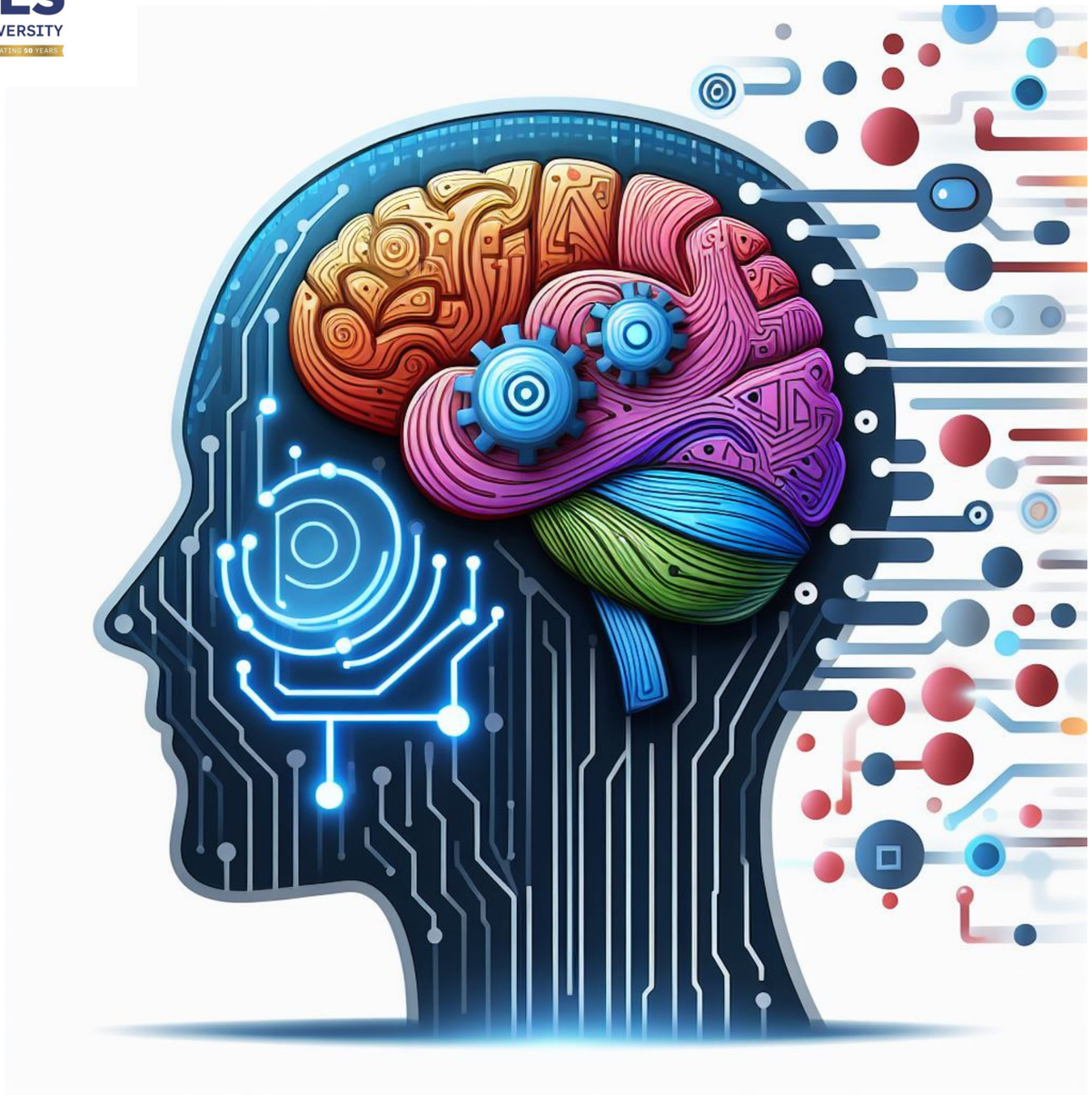
PES UNIVERSITY
CELEBRATING 50 YEARS



Mental Health Disorder Detection

TEAM COMPOSITION

TEAM NUMBER : 12

SRN	NAME
PES1UG21EC025	AKASH RAVI BHAT
PES1UG21EC910	TEJAS V P
PES1UG21EC052	ANUMULA BALAJI

CONTENTS

- Introduction
- Problem Statement
- Brief background
- Motivation
- Methadology
- Dataset
- Code snippet
- Output
- Literature Survey

