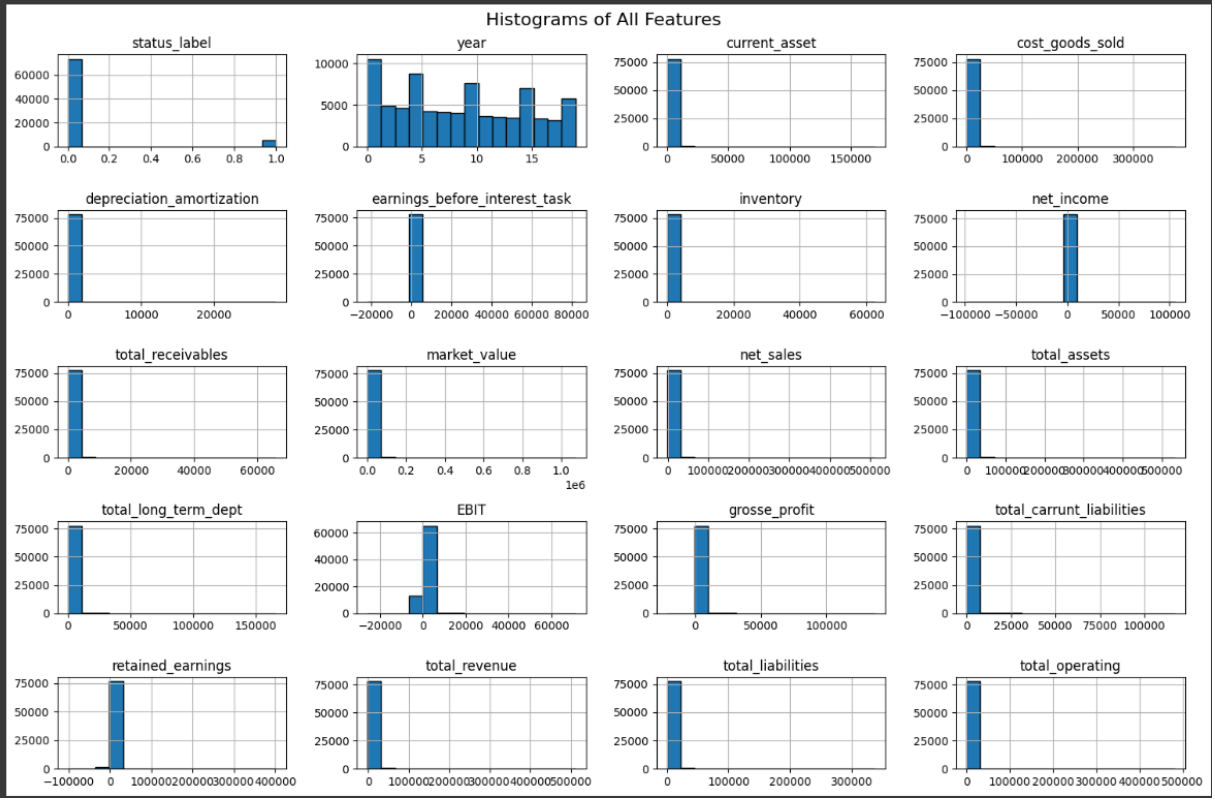
Correlation matrix :

```
## correlation

import matplotlib.pyplot as plt
import seaborn as sns

corr_matrix = x.corr()

# Create a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Matrix Heatmap')
plt.show()
```

