SKILL LAB AND PROJECT-I

TITLE:

Predicting The Likelihood Of Divorce using Machine Learning

Group Members:

Manaswini Mohapatra Roll 16 A1 SIC- 22BCEE69

Akankshya Pattnaik Roll 22 A1 SIC-22BCEF36

Mohit Prasad Mohanty Roll 23 A3 SIC-22BECF64

Abismruta Kar Roll 29 A3 SIC-22BCEE46

Guided By:

Dr. Rekha Sahu

(Asst. Professor, Dept. of CSE)

CONTENTS

Preface

The project, "Predicting the Likelihood of Divorce Using Machine Learning Algorithms," explores the use of advanced computational techniques to address a critical social issue—marital stability. With rising divorce rates globally, understanding the factors contributing to relationship dissolution has become more important than ever. This project leverages machine learning to analyze relationship dynamics and predict the likelihood of divorce, providing valuable insights for early intervention and counseling.

Divorce is a significant social issue that affects families, children, and communities worldwide. Understanding the factors contributing to divorce can help identify warning signs early and enable timely interventions, such as counseling or therapy. Traditional methods of analyzing relationship dynamics rely on manual surveys and subjective assessments, which can be time-consuming and prone to bias.

This project is necessary because it uses **machine learning algorithms** to provide data-driven insights into marital stability. By analyzing psychological, emotional, and behavioral patterns from relationship data, machine learning models can predict the likelihood of divorce with high accuracy. Such predictions can help individuals and counselors make informed decisions to improve relationships or address underlying issues.

Additionally, this project highlights how technology can be used to address critical societal challenges. It bridges the gap between computational tools and human relationships, making it possible to scale solutions and provide personalized support to couples. Ultimately, this project demonstrates the power of machine learning in promoting healthier relationships and fostering social well-being.

Objective

The primary objective of this project is to leverage the power of machine learning to predict the likelihood of divorce among couples by analyzing a comprehensive dataset containing demographic, socio-economic, and relationship-related factors. Divorce is a complex social issue with far-reaching personal and societal consequences, making it critical to identify the underlying causes and provide data-driven insights for early intervention strategies.

This study focuses on utilizing various supervised learning algorithms, including **K-Nearest Neighbors (KNN)**, **Logistic Regression**, **Support Vector Machines (SVM)**, **Naïve Bayes**, and **Decision Trees**, to build predictive models capable of accurately identifying at-risk relationships. By examining diverse relationship attributes such as communication patterns, trust levels, and emotional well-being, the models aim to uncover significant factors contributing to marital instability.

The project emphasizes rigorous evaluation of the machine learning models using key performance metrics such as **Accuracy**, **Precision**, **Recall**, **F1-score**, **Confusion Matrix**, **Classification Report**, and the **ROC-AUC Curve**. These metrics provide a comprehensive understanding of the models' predictive performance and their ability to generalize to new, unseen data.

Ultimately, this project aspires to not only identify the best-performing model but also contribute valuable insights into relationship dynamics. These findings can guide the development of tools and strategies aimed at fostering healthier relationships, improving marital outcomes, and mitigating the risks of divorce.

Tools and Techniques

This project leverages various tools and techniques to implement and evaluate the stock price prediction models:

- **Programming Language:**

Python, chosen for its versatility and extensive library support.

- **Libraries:**

Pandas: Used for data manipulation and preprocessing, enabling efficient handling of large datasets.

NumPy: Provides numerical computation tools for efficient array operations and mathematical functions.

Matplotlib: Enables the creation of static, animated, and interactive visualizations, aiding in data exploration.

Scikit-learn: Includes machine learning tools for implementing Linear Regression, SVR models, and evaluation metrics.

- **Environment:**

Jupyter Notebook: An interactive platform for coding, analysis, and visualization.

Dataset Description

The "Predicting Divorce" dataset is a structured collection of relational and psychological features derived from survey responses, aimed at analyzing factors that influence divorce likelihood. Each record corresponds to an individual or a couple, capturing vital aspects of their relationship dynamics, such as communication quality, emotional satisfaction, and conflict resolution. This dataset is well-suited for binary classification tasks, where the target variable distinguishes between stable relationships (0) and those likely to end in divorce (1).