INTRODUCTION

- The paper states that mental illness has become a serious problem in the world, especially in Asian countries, and that Data Science technologies can help in finding solutions.

- The research utilizes datasets from Kaggle and focuses on depression detection, aiming to differentiate between ML and DL models in this context.

PROBLEM STATEMENT

We will be increasing the efficiency of the Machine Learning Model by increasing accuracy and precision. Most of the models had issues with the overfitting of data. So we are going to use techniques which avoid such a problem.

BRIEF BACKGROUND

- The paper uses four machine learning models (Logistic Regression, K-Neighbors Classifier, Decision Tree Classifier and Bagging) to predict whether an individual has sought for any treatment for a mental health issue.
- Data Preprocessing: Extensive preprocessing steps such as cleaning, encoding, and splitting were performed to ready the data for the ML models

BRIEF MOTIVATION

- 10.7% of the global population: Over 792 million people suffer from mental illness, highlighting a critical worldwide issue demanding solutions.

- The confusion among developers and medical professionals regarding the distinctions between machine learning and deep learning models in the mental health domain is another compelling reason to delve into this problem

METHODOLOGY

- Data Preprocessing: The initial stage involves collecting, importing, and cleaning the data. The dataset is then encoded and split into training and test sets

- Machine Learning Module: This includes various algorithms like Logistic Regression, K-Neighbors Classifier, Decision Tree Classifier, and Bagging. The module predicts whether an individual is seeking treatment for mental health issues

- Training and Testing: The models are trained on the preprocessed data, and their performance is evaluated using accuracy, precision, and other metrics.

THE TIMELINE!

- we formed our team on 23 th january
- we read research papers from the given papers and choose the topic on 24 th jan
- we started working on the problem statement from 30 th jan
- we did code for additional 4 models with bease paper’s 4 models
- we found more precision and accuracy
- we solved overfitting problem
- we did our final report on april 1st

DATA SET

- surevey.csv is our dataset
- Data Source: The datasets for this model were obtained from Kaggle, a platform hosting a wide range of datasets.
- Data Content: The datasets include information like timestamps, activity scores, and mental health states of individuals.
- Data Preprocessing: Significant preprocessing steps such as cleaning, encoding, and splitting were performed to prepare the data for the ML and DL models.

DATA SET FEATURES

Demographic Information: Features like Timestamp, Age, Gender, Country, and state provide basic demographic details of the survey respondents.

Employment Details: Attributes such as self_employed, no_employees, remote_work, tech_company, and benefits offer insights into the respondents' employment status and workplace environment.

Mental Health Specifics: Fields like family_history, treatment, work_interfere, care_options, wellness_program, seek_help, anonymity, leave, mental_health_consequence, phys_health_consequence, coworkers, supervisor, mental_health_interview, phys_health_interview, mental_vs_physical, and obs_consequence delve into personal mental health history.

