Final Project Report: Medical Treat Dataset

Kiet Nguyen

John Mintsa

Nathan Magno

Kierra Manuel

California State University, Long Beach

CECS 456: Machine Learning

Mostafa Majidpour

December 10, 2023

Introduction:

Mental health includes our emotional, psychological, and social well-being. It affects how we think, feel, and act. It also helps determine how we handle stress, relate to others, and make choices. Mental health is important at every stage of life, from childhood and adolescence through adulthood. Mental Health has become an increasingly important issue for people in the workplace as burnout and worker fatigue are rising and more people than ever are quitting their jobs citing toxic workplace culture and to maintain their mental health.

The dataset that our group is using is called the Medical Treatment Dataset from Kaggle. It was created by Shadab Hussain and the data set contains various attributes relating to mental health and aims to predict treatment for each patient in the data set with mental health issues. The datasets will include variables such as S.no which is the ID number for each patient, Timestamp to track the time, Age of the patient, Gender of the patient, Country they are from, State they reside in, whether they are self-employed, if they have a family history of mental health issues, and their number of employees. From this data set, the model(s) is/are created to predict the variable Treatment from the test set to see whether treatment is needed with a “yes” or a “no”. We will use exploratory data analysis and several machine learning models to answer the following questions:

Methodology:

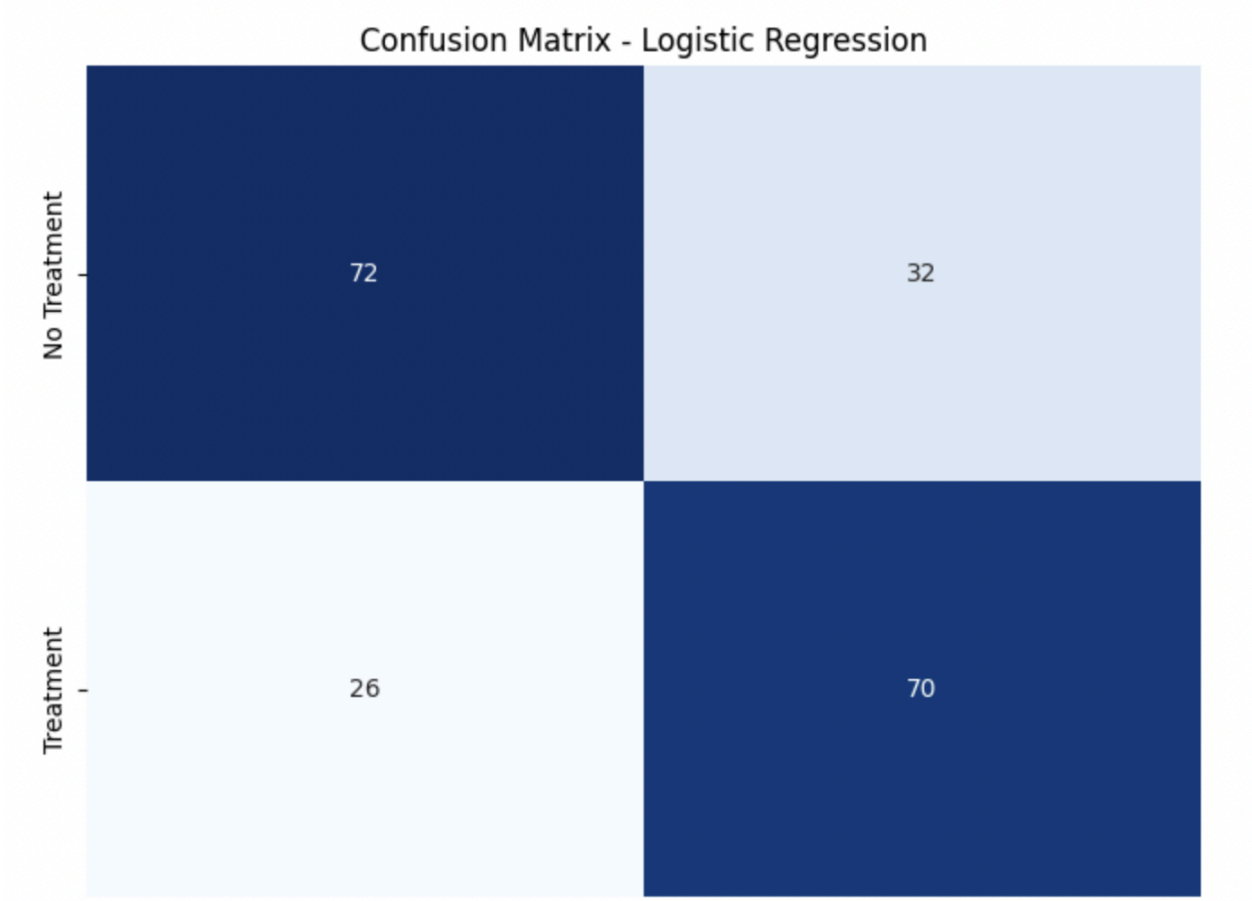
In our exploration of the Medical Treatment Dataset, we conducted a thorough analysis aiming to predict the necessity of mental health treatment for individuals. The dataset encompasses diverse attributes, ranging from demographic information to workplace-related factors. Mental health is a crucial aspect of overall well-being, especially in the context of a

workplace where issues like burnout and toxic cultures have become prevalent.