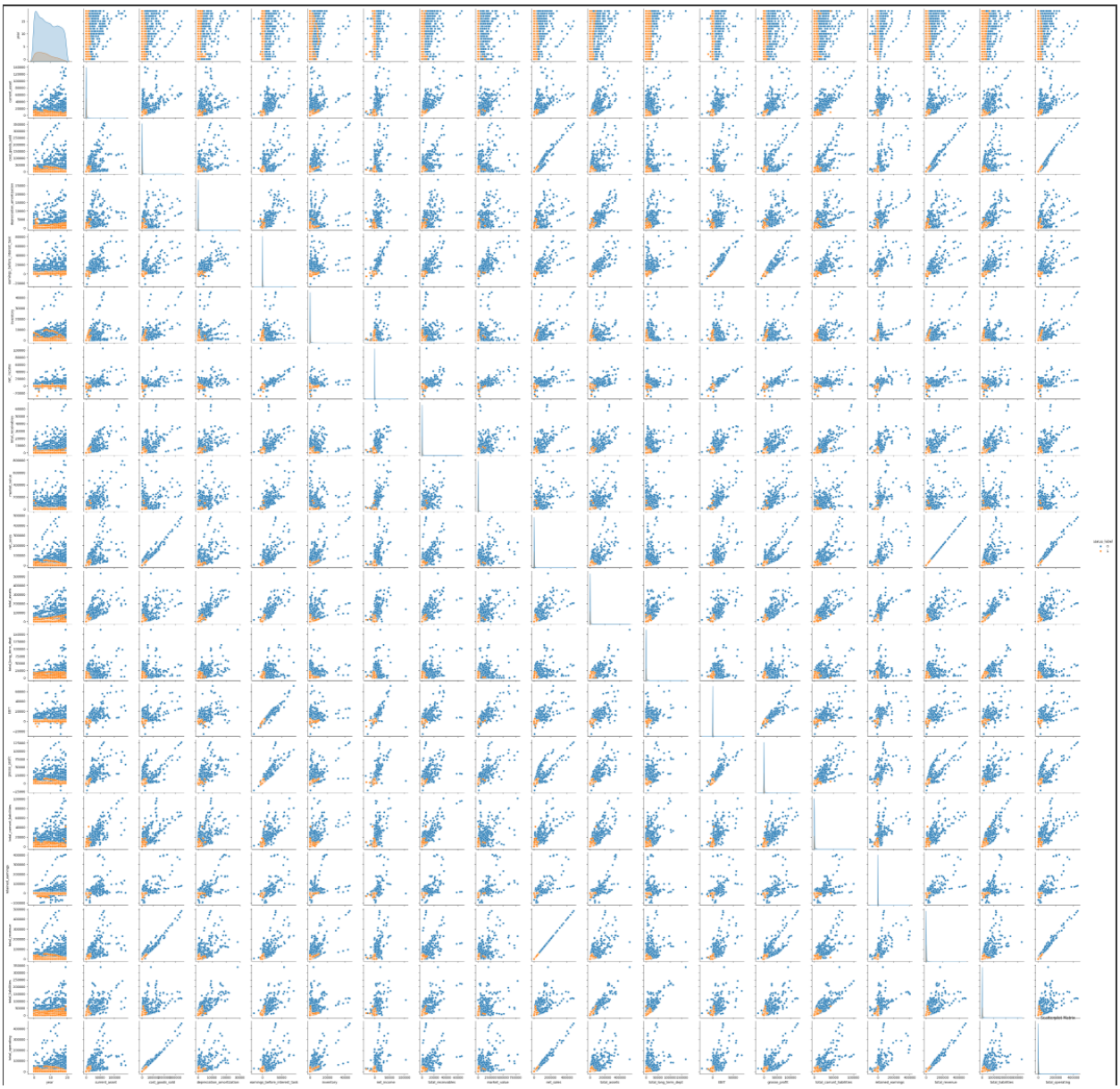


Histogram we see her imbalance class :

```
## histogram

data.hist(bins=15, figsize=(15, 10), edgecolor='black')
plt.suptitle('Histograms of All Features', fontsize=16)
plt.tight_layout()
plt.show()
```



Scatter Plot we see here overlapping with small class:



College of Computer Science and Engineering
Department of Computer Science and Artificial Intelligence
CCAI-312: Pattern Recognition

Evaluation and model Tuning:

Model Testing and Selection

The analysis begins with testing multiple machine learning models, including K-Nearest Neighbors (KNN), Gaussian Naive Bayes (GaussianNB), Decision Tree, Random Forest, Bernoulli Naive Bayes (BernoulliNB), Logistic Regression, and Support Vector Classifier (SVC). These models were initially evaluated using the full dataset and then retested after reducing the "alive" class to 35,000 instances for better balance.

Evaluation Process

- **Initial Testing (Imbalanced Data):**
  Using all data, Random Forest, KNN, and BernoulliNB models achieved high accuracy (93%), while the Decision Tree and GaussianNB showed slightly lower performance. However, due to the class imbalance, precision, recall, and F1-scores were affected.
- **Balanced Testing (Reduced "Alive" Class):**
  After reducing the majority class, the models were reevaluated, revealing improved metric balance.