CODE SNIPPETS

Logistic Regression

```
[ ]  def logisticRegression():
         # train a logistic regression model on the training set
         logreg = LogisticRegression()
         logreg.fit(X_train, y_train)

         # make class predictions for the testing set
         y_pred_class = logreg.predict(X_test)

         accuracy_score = evalClassModel(logreg, y_test, y_pred_class, True)

         #Data for final graph
         methodDict['Log. Regression'] = accuracy_score * 100
```

```
logisticRegression()
```

KNeighbors Classifier

```
def Knn():
    # Calculating the best parameters
    knn = KNeighborsClassifier(n_neighbors=5)

    # define the parameter values that should be searched
    k_range = list(range(1, 31))
    weight_options = ['uniform', 'distance']

    # specify "parameter distributions" rather than a "parameter grid"
    param_dist = dict(n_neighbors=k_range, weights=weight_options)
    tuningRandomizedSearchCV(knn, param_dist)

    # train a KNeighborsClassifier model on the training set
    knn = KNeighborsClassifier(n_neighbors=27, weights='uniform')
    knn.fit(X_train, y_train)

    # make class predictions for the testing set
    y_pred_class = knn.predict(X_test)

    accuracy_score = evalClassModel(knn, y_test, y_pred_class, True)

    #Data for final graph
    methodDict['K-Neighbors'] = accuracy_score * 100
```

```
[ ] Knn()
```

Decision Tree classifier

```
def treeClassifier():
    # Calculating the best parameters
    tree = DecisionTreeClassifier()
    featuresSize = feature_cols.__len__()
    param_dist = {"max_depth": [3, None],
                  "max_features": randint(1, featuresSize),
                  "min_samples_split": randint(2, 9),
                  "min_samples_leaf": randint(1, 9),
                  "criterion": ["gini", "entropy"]}
    tuningRandomizedSearchCV(tree, param_dist)

    # train a decision tree model on the training set
    tree = DecisionTreeClassifier(max_depth=3, min_samples_split=8, max_features=6, criterion='entropy', min_samples_leaf=7)
    tree.fit(X_train, y_train)

    # make class predictions for the testing set
    y_pred_class = tree.predict(X_test)

    accuracy_score = evalClassModel(tree, y_test, y_pred_class, True)

    #Data for final graph
    methodDict['Decision Tree Classifier'] = accuracy_score * 100
```

```
[ ] treeClassifier()
```

Random Forests

```
def randomForest():
    # Calculating the best parameters
    forest = RandomForestClassifier(n_estimators = 20)

    featuresSize = feature_cols.__len__()
    param_dist = {"max_depth": [3, None],
              "max_features": randint(1, featuresSize),
              "min_samples_split": randint(2, 9),
              "min_samples_leaf": randint(1, 9),
              "criterion": ["gini", "entropy"]}
    tuningRandomizedSearchCV(forest, param_dist)

    # Building and fitting my_forest
    forest = RandomForestClassifier(max_depth = None, min_samples_leaf=8, min_samples_split=2, n_estimators = 350, random_state = 1)
    my_forest = forest.fit(X_train, y_train)

    # make class predictions for the testing set
    y_pred_class = my_forest.predict(X_test)

    accuracy_score = evalClassModel(my_forest, y_test, y_pred_class, True)

    #Data for final graph
    methodDict['Random Forest'] = accuracy_score * 100
```

```
[ ] randomForest()
```

