ANALYZING PATTERNS AND TRENDS IN ACCIDENTAL DRUG-RELATED DEATHS:

A HIGH-PERFORMANCE COMPUTING APPROACH TO SUBSTANCE DETECTION (2012-2023)

Vudem Shruthi Reddy

TABLE OF CONTENTS

SECTIONS	SUBSECTIONS
INTRODUCTION	Project Objective, Purpose of the Study, Approach to Data Analysis, Significance of the Study
DATA DESCRIPTION	Dataset Overview, Data Source, Substance Detection, Morphine (Not Heroin), "Any Opioid" Column
DATA ATTRIBUTES	Personal Information, Location Details, Substance Information

INTRODUCTION

Objective: Analyze patterns and trends in accidental drug-related deaths in Connecticut (2012-2023) using high-performance computing techniques.

Purpose: Utilize advanced data analysis to identify and study substances involved in overdose-related deaths, providing insight into the evolving drug crisis.

Approach: Apply statistical models and computational tools to detect trends in substance use and assess the impact of various drugs on public health.

Significance: Drive evidence-based decision-making for public health interventions, improve prevention strategies, and inform policy development regarding drug overdose deaths.

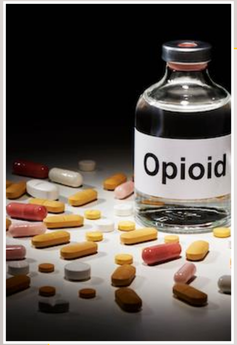
DATA DESCRIPTION

Dataset Overview: A comprehensive listing of accidental drug-related deaths in Connecticut from 2012 to 2023, with data on various substances involved in overdoses. The total number of records are 11982

Data Source: Derived from investigations conducted by the Office of the Chief Medical Examiner, incorporating death certificates, scene investigations, and toxicity reports.

Substance Detection: Each record includes a series of columns indicating the presence of specific substances (e.g., Heroin, Cocaine, Fentanyl, Methadone) detected in the deceased's system.

Morphine (Not Heroin): Includes cases where morphine is detected but cannot be distinguished from heroin due to the metabolic process; based on scene investigations, the cause of death may be categorized accordingly.



"Any Opioid" Column: Used when the Medical Examiner cannot determine whether morphine is from prescription or heroin-based sources; indicates general opioid involvement.

LINK: https://catalog.data.gov/dataset/accidental-drug-related-deaths-2012-2018

DATA ATTRIBUTES

PERSONAL INFORMATION	LOCATION DETAILS	SUBSTANCE INFORMATION
Date	Residence City	Heroin
Data Type	Residence County	Heroin death certificate(DC)
Age	Residence State	Cocaine
Sex	Injury City	Fentanyl
Race	Injury County	Fentanyl Analogue
Ethnicity	Injury State	Oxycodone
	Injury Place	Oxymorphone
	Death City	Ethanol
	Death County	Hydrocodone
	Death State	Benzodiazepine
	ResidenceCityGeo	Methadone
	IniuryCityGeo	Meth/Amphetamine
	DeathCityGeo	Amphet
		Tramad
		Hydromorphone

FEATURE SELECTION & INITIAL ANALYSIS

Selected Features:

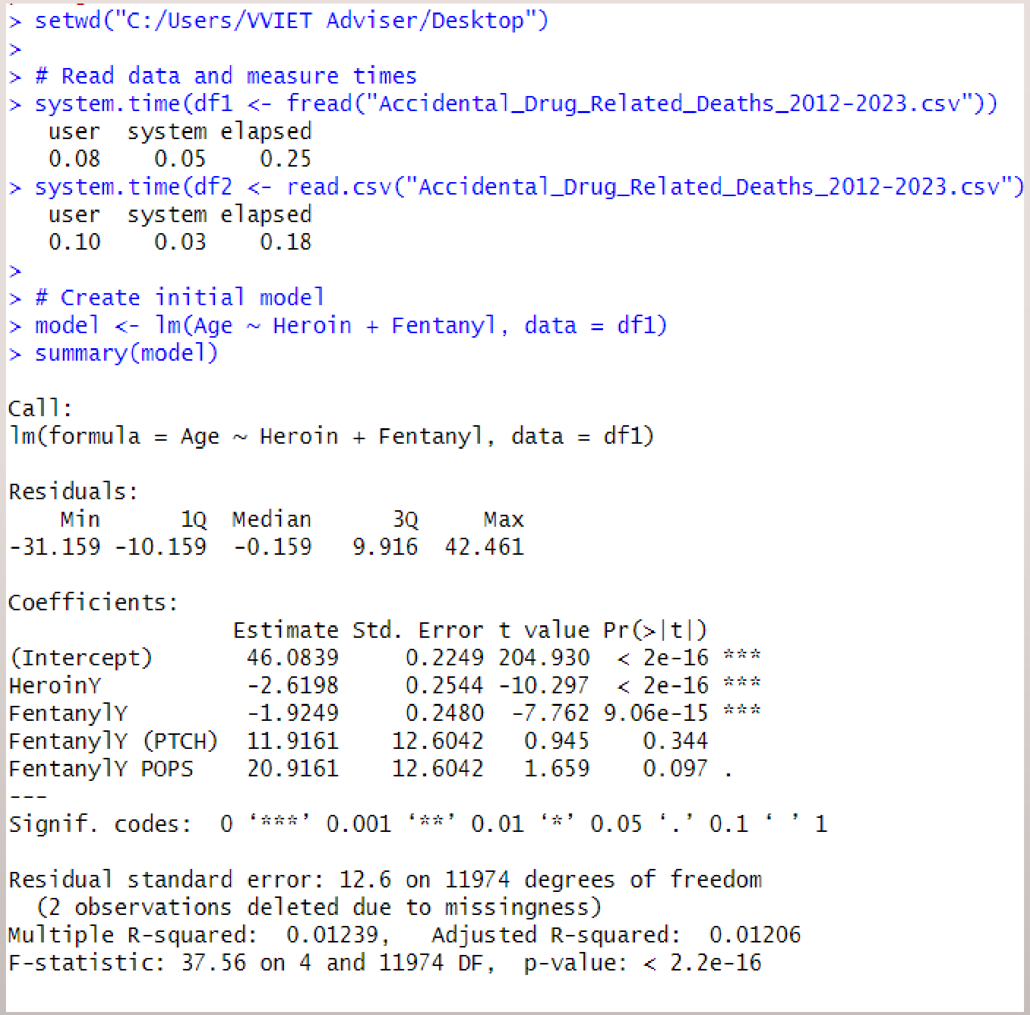
- Age (Dependent Variable)
- Heroin (Independent Variable)
- Fentanyl (Independent Variable)

Feature Importance using Linear Regression (LR) with read.csv and fread:

- Intercept: 46.08
- HeroinY: -2.62 (Significant)
- FentanylY: -1.92 (Significant)
- Fentanyl Y (PTCH): 11.92 (Not Significant)
- Fentanyl Y POPS: 20.91 (Marginally Significant)

Model Performance:

- Adjusted $R^2 = 0.01206$
- p-value < 2.2e-16 (Model is statistically significant)



```
> setwd("C:/Users/VVIET Adviser/Desktop")
>
> # Read data and measure times
> system.time(df1 <- fread("Accidental_Drug_Related_Deaths_2012-2023.csv"))
   user  system elapsed
   0.08    0.05    0.25
> system.time(df2 <- read.csv("Accidental_Drug_Related_Deaths_2012-2023.csv"))
   user  system elapsed
   0.10    0.03    0.18
>
> # Create initial model
> model <- lm(Age ~ Heroin + Fentanyl, data = df1)
> summary(model)

Call:
lm(formula = Age ~ Heroin + Fentanyl, data = df1)

Residuals:
    Min      1Q  Median      3Q     Max
-31.159 -10.159  -0.159   9.916  42.461

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)       46.0839     0.2249 204.930  < 2e-16 ***
HeroinY           -2.6198     0.2544 -10.297  < 2e-16 ***
FentanylY         -1.9249     0.2480  -7.762 9.06e-15 ***
FentanylY (PTCH)  11.9161    12.6042   0.945    0.344
FentanylY POPS    20.9161    12.6042   1.659    0.097 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 12.6 on 11974 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.01239,	Adjusted R-squared:  0.01206
F-statistic: 37.56 on 4 and 11974 DF,  p-value: < 2.2e-16
```

RUNNING TIME & IMPORTANCE VALUES BEFORE & AFTER PROFILING

Code Execution Time (Before Optimization):