Algorithm Comparison and Discussing:

All data	Cros-validation accuracy	precision	Recall	F-score	Confusion matrix
RandomForestClassifier	93%	0.93	0.53	0.55	[[22056 8] [1439 102]]
KNeighborsClassifier	93%	0.83	0.51	0.50	[[22056 8] [1519 22]]
DecisionTreeClassifier	88%	0.61	0.62	0.61	[[20865 1199] [1101 441]]
BernoulliNB	93%	0.47	0.50	0.48	[[22064 0] [1541 0]]

reduce class alive to 35,000	Cros-validation accuracy	precision	Recall	F-score	Confusion matrix
RandomForestClassifier	86%	0.76	0.69	0.72	[[10073 416] [925 652]]
KNeighborsClassifier	87%	0.76	0.52	0.52	[[10440 49] [1491 86]]
DecisionTreeClassifier	81%	0.64	0.64	0.64	[[9431 1058] [961 616]]
BernoulliNB	86%	0.43	0.50	0.47	[[10489 0] [1577 0]]

All Data:

- Random Forest and K-Neighbors both achieved the highest cross-validation accuracy (93%), with Random Forest having better precision (0.93 vs. 0.83) and F-score (0.55 vs. 0.50).

Minimal false positives, but a relatively high number of false negatives impacts recall.

- Decision Tree had lower accuracy (88%) but balanced recall (0.62) and F-score (0.61).

The Decision Tree misclassifies more of both classes compared to Random Forest and K-Neighbors.

- Bernoulli Naive Bayes also achieved 93% accuracy but had poor precision (0.47) and F-score (0.48).

This classifier is unsuitable for imbalanced datasets since it completely ignores the minority class.

Reduced Data:

- Random Forest had the highest F-score (0.72) and balanced recall (0.69).

Performs well even with reduced data, maintaining a strong trade-off between precision and recall.

- K-Neighbors maintained competitive accuracy (87%) but lower F-score (0.52).

Performs worse than Random Forest due to its struggle with the minority class.

- Decision Tree performed reasonably (81% accuracy and 0.64 F-score).

Handles minority class reasonably but sacrifices performance on the majority class.

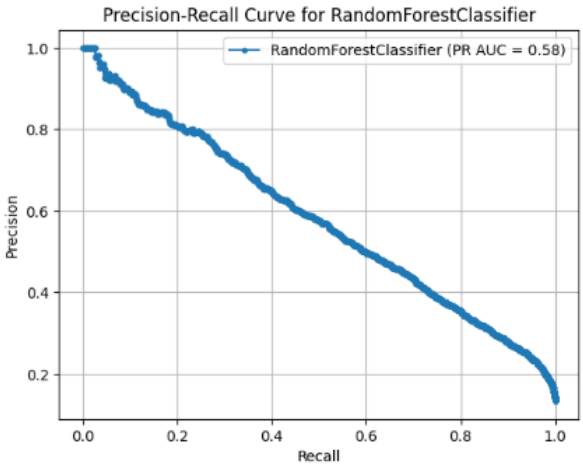
- Bernoulli Naive Bayes struggled with low F-score (0.47), despite an 86% accuracy.

Completely unsuitable for imbalanced datasets, even with reduced class size.

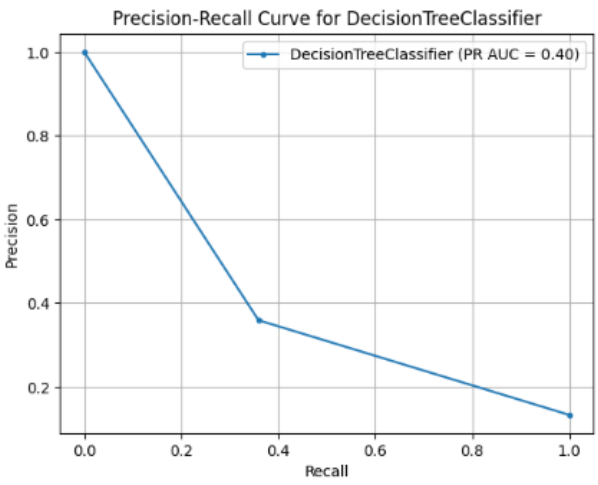
Best to Worst Classifiers:

1. **Random Forest Classifier: Best overall performance across both datasets with the highest F-scores and strong precision-recall balance.**
2. **K-Neighbors Classifier: Competitive accuracy but weaker F-scores compared to Random Forest.**
3. **Decision Tree Classifier: Lower accuracy but decent recall and F-score, suitable if interpretability is prioritized.**
4. **Bernoulli Naive Bayes: Poor precision and F-score, especially on imbalanced datasets, making it the least reliable option here.**