Features and Attributes

1. **Relational Attributes**:
    - Metrics assessing key relationship factors, including trust, communication, and conflict resolution strategies, providing insights into interpersonal dynamics.
2. **Psychological Indicators**:
    - Features highlighting emotional connection, satisfaction levels, and partner compatibility, which are pivotal for understanding marital stability.
3. **Target Variable**:
    - A binary outcome representing the likelihood of divorce (0 = stable, 1 = divorce), which serves as the predictive target for machine learning models.

Dataset Size and Distribution

The dataset contains 170 records (observations) and 54 attributes (columns). Each record represents a couple's responses to structured survey questions, with the columns detailing relational and psychological aspects alongside the target variable. Its compact size allows for quick exploratory analysis and efficient training of lightweight machine learning models, making it an excellent resource for academic research and experimental applications.

Methods

K-Nearest Neighbors (KNN)

KNN is a simple and intuitive machine learning algorithm that classifies data points based on the majority class of their 'k' nearest neighbors in the feature space. It is non-parametric, meaning it makes no assumptions about the underlying data distribution, making it particularly effective for small datasets. The algorithm uses distance metrics, such as Euclidean distance, to identify the closest neighbors.

Euclidean Distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Where:

- x, y are data points
- x_i, y_i are the feature values of x and y

Logistic Regression

Logistic Regression is a statistical model designed for binary classification tasks. It predicts the probability of an outcome belonging to one of two classes using the sigmoid (logistic) function. The function maps input features to probabilities, ensuring values remain between 0 and 1, making it ideal for problems where the target variable is binary.

Sigmoid Function:

$$P(y = 1|X) = \frac{1}{1 + e^{-z}}$$

Where:

$$z = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

- z: Linear combination of weights (w_i) and features (x_i)
- $P(y = 1|X)$: Probability of the outcome being 1

The decision boundary is:

$$y = \begin{cases} 1, & \text{if } P(y = 1|X) \geq 0.5 \\ 0, & \text{otherwise} \end{cases}$$

Support Vector Machine (SVM)

SVM is a powerful supervised learning algorithm used for classification and regression tasks. It identifies the optimal hyperplane that separates data points into distinct classes with the maximum margin. SVM can handle both linear and non-linear data using the kernel trick, which transforms input features into higher dimensions for better separability.

Naïve Bayes

Naive Bayes is a probabilistic classifier based on Bayes' Theorem, assuming independence among predictors. Despite its simplicity, it is highly effective for tasks such as text classification and spam detection. It calculates the conditional probability of a class given the input features and selects the class with the highest probability.

Based on Bayes' Theorem:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

For multiple features $X = \{x_1, x_2, \ldots, x_n\}$, assuming independence:

$$P(X|y) = \prod_{i=1}^{n} P(x_i|y)$$

The class prediction is:

$$y = \arg\max_{y} \; P(y) \prod_{i=1}^{n} P(x_i|y)$$

Decision Tree

A Decision Tree is a flowchart-like algorithm that splits the dataset into subsets based on feature values, creating a tree structure. Each internal node represents a feature condition, each branch represents the outcome, and each leaf node corresponds to a class label or value. It is widely used for both classification and regression tasks due to its interpretability and simplicity.

Evaluation Metrics

1. Accuracy

Accuracy measures the proportion of correctly classified instances (both positive and negative) out of the total number of instances.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Where:

- **TP (True Positive)**: Correctly predicted positive cases
- **TN (True Negative)**: Correctly predicted negative cases
- **FP (False Positive)**: Incorrectly predicted positive cases
- **FN (False Negative)**: Incorrectly predicted negative cases

2. Precision

Precision measures the proportion of true positive predictions out of all positive predictions.

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

3. Recall (Sensitivity or True Positive Rate)

Recall measures the proportion of actual positive cases that are correctly identified.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

4. F1-Score

The F1-Score is the harmonic mean of precision and recall, providing a balance between the two.

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

5. Confusion Matrix

A confusion matrix is a tabular representation of model predictions versus actual values, providing a detailed view of prediction outcomes.

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive	False Negative
Actual Negative	False Positive	True Negative

6. Classification Report

A classification report is a comprehensive summary that includes metrics like precision, recall, F1-score, and support (number of samples per class).

7. ROC-AUC Curve (Receiver Operating Characteristic - Area Under Curve)

The ROC curve plots the True Positive Rate (Recall) against the False Positive Rate (FPR).

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

AUC (Area Under the Curve): Represents the likelihood that the model will rank a random positive instance higher than a random negative instance.