Leveraging machine learning models, including Decision Tree, Random Forest, Boosting Tree, and Logistic Regression, we sought to understand and predict the "treatment" variable based on various features. Our analysis involved preprocessing steps such as handling missing values and encoding categorical variables, as well as feature engineering to enhance model accuracy.

We evaluated the models using metrics like accuracy, precision, recall, and F1-score, considering the implications of false positives and false negatives. Additionally, we delved into model interpretability to elucidate the factors influencing predictions. Through this comprehensive analysis, we aim to contribute valuable insights into the intersection of mental health and predictive modeling, shedding light on potential interventions for creating healthier workplace environments.

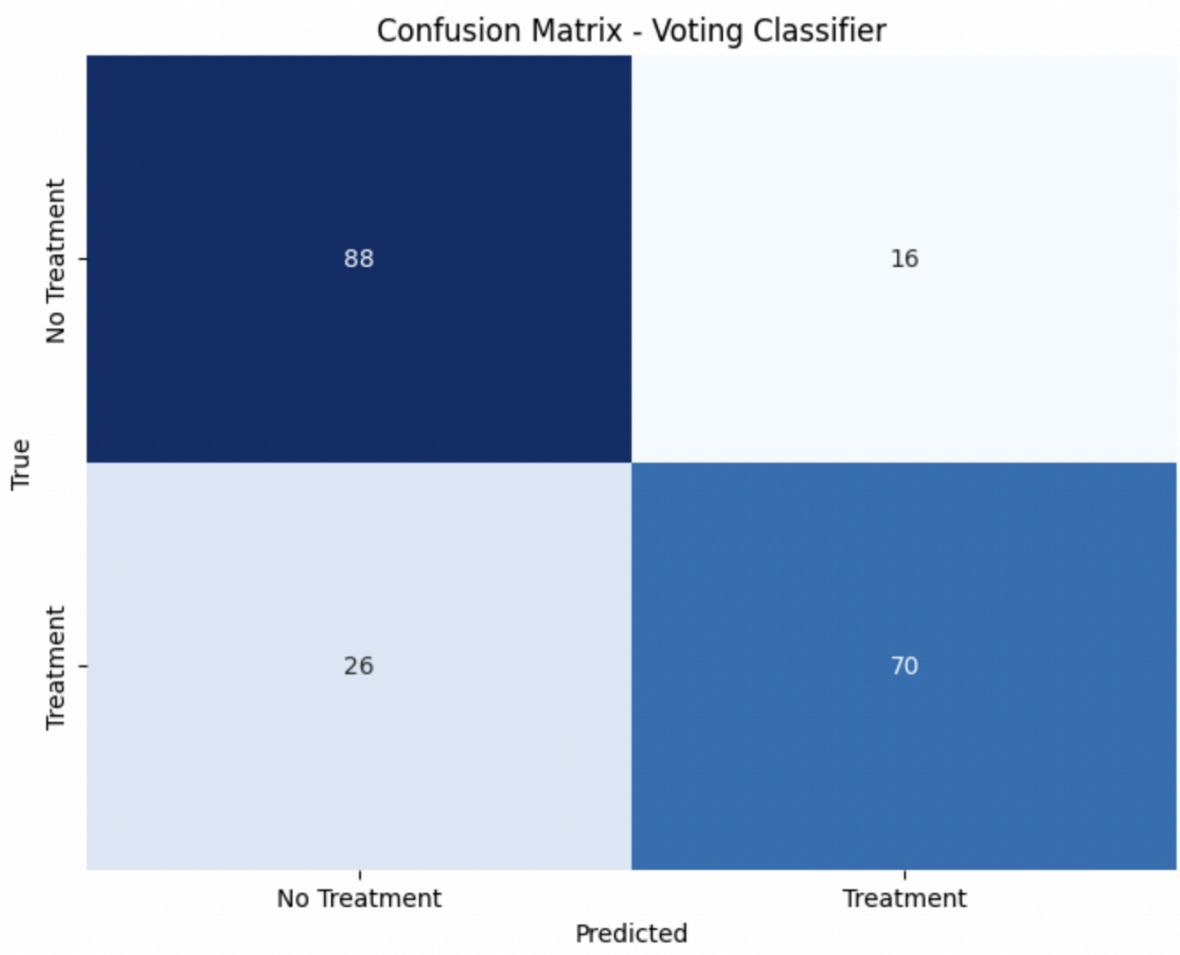
Confusion Matrix Explanation:

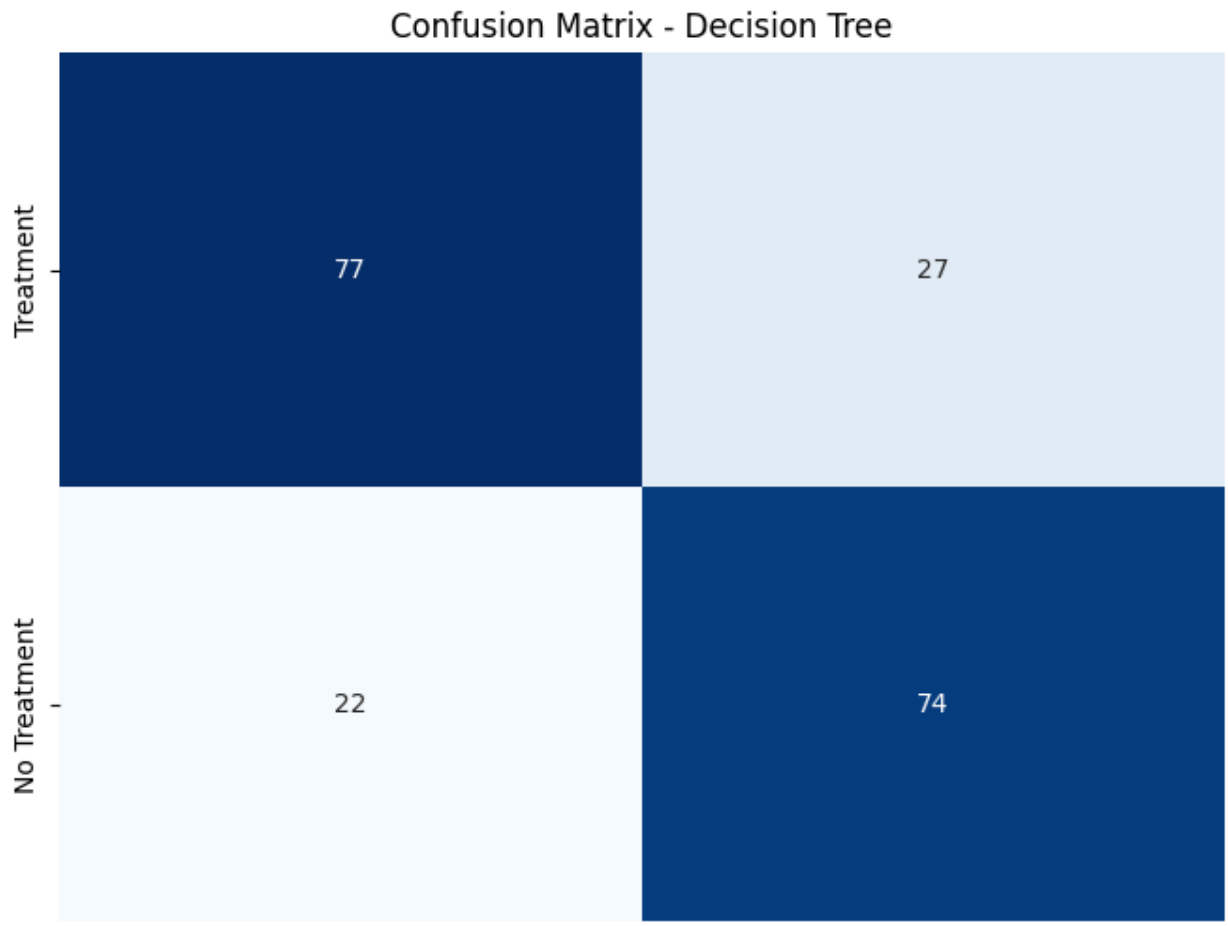


The Logistic Regression model achieved an overall accuracy of 71%. The confusion matrix breakdown reveals that 72 instances were correctly identified as requiring mental health treatment, while 32 were falsely predicted as positive. Additionally, there were 26 false negatives and 70 true positives. In an effort to enhance accuracy, we introduced an ensemble method, specifically the Voting Classifier. By combining the predictive capabilities of Logistic Regression and a Random Forest model, the ensemble method significantly improved the overall accuracy to an impressive 79%. The resulting confusion matrix showcased a notable advancement, with 88 instances correctly identified as requiring treatment, and only 16 false

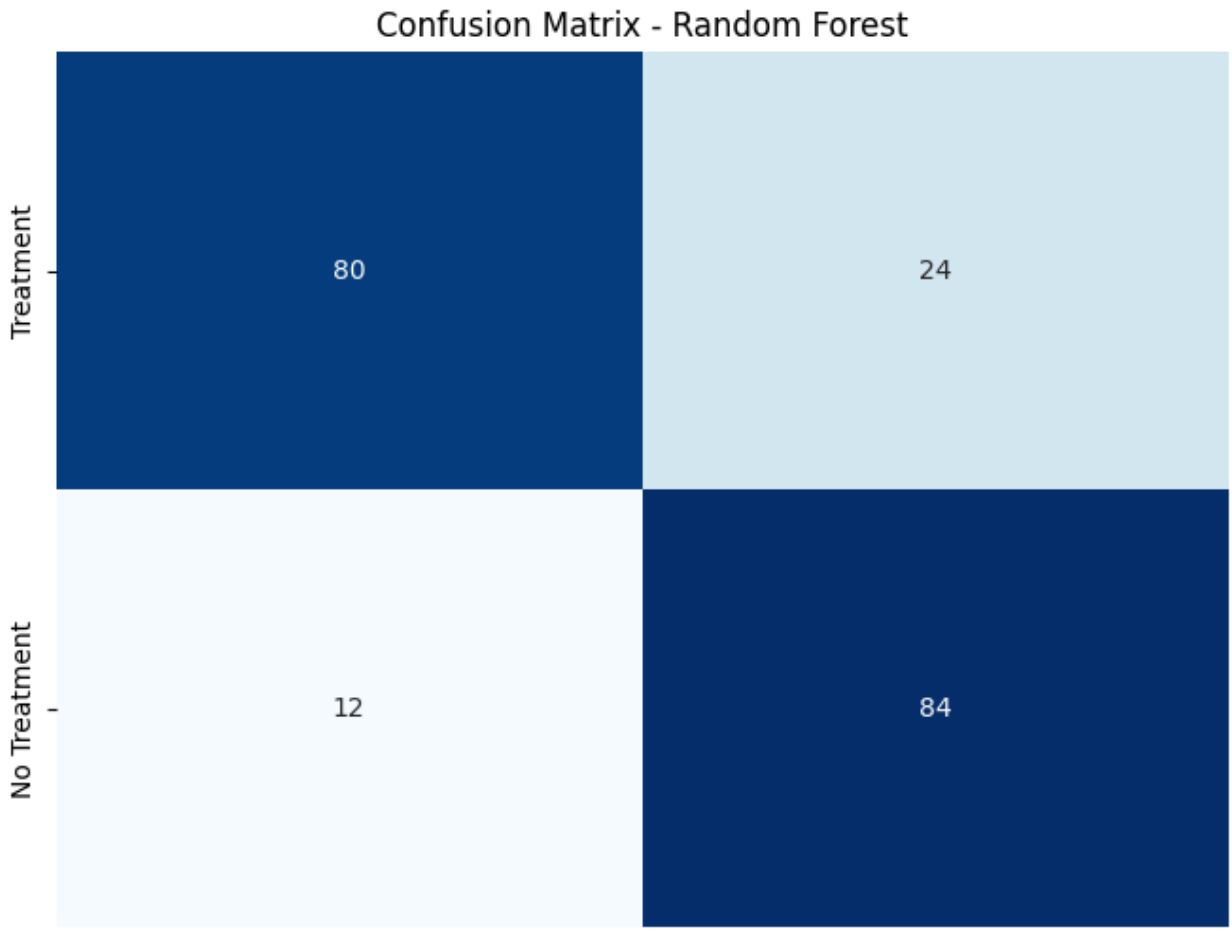
positives. The ensemble method effectively reduced false negatives to 26 while maintaining a robust 70 true positives.