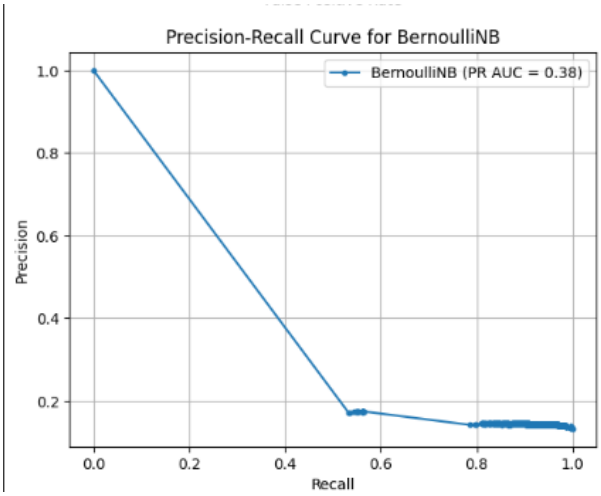
Why Random Forest? It consistently provides the best precision-recall tradeoff, high accuracy, and robust F-score, making it the most reliable classifier for dataset.



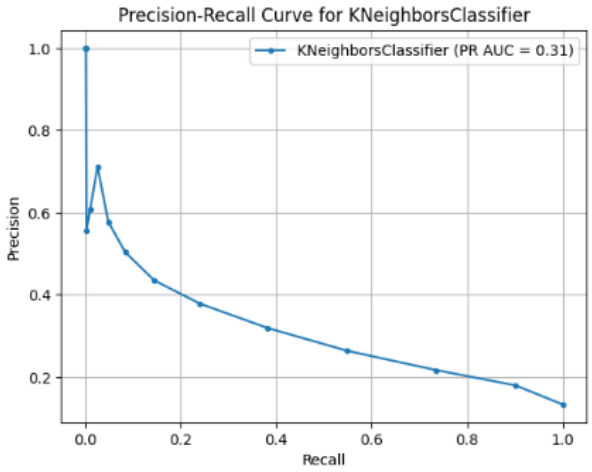
- **PR AUC = 0.58**: The best performer among the models shown.
- **Observation**: The curve is smoother and maintains better precision as recall increases. This suggests that RandomForest is better at balancing precision and recall compared to the other classifiers.



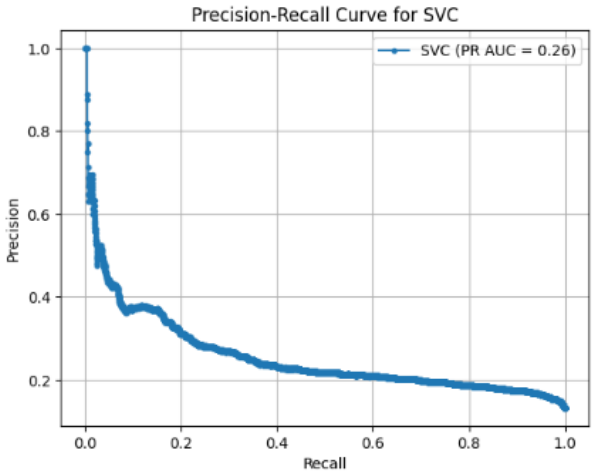
- **PR AUC = 0.40**: Performs moderately better than GaussianNB and KNeighborsClassifier.
- **Observation**: The curve has a steep drop, showing the model can achieve high precision for a small recall. However, as it captures more positive instances, precision declines significantly.



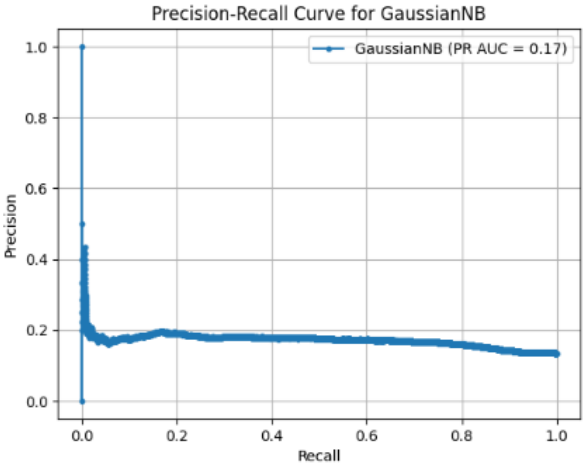
- **PR AUC = 0.38**: Performs slightly better than GaussianNB but worse than DecisionTree and RandomForest.
- **Observation**: Precision declines steeply as recall increases, indicating that the model is less confidentabout its positive predictions at higher recall values.