Code Snippets

Importing the dataset:

```
import pandas as pd
df=pd.read_csv('divorce.csv')
df
```

Importing the Libraries:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
```

Preparing the data:

```
# Load the dataset
df = pd.read_csv('divorce.csv')

# Separate features and target variable
X = df.iloc[:, :-1]  # Features
y = df['Divorce_Y_N']  # Target variable

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

KNN:

```
class KNN:
    def __init__(self, k=3):
        self.k = k

    def fit(self, X, y):
        self.X_train = X
        self.y_train = y

    def predict(self, X):
        predictions = []
        for x in X:
```

```
            predictions.append(self._predict(x))
        return predictions

    def _predict(self, x):
        """Predicts the class label for a single data point."""
        distances = [euclidean_distance(x, x_train) for x_train in
self.X_train]
        k_indices = np.argsort(distances)[:self.k]
        k_nearest_values = [self.y_train[i] for i in k_indices]
        return Counter(k_nearest_values).most_common(1)[0][0]

    def predict_proba(self, X):
        probabilities = []
        for x in X:
            distances = [euclidean_distance(x, x_train) for x_train in
self.X_train]
            k_indices = np.argsort(distances)[:self.k]
            k_nearest_labels = [self.y_train[i] for i in k_indices]
            class_counts = Counter(k_nearest_labels)
            class_probs = [class_counts[label] / self.k for label in
np.unique(self.y_train)]
            probabilities.append(class_probs)
        return np.array(probabilities)
```

LOGISTIC REGRESSION:

```
# Create a logistic regression model
model = LogisticRegression()

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = model.predict(X_test)
```

SVM:

```
# Initialize SVC with a linear kernel
svc_model = SVC(kernel='linear',probability=True)

# Train the model
svc_model.fit(X_train, y_train)

# Predict on the test set
```

```
y_pred = svc_model.predict(X_test)
```

NAÏVE BAYES:

```
# Create and train the Gaussian Naive Bayes model
model = GaussianNB()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)
```

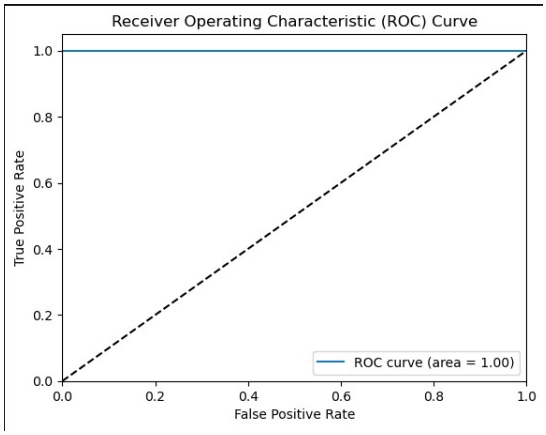
DECISION TREE:

```
# Create a Decision Tree Classifier
dtc = DecisionTreeClassifier()

# Train the model
dtc.fit(X_train, y_train)

# Make predictions on the test set
y_pred = dtc.predict(X_test)
```

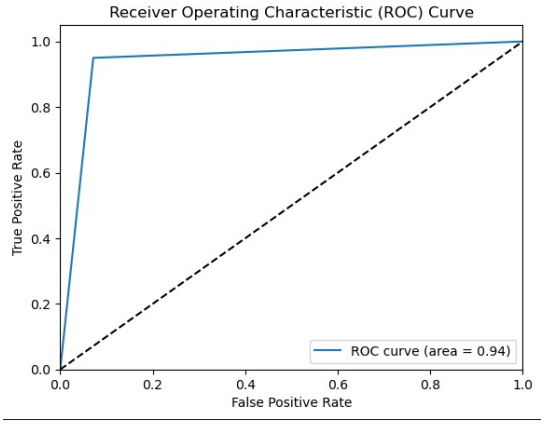
Results and Findings

MODEL	ACCURACY	PRECISION (class 0,1)	RECALL (class 0,1)	F1-SCORE (class 0,1)	CONFUSION MATRIX	ROC-AUC
KNN	0.97	0.93, 1.00	1.00, 0.95	0.97, 0.97	[[14,0], [1,19]]	1.00
SVM	1.00	1.00, 1.00	1.00, 1.00	1.00, 1.00	[[14, 0], [0, 20]]	1.00
NAÏVE BAYES	0.9412	0.93, 0.95	0.93, 0.95	0.93, 0.95	[[13, 1], [1, 19]]	0.96
LOGISTIC REGRESSION	1.00	1.00, 1.00	1.00, 1.00	1.00, 1.00	[[14, 0], [0, 20]]	1.00
DECISION TREE	0.9412	0.93, 0.95	0.93, 0.95	0.93, 0.95	[[13, 1], [1, 19]]	0.94