Bagging

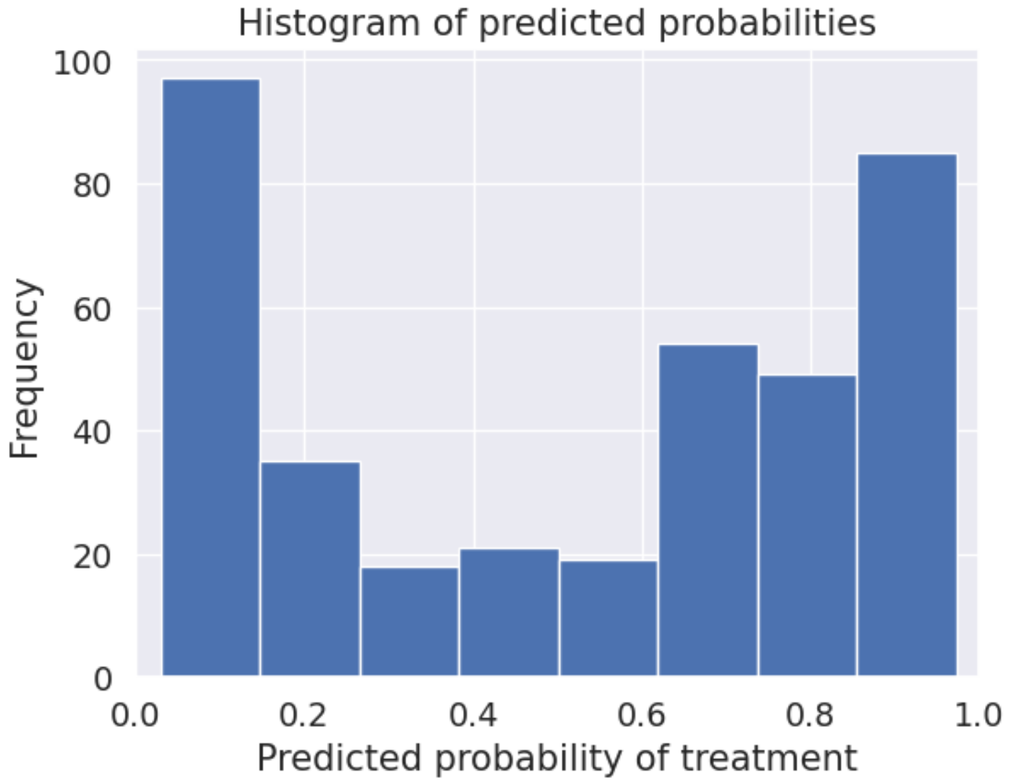
```
def bagging():
    # Building and fitting
    bag = BaggingClassifier(DecisionTreeClassifier(), max_samples=1.0, max_features=1.0, bootstrap_features=False)
    bag.fit(X_train, y_train)

    # make class predictions for the testing set
    y_pred_class = bag.predict(X_test)

    accuracy_score = evalClassModel(bag, y_test, y_pred_class, True)

    #Data for final graph
    methodDict['Bagging'] = accuracy_score * 100
```