- **PR AUC = 0.31**: The model struggles to maintain high precision and recall simultaneously, indicating weak performance.
- **Observation**: Precision drops significantly as recall increases. This suggests that the classifier's ability to identify positive instances accurately diminishes when it tries to capture more positives.



- **PR AUC = 0.26**: The curve shows suboptimal performance.
- **Observation**: The curve declines sharply, and precision is consistently low, suggesting that the SVC model struggles to identify positive cases effectively in this scenario.

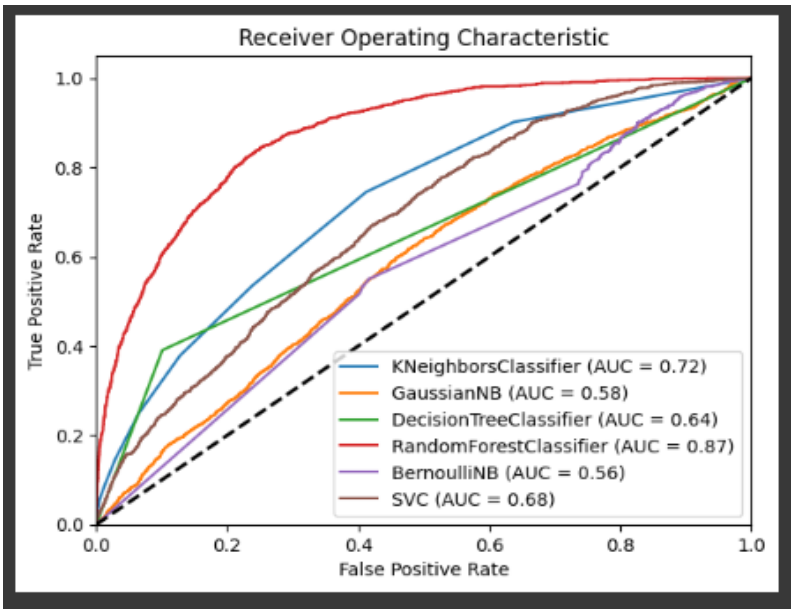


- **PR AUC = 0.17**: The curve indicates poor performance.

Observation: Precision is very low throughout most of the curve, and recall increases only slightly. This suggests that the model has difficulty distinguishing between classes, resulting in many false positives **.**

College of Computer Science and Engineering
Department of Computer Science and Artificial Intelligence
CCAI-312: Pattern Recognition

Evaluation and Analysis:



- This figure shows the **Receiver Operating Characteristic (ROC) Curves** for six classifiers, along with their respective **Area Under the Curve (AUC)** values. The ROC curve represents the trade-off between the **True Positive Rate (TPR)** (y-axis) and the **False Positive Rate (FPR)** (x-axis) at different classification thresholds.

1. RandomForestClassifier

- **AUC = 0.87**: The best performer among all models, with a curve close to the top-left corner.
- **Observation**: This classifier has the highest ability to distinguish between positive and negative classes, making it the most effective in this scenario.

---

2. KNeighborsClassifier

- **AUC = 0.72**: Performs well, though less effective than RandomForestClassifier.
- **Observation**: The curve stays relatively close to the top-left corner but starts to deviate at higher FPR values, indicating reduced effectiveness at higher thresholds.

---

3. SVC (Support Vector Classifier)

- **AUC = 0.68**: Performs moderately, slightly better than DecisionTreeClassifier.
- **Observation**: The curve shows decent performance but is farther from the ideal top-left corner than the top classifiers.

---

4. DecisionTreeClassifier

- **AUC = 0.64**: Moderate performance.
- **Observation**: The curve is slightly above the diagonal (random baseline), showing limited ability to distinguish between classes effectively.

---

5. GaussianNB

- **AUC = 0.58**: Performs poorly compared to most other classifiers.
- **Observation**: The curve is close to the diagonal, indicating that its predictions are only marginally better than random guessing.

---

6. BernoulliNB

- **AUC = 0.56**: The poorest performer in this evaluation.
- **Observation**: The curve is close to the diagonal (random baseline), indicating a very weak classification performance.

General Insights

1. **Best Algorithm**: **RandomForestClassifier** (AUC = 0.87), with a strong ability to balance TPR and FPR across thresholds.
2. **Moderate Performers**: **KNeighborsClassifier** (AUC = 0.72) and **SVC** (AUC = 0.68) show reasonable performance.
3. **Poor Performers**: **GaussianNB** (AUC = 0.58) and **BernoulliNB** (AUC = 0.56) have the weakest ability to separate classes.