This augmentation in accuracy and the refined confusion matrix underscore the potency of ensemble methods in harnessing the strengths of individual models, providing a more comprehensive and accurate prediction mechanism for mental health treatment needs. The ensemble approach demonstrates its effectiveness in mitigating the limitations of standalone models and presents a valuable avenue for improving predictive performance in complex medical datasets.

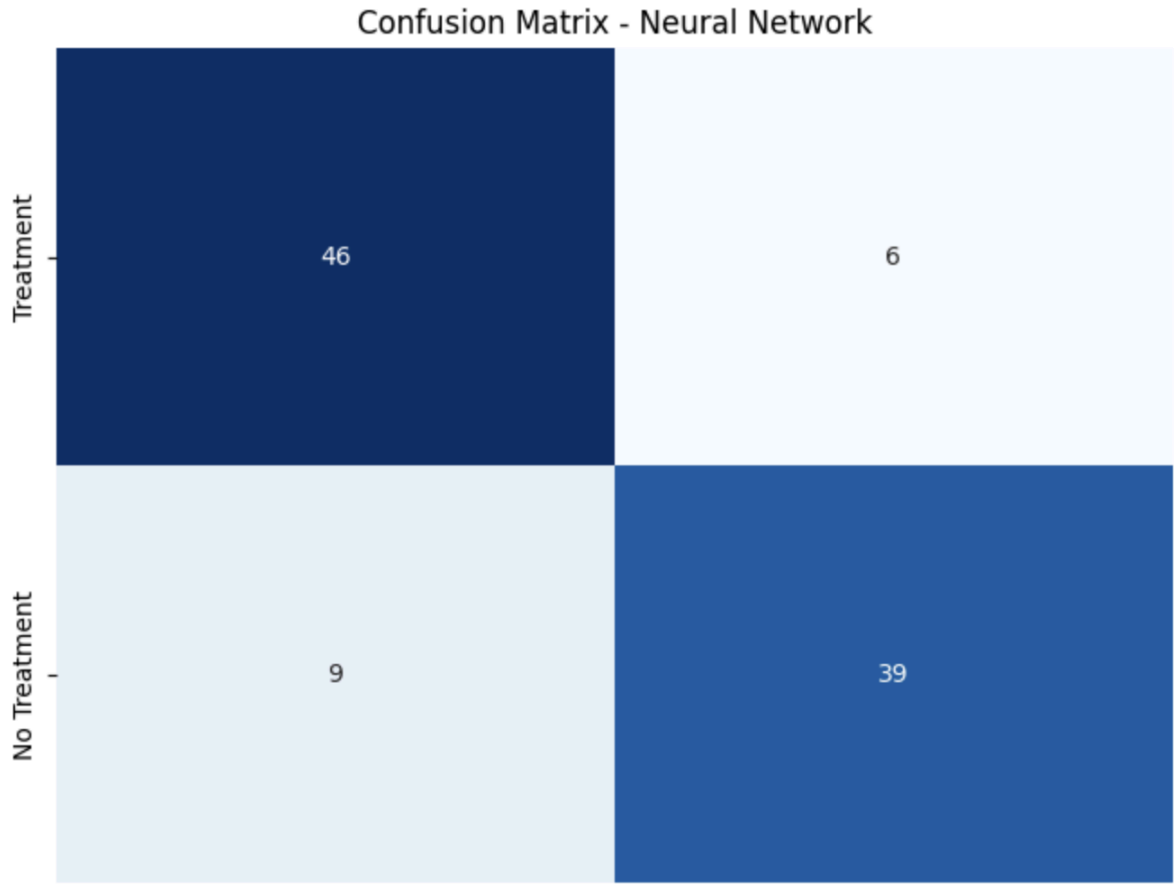




The decision tree model achieved an overall accuracy of 75.5%. The confusion matrix breakdown reveals that 77 instances were correctly identified as requiring mental health treatment, while 27 cases were falsely predicted as positive. Additionally, 74 instances were correctly classified as not needing treatment, with 22 instances being false negatives.