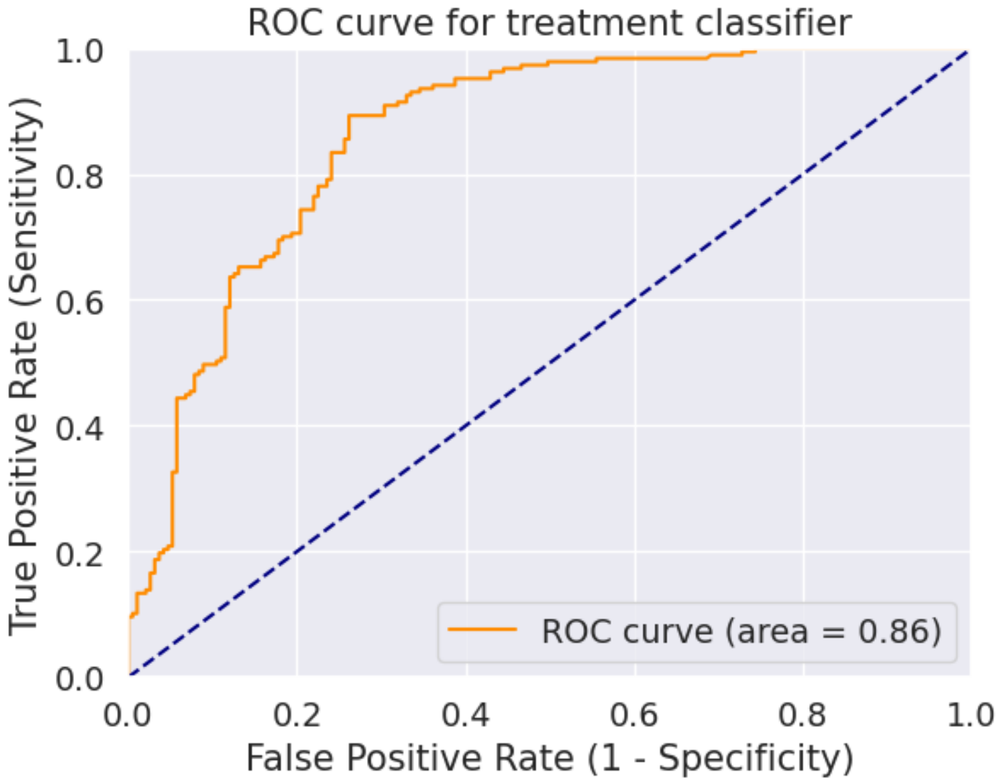
```
[ ] bagging()
```

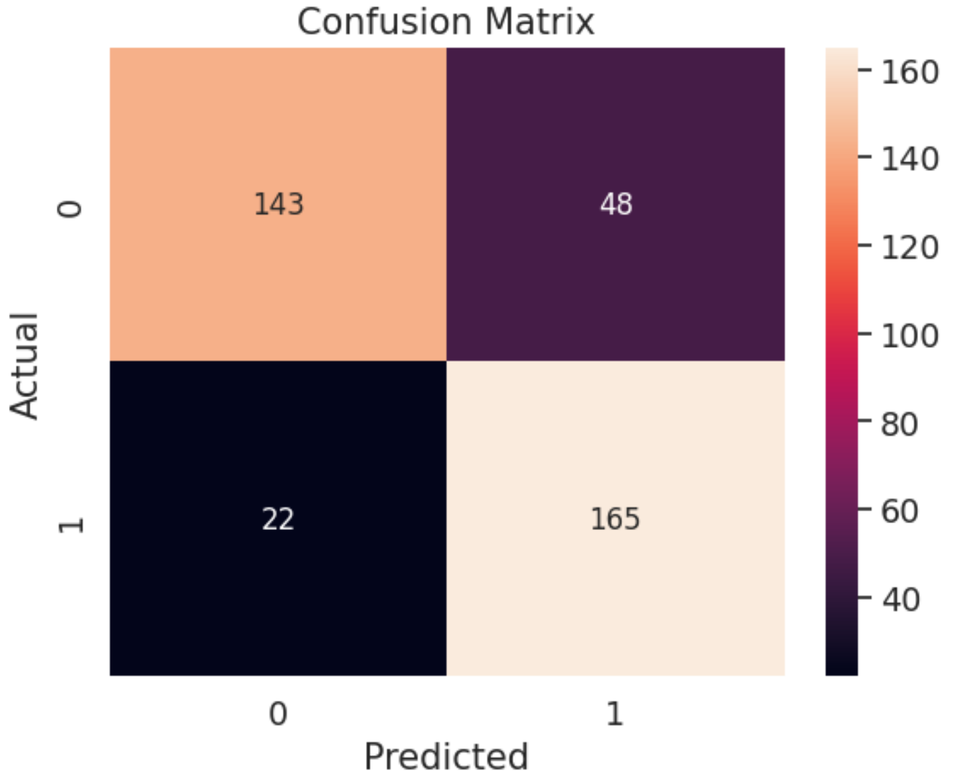
Boosting

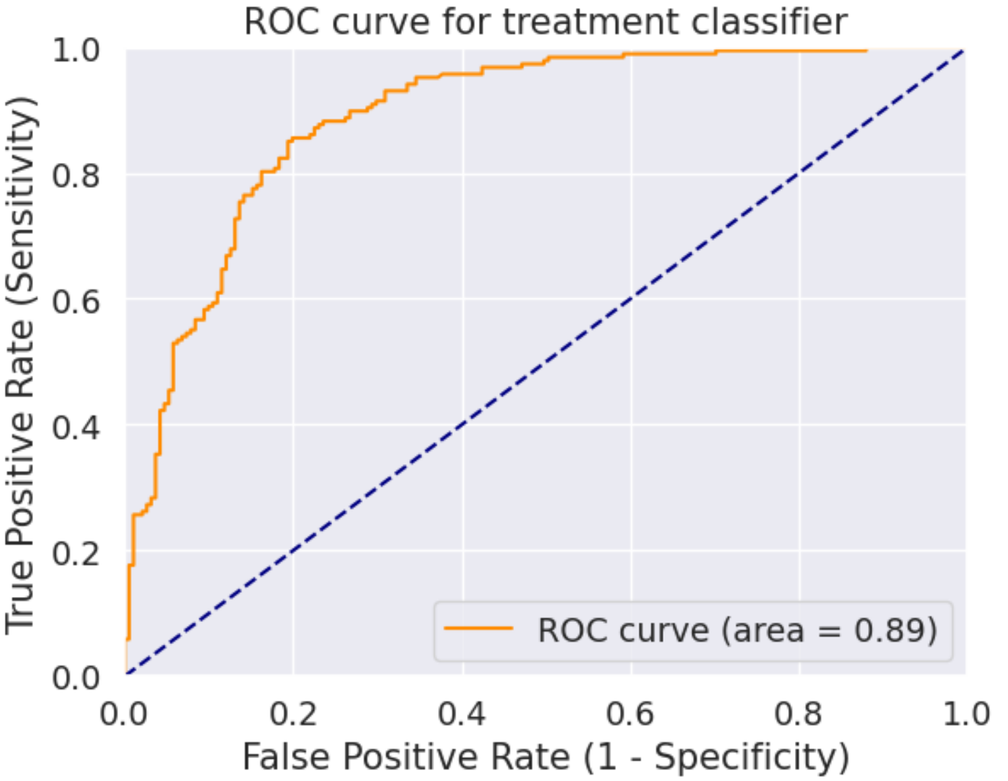
```
def boosting():
    # Building and fitting
    clf = DecisionTreeClassifier(criterion='entropy', max_depth=1)
    boost = AdaBoostClassifier(base_estimator=clf, n_estimators=500)
    boost.fit(X_train, y_train)

    # make class predictions for the testing set
    y_pred_class = boost.predict(X_test)

    accuracy_score = evalClassModel(boost, y_test, y_pred_class, True)

    #Data for final graph
    methodDict['Boosting'] = accuracy_score * 100
```

```
[ ] boosting()
```