Comparing PR Curve with ROC Curve and accuracy:

- **RandomForestClassifier** performs best in all evaluations.
- **We can use K-Neighbors Classifier (KNN)**
    - When you need to rapidly evaluate performance or test predictions.
    - When computational resources or time are limited.
- Some algorithms, like BernoulliNB and GaussianNB, perform poorly in metrics.

Algorithm Explanation:

Explain each algorithm we use and parameter :

1. K-Nearest Neighbors (KNN)

- **How it works**:
  - KNN classifies a data point by identifying the majority class of its KKK nearest neighbors in the feature space.
- **Key Parameters**:
  - K: The number of nearest neighbors to consider.
  - Distance metric: Method for calculating distance (e.g., Euclidean, Manhattan, Minkowski).
  - Weighting: Whether closer neighbors are given more influence (e.g., uniform or distance-weighted).

```
KNeighborsClassifier(n_neighbors=15),
```

---

2. Gaussian Naive Bayes (GaussianNB)

- **How it works**:
  - GaussianNB uses Bayes' Theorem and assumes that the features follow a Gaussian (normal) distribution.
- **Key Parameters**:
  - Priors: Optional array specifying the prior probabilities of each class.
  - Variance smoothing: A small value added to variances to stabilize computations.

```
GaussianNB(),
```

---

3. Decision Tree

- **How it works**:
    - A Decision Tree splits the dataset recursively based on feature values to build a tree that predicts class labels at leaf nodes.
- **Key Parameters**:
    - Criterion: Metric to measure the quality of a split (e.g., "gini" for Gini Impurity or "entropy" for Information Gain).
    - Max depth: The maximum depth of the tree.
    - Min samples split: The minimum number of samples required to split an internal node.
    - Min samples leaf: The minimum number of samples required to be at a leaf node.

```
DecisionTreeClassifier(random_state=42),
```

---

4. Random Forest

- **How it works**:
    - Random Forest builds multiple Decision Trees and aggregates their predictions to make a final decision.
- **Key Parameters**:
    - N estimators: The number of trees in the forest.
    - Max features: The number of features to consider for the best split.
    - Max depth: The maximum depth of the individual trees.
    - Min samples split: The minimum number of samples required to split a node.
    - Min samples leaf: The minimum number of samples required to be at a leaf node.
    - Bootstrap: Whether to use bootstrap samples when building trees.

```
RandomForestClassifier(n_estimators=250,
max_depth=20,
min_samples_split=10,
min_samples_leaf=4,
max_features='log2',
bootstrap=False,
class_weight='balanced',
random_state=42,
n_jobs=-1),
```

---

5. Bernoulli Naive Bayes (BernoulliNB)

- **How it works**:
  - BernoulliNB assumes binary features and calculates probabilities for each class based on their presence or absence.
- **Key Parameters**:
  - Alpha: Smoothing parameter to avoid zero probabilities.
  - Binarize: Threshold for binarizing features (default is 0, meaning binary features are assumed).
  - Fit prior: Whether to learn class prior probabilities from the data.

```
BernoulliNB(),
```

---

6. Logistic Regression

- **How it works**:
  - Logistic Regression uses a linear combination of features and a sigmoid function to predict probabilities of class membership.
- **Key Parameters**:
  - Penalty: Regularization type (e.g., "l1", "l2", "elasticnet").
  - C: Inverse of regularization strength (smaller values specify stronger regularization).
  - Solver: Algorithm to optimize the model (e.g., "liblinear", "lbfgs", "saga").
  - Max iter: Maximum number of iterations for the solver to converge.

```
    LogisticRegression(
    penalty='l2',
    C=1.0,
    class_weight='balanced',
    solver='lbfgs',
    random_state=42
),
```

---

7. Support Vector Classifier (SVC)

- **How it works**:
    - SVC finds the hyperplane that separates classes with the maximum margin, using kernel functions for non-linear data.
- **Key Parameters**:
    - C: Regularization parameter; trades off margin size and classification error.
    - Kernel: Specifies the kernel type (e.g., "linear", "rbf", "poly").
    - Gamma: Kernel coefficient for "rbf", "poly", and "sigmoid" kernels.
    - Degree: Degree of the polynomial kernel (used only for "poly").
    - Tolerance: Stopping criterion for optimization.

```
    SVC(
    kernel='rbf',
    C=1.0,
    gamma='scale',
    class_weight='balanced',
    probability=True,
    random_state=42
)
```

This all best parameter we use .