- read.csv: 0.18 sec
- fread: 0.25 sec
- Linear Regression Model Execution Time: ~0 sec

Feature Importance (Before Profiling):

- Similar results for read.csv and fread
- No major time difference between both methods

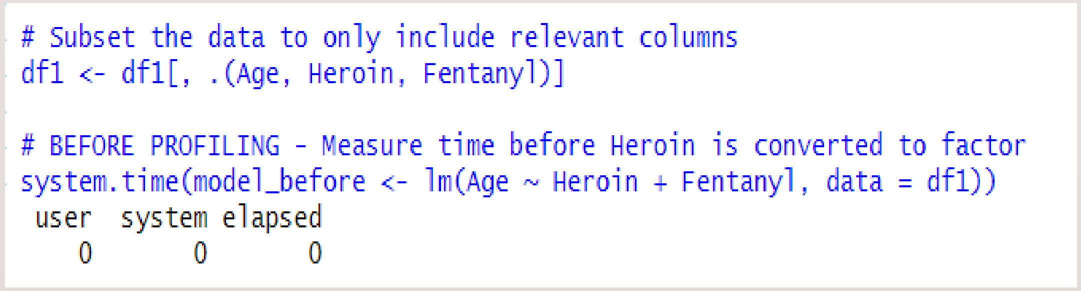
Optimization Step: Converted Heroin to a factor

Model Execution Time (After Profiling):

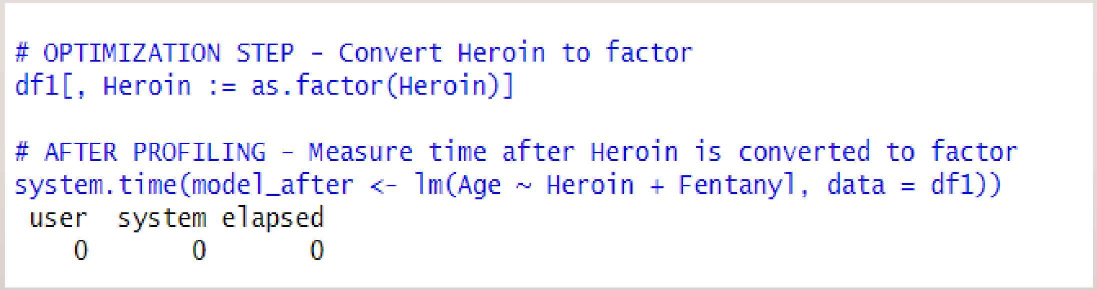
- Before Optimization: ~0 sec
- After Optimization: ~0 sec

Feature Importance (After Profiling):

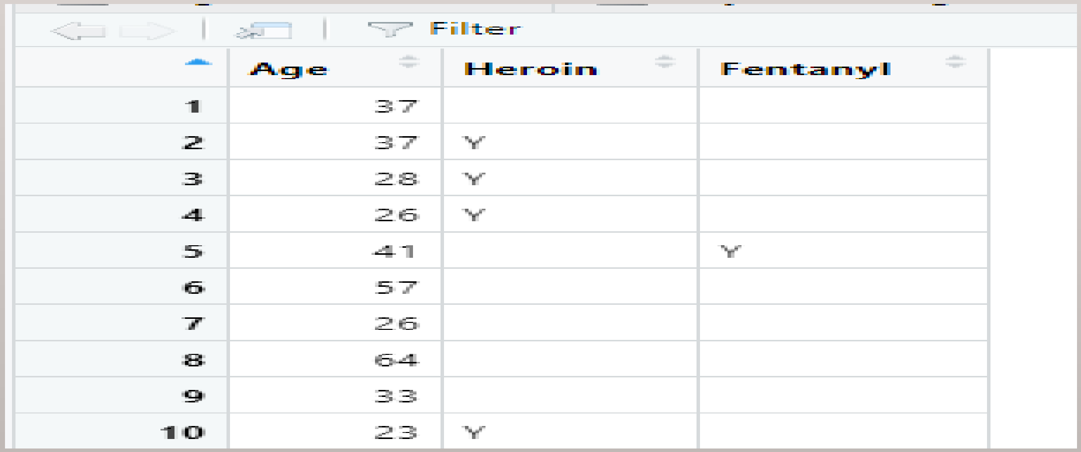
- No major change in coefficients or significance



```
# Subset the data to only include relevant columns
df1 <- df1[, .(Age, Heroin, Fentanyl)]

# BEFORE PROFILING - Measure time before Heroin is converted to factor
system.time(model_before <- lm(Age ~ Heroin + Fentanyl, data = df1))
 user  system elapsed 
    0       0       0 
```