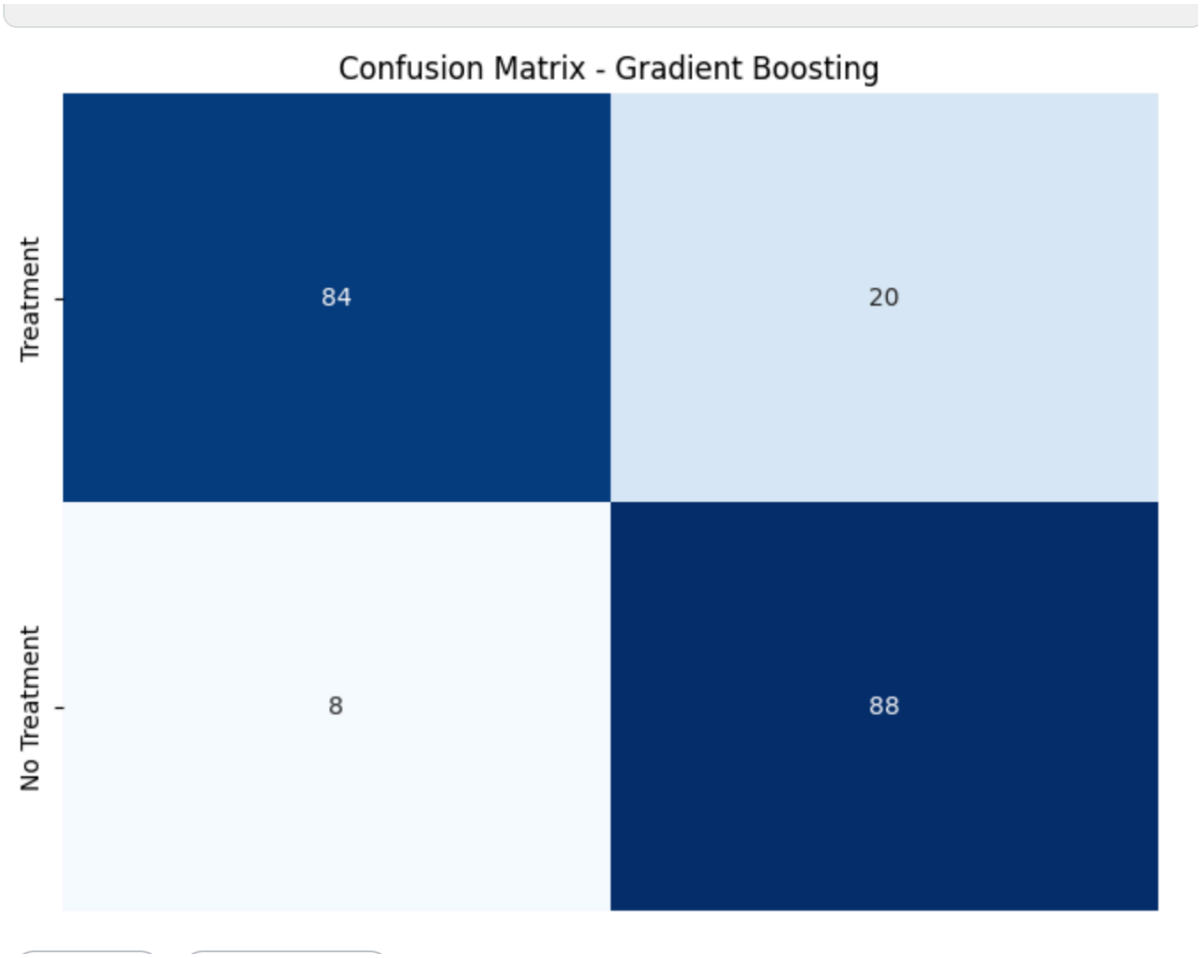


The random forest model achieved an overall accuracy of 82%. The confusion matrix breakdown reveals that 80 instances were correctly identified as requiring mental health treatment, while 24 cases were falsely predicted as positive. Additionally, 84 instances were correctly classified as not needing treatment, with 12 instances being false negatives.




```
Accuracy: 0.85
```

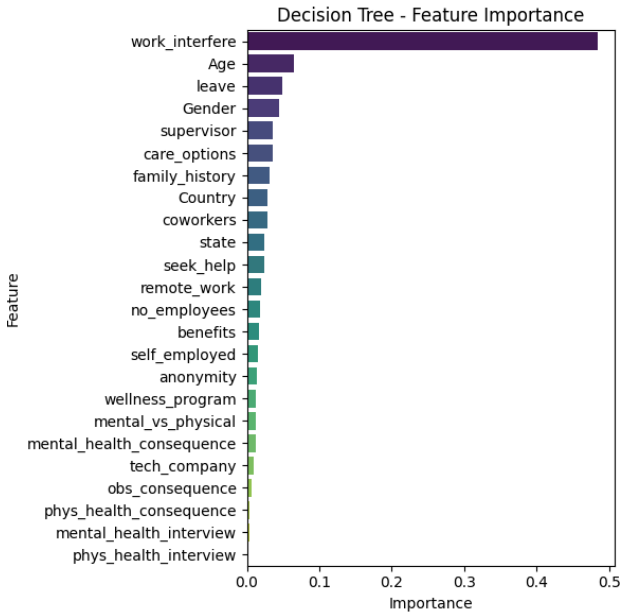
Our Neural Network implementation was able to make predictions on our Medical Treatment Dataset with 85% accuracy. Out of 100 total samples, the model was able to identify 46 true positives, 6 false positives, 39 true negatives, and 9 false negatives. The test size was decreased in order to give the neural network more data to train on, since we were working with a limited dataset.