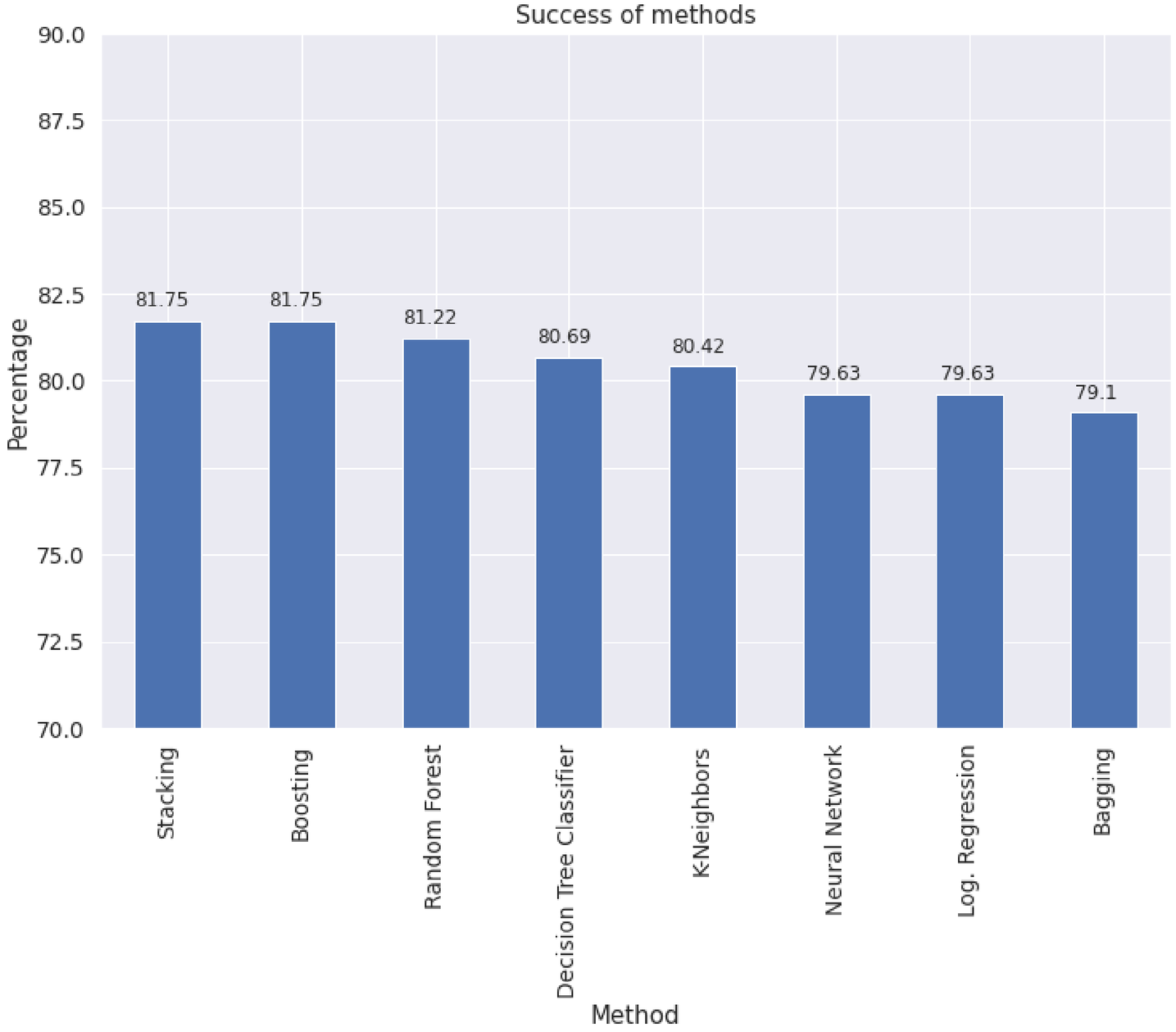

Histogram of predicted probabilities
Frequency
Predicted probability of treatment
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0


ROC curve for treatment classifier
True Positive Rate (Sensitivity)
False Positive Rate (1 - Specificity)
ROC curve (area = 0.86)
0.0
0.2
0.4
0.6
0.8
1.0


Confusion Matrix
143
48
22
165
Actual
0
1
Predicted
0
1
160
140
120
100
80
60
40


ROC curve for treatment classifier
True Positive Rate (Sensitivity)
False Positive Rate (1 - Specificity)
ROC curve (area = 0.89)
0.0
0.2
0.4
0.6
0.8
1.0

MODEL	BASE PAPER	OUR OUTPUT
Logistic Regression	75.8	79.63
Knn Classifier	76.9	80.42
Decision Tree Classifier	72.8	80.69
Random Forest	78	81.22
Stacking	-	81.75
Bagging	-	79.1
Boosting	-	81.75
Neural Network	-	79.63


Success of methods
90.0
87.5
85.0
82.5
80.0
77.5
75.0
72.5
70.0
Percentage
81.75
81.75
81.22
80.69
80.42
79.63
79.63
79.1
Stacking
Boosting
Random Forest
Decision Tree Classifier
K-Neighbors
Neural Network
Log. Regression
Bagging
Method

NOVELTY

- Overfitting Reduction: Implemented techniques to decrease overfitting in Logistic Regression, enhancing model generalizability.
- Accuracy Improvement: Achieved higher accuracy across all models, indicating more reliable predictions.
- Model Expansion: Added four new models — Random Forest, Stacking, Boosting, and Neural Networks — expanding the predictive capabilities beyond the base paper's models.
- Diverse Techniques: The inclusion of ensemble methods like Random Forest, Stacking, and Boosting introduces a variety of decision-making processes for better performance.