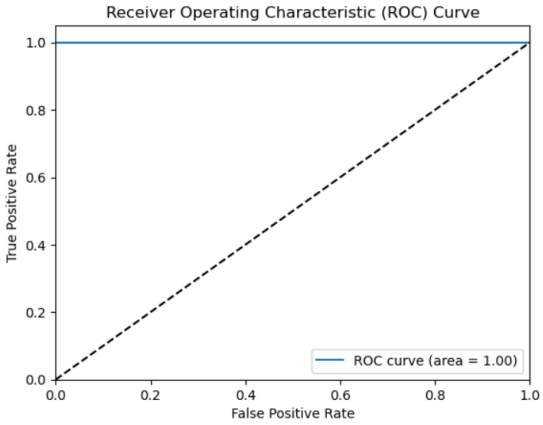
ROC-AUC Curves



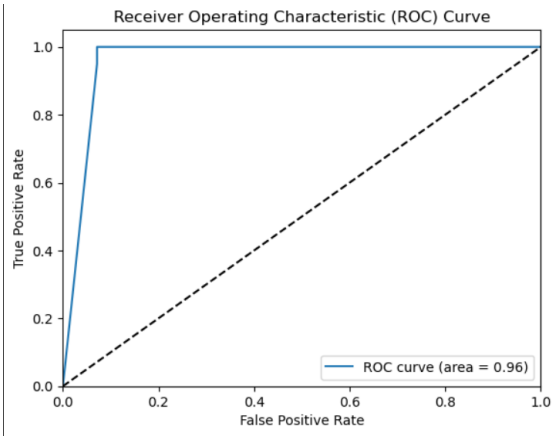
KNN



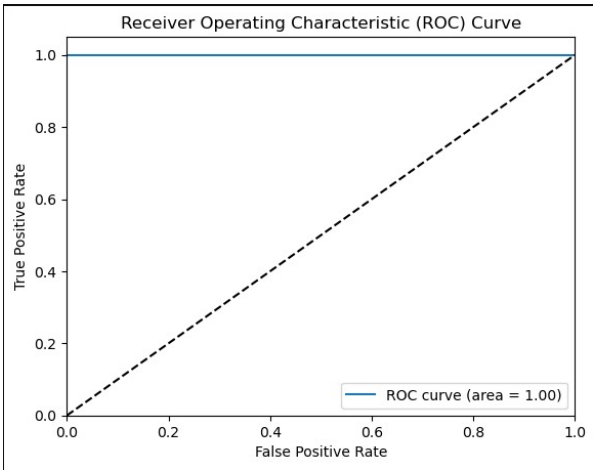
DECISION TREE



LOGISTIC REGRESSION

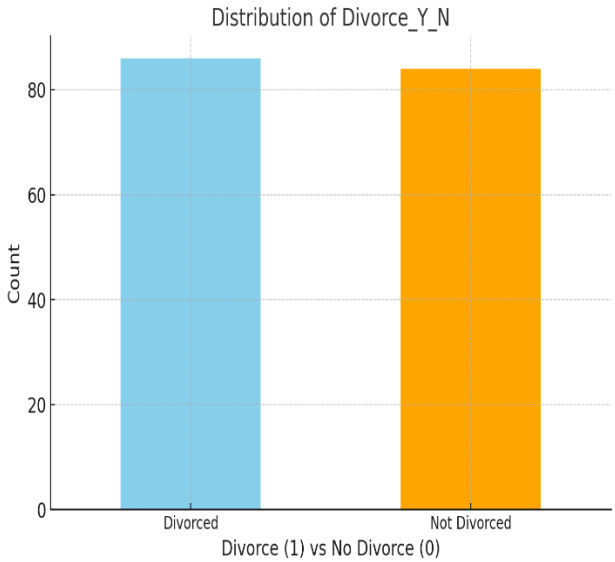


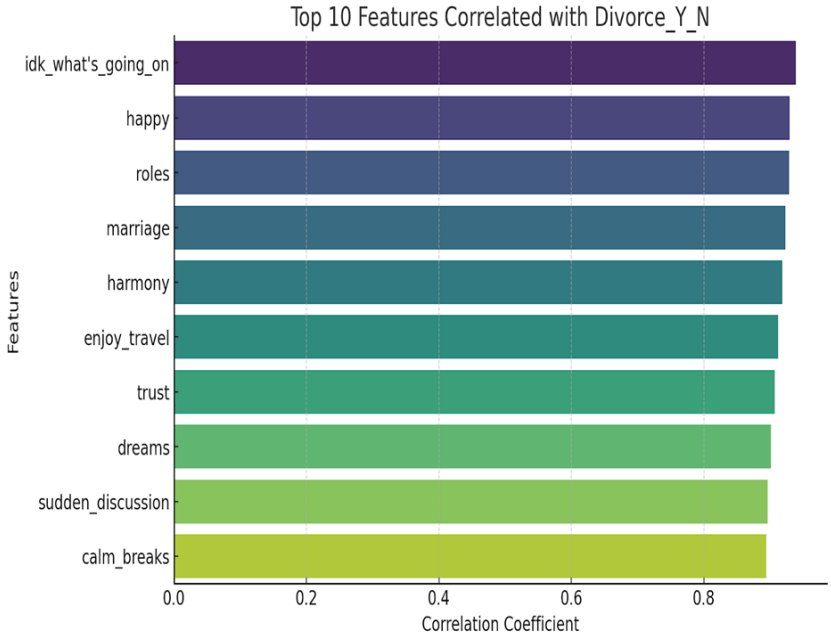
NAÏVE BAYES


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
ROC curve (area = 1.00)
0.0
0.2
0.4
0.6
0.8
1.0

SVM

Graphs


Distribution of Divorce_Y_N
Count
80
60
40
20
0
Divorced
Not Divorced
Divorce (1) vs No Divorce (0)


Top 10 Features Correlated with Divorce_Y_N
Features
idk_what's_going_on
happy
roles
marriage
harmony
enjoy_travel
trust
dreams
sudden_discussion
calm_breaks
0.0
0.2
0.4
0.6
0.8
Correlation Coefficient

Pearson Correlation Formula:

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \cdot \sum (y_i - \bar{y})^2}}$$

Where:

- x_i: Individual data points for feature x.
- y_i: Individual data points for target y (`Divorce_Y_N` in this case).
- $\bar{x}$: Mean of the feature x.
- $\bar{y}$: Mean of the target y.
- r: Pearson correlation coefficient.

The correlation coefficients between each feature and the target column Divorce_Y_N have been computed. Here are the top correlated features (excluding the target itself):

Feature	Correlation with Divorce_Y_N
idk_what's_going_on	0.94
happy	0.93
roles	0.93
marriage	0.92
harmony	0.92
enjoy_travel	0.91
trust	0.91
dreams	0.90
sudden_discussion	0.90
calm_breaks	0.89

Conclusion