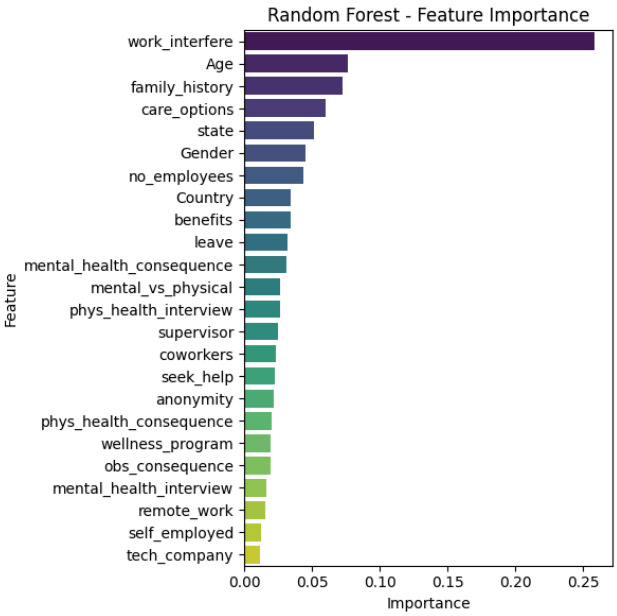


The Gradient Boosting model achieved an overall accuracy of 82%. The confusion matrix breakdown reveals that 84 instances were correctly identified as requiring mental health treatment, while 20 were falsely predicted as positive. Additionally, there were 8 false negatives and 88 true positives.

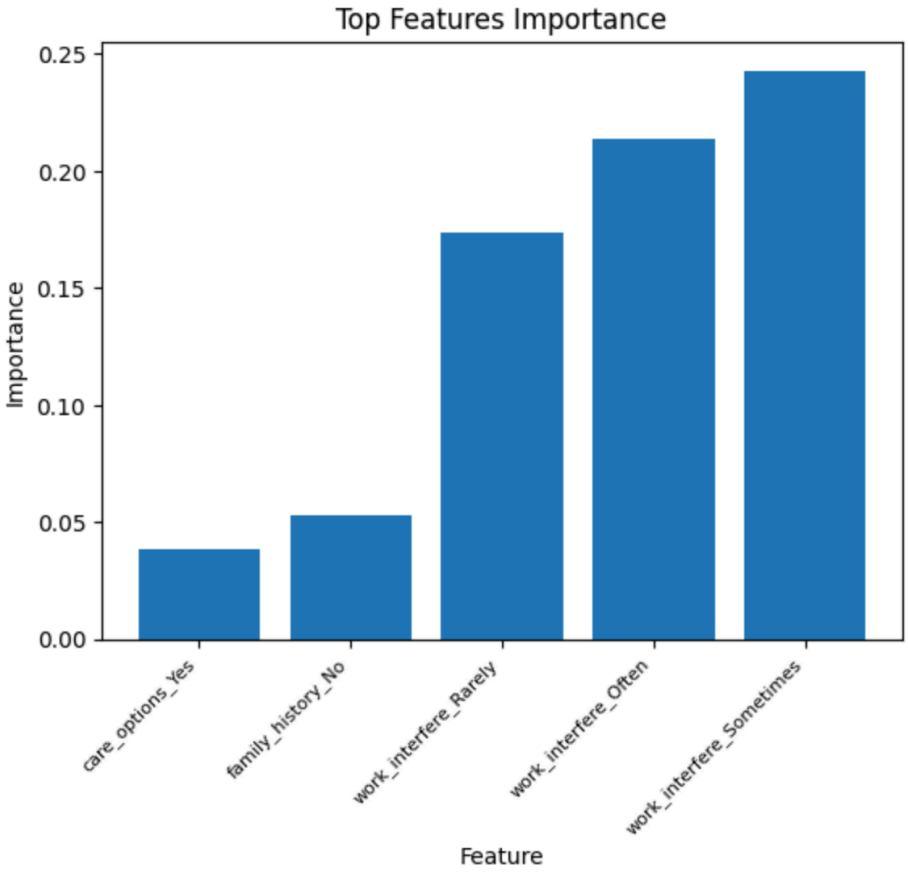
Feature Importance Explanation:

- **Decision Tree & Random Forest**





- **Gradient Boosting**



The chart indicates that 'work_interfere,' expressed through values such as 'often,' 'sometimes,' 'always,' or 'never,' is a crucial factor in predicting the necessity of mental health treatment. This significance is particularly evident in both the decision tree and random forest models above. In the case of the decision tree, 'work_interfere' appears overwhelmingly influential, contributing to more than ~50% of the predictive power. The random forest model also underscores the importance of 'work_interfere' with a slightly reduced impact of ~25%. Additionally, age and family history are identified as relevant features in the random forest algorithm. On the other hand, The gradient Boosting model emphasize the importance of ‘work_interfere’ and brings to our attention two important features that are ‘family_history’ and ‘care_option’.