LITERATURE SURVEY

TITLE	Mental Health Disorder Detection using Machine Learning and Deep Learning Tehniques
AUTHORS' WORK	The paper cites various studies that have used Machine Learning and Deep Learning models to detect mental health disorders, such as depression, stress, schizophrenia, etc.
INFERENCES	the Logistic Regression model is the best for predicting whether an individual needs treatment for a mental health issue, and the CNN model is the best for predicting whether an individual is depressed or not. The paper also analyzes the factors that affect the performance and generalization of the models, such as data quality, quantity, balance, normalization, etc.
LIMITATIONS	overfitting and underfitting problems

LITERATURE SURVEY

TITLE	Study on Mental Disorder Detection via Social Media Mining
AUTHORS' WORK	The author's work involves investigating and presenting methods for detecting mental disorders through the analysis of social media content. The authors, Iwan Syarif, Nadia Ningtias, and Tessy Badriyah, likely discuss techniques, findings, and implications related to leveraging social media data for mental disorder detection.
INFERENCES	The study explores the detection of mental disorders through social media mining, presenting findings and insights at the 2019 4th International Conference on Computing, Communications, and Security (ICCCS).
LIMITATIONS	The limitations of the study may include potential challenges in accurately identifying mental disorders through social media, ethical concerns related to user privacy, and the dynamic nature of online content that could impact the reliability of detection methods.

LITERATURE SURVEY

TITLE	Predicting Mental Health Disorders Using Machine Learning for Employees in Technical and Non-Technical Companies
AUTHORS' WORK	The author's work involves investigating and presenting methods for detecting mental disorders through the analysis of social media content. The authors, Iwan Syarif, Nadia Ningtias, and Tessy Badriyah, likely discuss techniques, findings, and implications related to leveraging social media data for mental disorder detection.
INFERENCES	aims to provide insights into the effectiveness of machine learning in identifying potential mental health issues among employees, particularly in diverse work environments.
LIMITATIONS	challenges in obtaining accurate and representative datasets, addressing biases in the machine learning models, and acknowledging the complex and multifaceted nature of mental health

LITERATURE SURVEY

TITLE	Predicting Mental Health Disorders Using Machine Learning for Employees in Technical and Non-Technical Companies
AUTHORS' WORK	The author's work involves investigating and presenting methods for detecting mental disorders through the analysis of social media content. The authors, Iwan Syarif, Nadia Ningtias, and Tessy Badriyah, likely discuss techniques, findings, and implications related to leveraging social media data for mental disorder detection.
INFERENCES	aims to provide insights into the effectiveness of machine learning in identifying potential mental health issues among employees, particularly in diverse work environments.
LIMITATIONS	challenges in obtaining accurate and representative datasets, addressing biases in the machine learning models, and acknowledging the complex and multifaceted nature of mental health

CONCLUSION

Effective Detection: The project successfully implements various machine learning models to predict mental health treatment needs, with Logistic Regression being highlighted as particularly effective.

Overfitting Mitigation: Techniques have been applied to reduce overfitting, improving the generalizability of the models.

Model Enhancements: The addition of new models like Random Forest, Stacking, Boosting, and Neural Networks has expanded the predictive capabilities and accuracy.

Significant Impact: The research addresses a critical global issue, demonstrating the potential of Data Science technologies in aiding mental health disorder detection.


THANK
YOU