This project effectively used SVM and Logistic Regression to predict divorce, achieving an impressive 100% accuracy. SVM excelled at identifying complex patterns, while Logistic Regression offered simplicity and highlighted key factors like communication and trust. Both models proved reliable and insightful, demonstrating the potential of machine learning in analyzing relationships. Future enhancements, such as adding more data and exploring additional features, could further improve prediction accuracy and provide deeper insights into marital dynamics.

References

DATASET SOURCED FROM
https://www.kaggle.com/datasets/csafrit2/predicting-divorce

Scikit-learn Documentation

• Scikit-learn, the Python machine learning library used for model implementation, provides comprehensive resources for algorithms and tools for data preprocessing, model evaluation, and more.

• https://scikit-learn.org/

Pandas Documentation

• For data manipulation and analysis, the Pandas library was used.

• https://pandas.pydata.org/

NumPy Documentation

• NumPy is a core library for numerical computing in Python, used for array manipulation and mathematical operations in this project.

• https://numpy.org/

Support Vector Machines - A Practical Guide

• This guide provides a detailed explanation of Support Vector Machines, the model used for regression in this project. •

https://www.svm-tutorial.com/

Matplotlib Documentation

• Matplotlib was used for any potential data visualization in the project. The oAicial site contains extensive guides for plotting and customizing graphs.

• https://matplotlib.org/