Error Analysis:

- **Decision Tree & Random Forest**

```
Common Attributes for Random Forest Errors:
Country                       33.69
Age                           33.14
state                         28.39
Gender                        17.97
no_employees                   3.36
work_interfere                 1.53
benefits                       1.08
mental_health_consequence      1.08
leave                          1.03
wellness_program               1.00
care_options                   0.94
seek_help                      0.92
mental_health_interview        0.92
phys_health_consequence        0.89
tech_company                   0.86
coworkers                      0.83
phys_health_interview          0.81
supervisor                     0.72
mental_vs_physical             0.72
anonymity                      0.64
remote_work                    0.31
family_history                 0.17
self_employed                  0.11
obs_consequence                0.11
```

```
Common Attributes for Decision Tree Errors:
Age                          32.27
state                        31.29
Country                      29.14
Gender                       19.73
no_employees                  2.96
work_interfere                1.82
leave                         1.31
benefits                      1.00
wellness_program              1.00
mental_health_consequence     0.98
mental_health_interview       0.94
mental_vs_physical            0.92
care_options                  0.90
tech_company                  0.88
seek_help                     0.86
supervisor                    0.86
coworkers                     0.78
phys_health_interview         0.78
phys_health_consequence       0.76
anonymity                     0.49
family_history                0.31
remote_work                   0.24
self_employed                 0.20
obs_consequence               0.20
```

In the context of Decision Tree errors, it appears that the model is more sensitive to factors such as age, geographical location, and the number of employees in a company. On the other hand, Random Forest errors show a similar sensitivity to age and country but also emphasize the influence of benefits, mental health consequences, and leave policies, suggesting that these factors contribute more significantly to misclassifications in the Random Forest model.

- **Gradient Boosting Errors:**

```
Common Attributes for Gradient boosting Errors:
Age                                30.79
obs_consequence_No                  0.89
tech_company_Yes                    0.89
family_history_No                   0.86
self_employed_No                    0.82
mental_health_interview_No          0.82
phys_health_consequence_No          0.75
coworkers_Some of them              0.68
remote_work_No                      0.68
wellness_program_No                 0.68
dtype: float64
```

In the context of Gradient Boosting errors, it appears that the model is more sensitive to only one factor which is age. All the other factors seem to hold the same importance.

Accuracy Improvement:

- **Decision Tree & Random Forest**

```
# Example: Feature Selection (Top 9 features)
top_features = dt_feature_importance_df['Feature'].head(9).tolist()
X_train_selected = X_train[top_features]
X_val_selected = X_val[top_features]

# Decision Tree after feature engineering
dt_model.fit(X_train_selected, y_train)
dt_pred_selected = dt_model.predict(X_val_selected)
dt_accuracy_selected = accuracy_score(y_val, dt_pred_selected)
print(f'Decision Tree Accuracy (After): {dt_accuracy_selected}')

# Random Forest after feature engineering
rf_model.fit(X_train_selected, y_train)
rf_pred_selected = rf_model.predict(X_val_selected)
rf_accuracy_selected = accuracy_score(y_val, rf_pred_selected)
print(f'Random Forest Accuracy (After): {rf_accuracy_selected}')
```

```
Decision Tree Accuracy (After): 0.77
Random Forest Accuracy (After): 0.835
```

After selecting the top 9 most important features based on the decision tree's feature importance scores, both the Decision Tree and Random Forest models were retrained, resulting in improved accuracies. The Decision Tree accuracy increased from 0.755 to the new accuracy of 0.77, and the Random Forest accuracy increased from 0.82 to the new accuracy of 0.835

- **Neural Network**

The initial implementation of the Neural Network model only consisted of 4 hidden layers with a highest neuronal count of 64. Test size was also initialized to 0.2 in the initial

development of the model. Epochs of the initial model was set to a value of 10. Initial accuracy of the neural network model is represented in the following accuracy percentage and confusion matrix:

In order to give the model more data to be trained with, test size was decreased from 0.2 to 0.1 in the final implementation of the neural network model, which means a split of 900 = train and 100 = test.

The amount of hidden layers was also doubled, with 8 hidden layers and a highest neuronal count of 1028. Each extra hidden layer was added one by one to the top of the hidden layer chain. Each time another layer was added, the amount of neurons in the layer was also doubled (ex. First extra layer 128, second extra layer 256, etc.). Further hidden layers were not added in order to prevent overfitting of train data, and improvements to accuracy also tapered off at this amount of hidden layers.

The amount of Epochs was also increased from 10 to 150 in order to compensate for the lack of more training data. During testing, Epochs was initialized to the highest value of 200, but upon investigating accuracy at each epoch, gain in accuracy seemed to stop at around 150 epochs. It was decreased from 200 to 150 in order to optimize for time.

Together, these changes were able to improve the accuracy of the neural network model by nearly 20 percent (final implementation accuracy = 85%). The final implementation will be shown in the following Code Snippets section.

- **Gradient Boosting**

To enhance the model's accuracy, we made adjustments to key hyperparameters, specifically increasing the learning rate from 0.01 to 0.1 and reducing the min_samples_leaf

parameter from 5 to 1. As a result, the accuracy improved significantly, rising from 0.82 to 0.86.