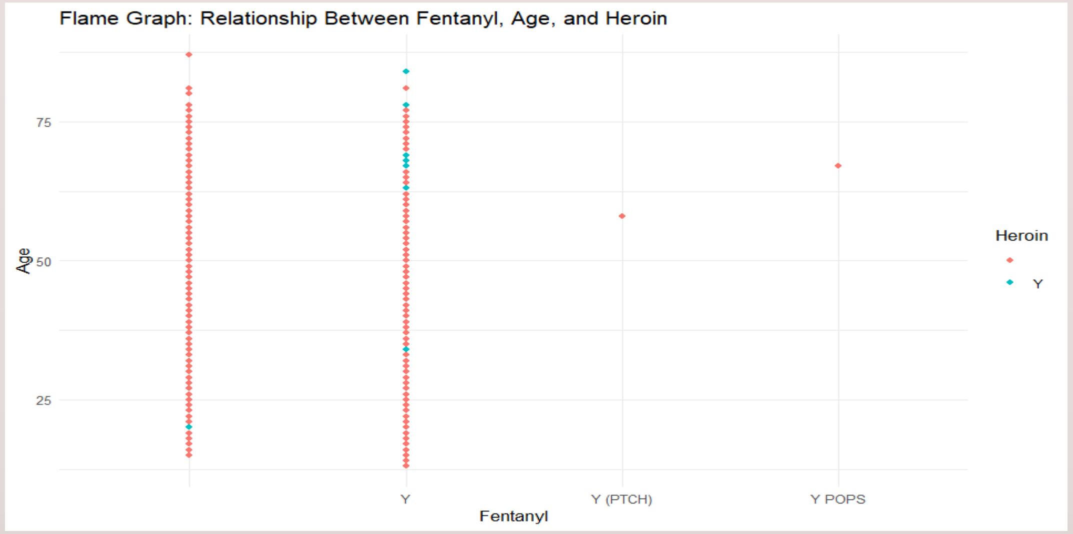


```
# OPTIMIZATION STEP - Convert Heroin to factor
df1[, Heroin := as.factor(Heroin)]

# AFTER PROFILING - Measure time after Heroin is converted to factor
system.time(model_after <- lm(Age ~ Heroin + Fentanyl, data = df1))
 user  system elapsed 
    0       0       0 
```



	Age	Heroin	Fentanyl
1	37		
2	37	Y	
3	28	Y	
4	26	Y	
5	41		Y
6	57		
7	26		
8	64		
9	33		
10	23	Y	

EXECUTION TIME COMPARISON & OPTIMIZATION ANALYSIS WITH FLAME GRAPH



Clear clustering patterns among different age groups.

Younger individuals (20s-30s) show higher Fentanyl involvement.

Older individuals (40s-50s) have more diverse substance involvement.

Some overlap suggests polysubstance use across different age groups.

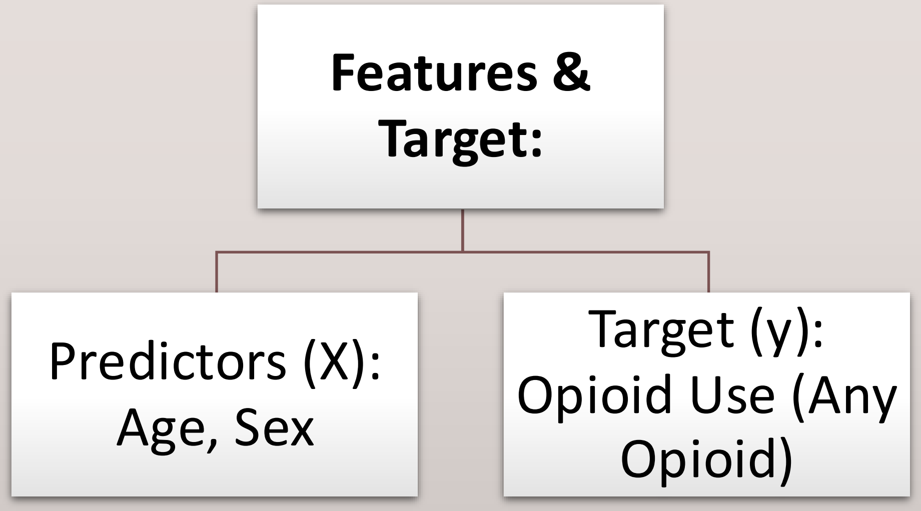
No major difference in execution time after profiling.