- **Logistic Regression**

Code Snippets:

- **Neural Network**

```
model = keras.Sequential([
    keras.layers.Input(shape=(24,)),

    keras.layers.Dense(1028, activation='relu'),
    keras.layers.Dense(512, activation='relu'),
    keras.layers.Dense(256, activation='relu'),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(16, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

model.fit(X_train, y_train, epochs=150, validation_data=(X_val, y_val))

# Make predictions
predictions = model.predict(X_val)
#print(predictions)
#print(len(predictions))
predictions_binary = (predictions > 0.5).astype(int)
accuracy = accuracy_score(y_val, predictions_binary)
print(f"Accuracy: {accuracy}")

confusion = confusion_matrix(y_val, predictions_binary)
test_confusion_df = pd.DataFrame(confusion, index=range(2), columns=range(2))
print(test_confusion_df)
```

- **Gradient Boosting**

Initializing the Data

```
test_data = pd.read_csv("/kaggle/input/medical-treatment-dataset/testms.csv")
train_data = pd.read_csv("/kaggle/input/medical-treatment-dataset/trainms.csv")
```

+ Code + Markdown

```
# Drop unnecessary columns (if needed)
train_data = train_data.drop(columns=['s.no', 'Timestamp', 'comments'])
test_data = test_data.drop(columns=['s.no', 'Timestamp', 'comments'])

# Split the data into features and target variable
X = train_data.drop('treatment', axis=1)
X = pd.get_dummies(X)
y = train_data['treatment']

# Split the data into training and testing sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

Train, fit and predict Data

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Create and train the GradientBoostingClassifier
model = GradientBoostingClassifier(n_estimators=100, random_state=42, learning_rate=0.001,subsample=1.0,min_samples_leaf=5)
model.fit(X_train, y_train)

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

Get Accuracy

```
# Evaluate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

Gradient Boosting Accuracy: 0.86

- **Decision Tree & Random Fores**

```
# Encode categorical features
le = LabelEncoder()
categorical_columns = ['Gender', 'Country', 'state', 'self_employed', 'family_history',
                       'work_interfere', 'no_employees', 'remote_work', 'tech_company',
                       'benefits', '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', 'obs_consequence']
for column in categorical_columns:
    if column in train_data.columns:
        train_data[column] = le.fit_transform(train_data[column])

# Split the data into features and target variable
X = train_data.drop('treatment', axis=1)
y = train_data['treatment']

# Split the data into training and testing sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# Decision Tree Classifier
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
dt_pred = dt_model.predict(X_val)

# Random Forest Classifier
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_val)

# Evaluate models
dt_accuracy = accuracy_score(y_val, dt_pred)
rf_accuracy = accuracy_score(y_val, rf_pred)

print(f'Decision Tree Accuracy: {dt_accuracy}')
print(f'Random Forest Accuracy: {rf_accuracy}')
```

```
Decision Tree Accuracy: 0.755
Random Forest Accuracy: 0.82
```

- **Logistic Regression & Improvement Method**

Logistic Regression

```
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import cross_val_score

# Feature Selection using Logistic Regression
selector = SelectFromModel(LogisticRegression())
X_train_selected = selector.fit_transform(X_train, y_train)
X_val_selected = selector.transform(X_val)

# Fit logistic regression on the selected features
logreg = LogisticRegression()
logreg.fit(X_train_selected, y_train)

# Predict on the validation set
y_pred = logreg.predict(X_val_selected)

# Evaluate the model
accuracy = accuracy_score(y_val, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Print classification report for more detailed evaluation
print("Classification Report:\n", classification_report(y_val, y_pred))
```

```
Accuracy: 0.71
Classification Report:
              precision    recall  f1-score   support

          No       0.73      0.69      0.71       104
         Yes       0.69      0.73      0.71        96

    accuracy                           0.71       200
   macro avg       0.71      0.71      0.71       200
weighted avg       0.71      0.71      0.71       200
```

```python
from sklearn.ensemble import VotingClassifier

voting_classifier = VotingClassifier(estimators=[
    ('logreg', logreg),
    ('rf', best_rf_model),
    # Add more models as needed
], voting='hard')

voting_classifier.fit(X_train_resampled, y_train_resampled)
y_pred_voting = voting_classifier.predict(X_val_scaled)

# Evaluate the ensemble model
accuracy_voting = accuracy_score(y_val, y_pred_voting)
print(f"Ensemble (Voting Classifier) Accuracy: {accuracy_voting:.2f}")

# Print classification report for more detailed evaluation
print("Classification Report (Voting Classifier):\n", classification_report(y_val, y_pred_voting))
```

```
Ensemble (Voting Classifier) Accuracy: 0.79
Classification Report (Voting Classifier):
              precision    recall  f1-score   support

          No       0.77      0.85      0.81       104
         Yes       0.81      0.73      0.77        96

    accuracy                           0.79       200
   macro avg       0.79      0.79      0.79       200
weighted avg       0.79      0.79      0.79       200
```

Conclusion

In conclusion, our analysis of the Medical Treatment Dataset aimed to predict the necessity of mental health treatment for individuals in the context of workplace-related factors. The machine learning models employed varied in accuracy performance, with Logistic Regression achieving the lowest accuracy at 71%, followed by Decision Tree at 75.5%, Random Forest at 82%, Neural Network at 85%, and Gradient Boosting exhibiting the highest accuracy at 86%. The confusion matrices provided insights into the strengths and weaknesses of each model, with ensemble methods like Voting Classifier significantly improving accuracy and reducing false negatives. Feature importance analyses highlighted the crucial role of 'work_interfere' in

predicting mental health treatment across Decision Tree and Random Forest models.