- Both fread and read.csv performed similarly, with fread being slightly faster (0.11s vs. 0.18s). However, the difference was not significant.

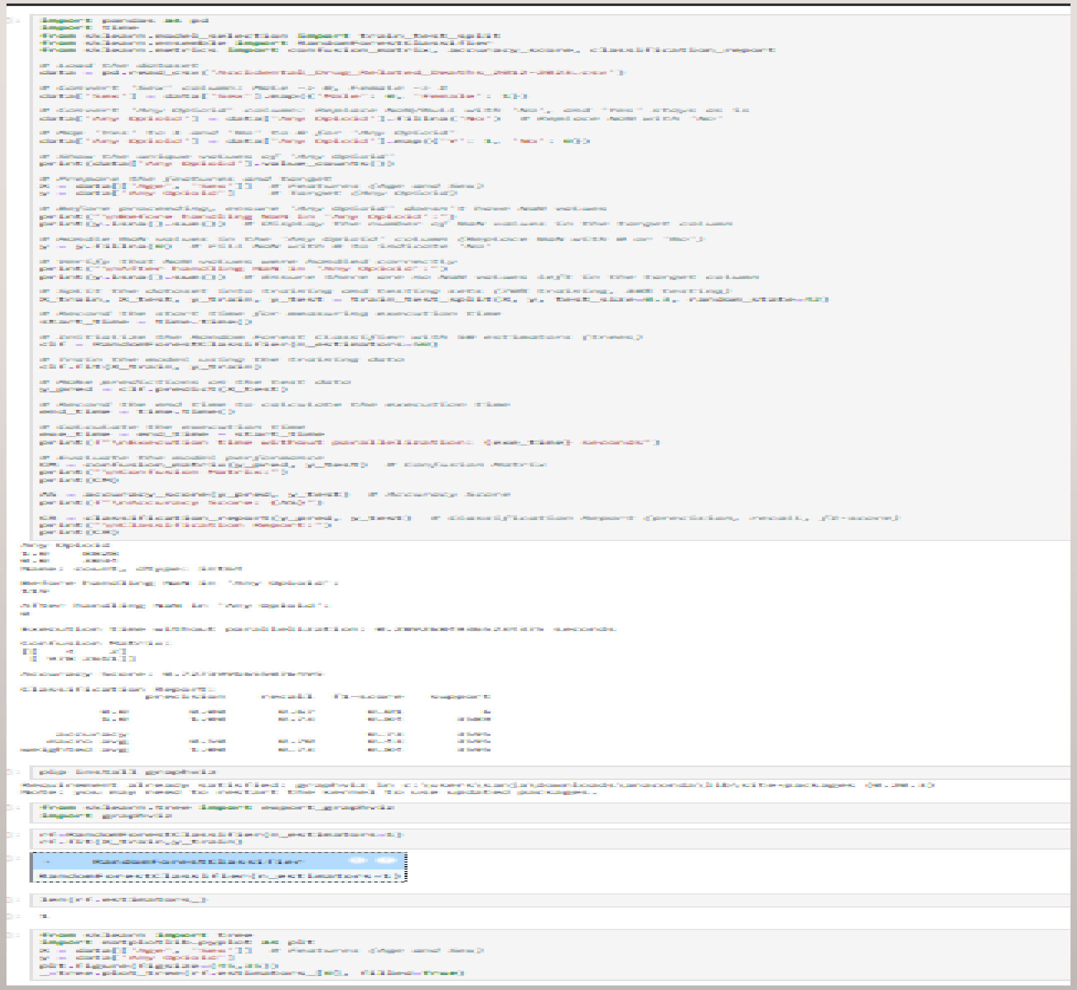
Optimization had no significant impact on runtime.

- Despite profiling and optimization, the runtime showed no major improvement, suggesting that the time cost is largely due to the data reading operation itself.

COMPARE RANDOM FOREST MODEL PERFORMANCE BEFORE & AFTER DASK PARALLELIZATION

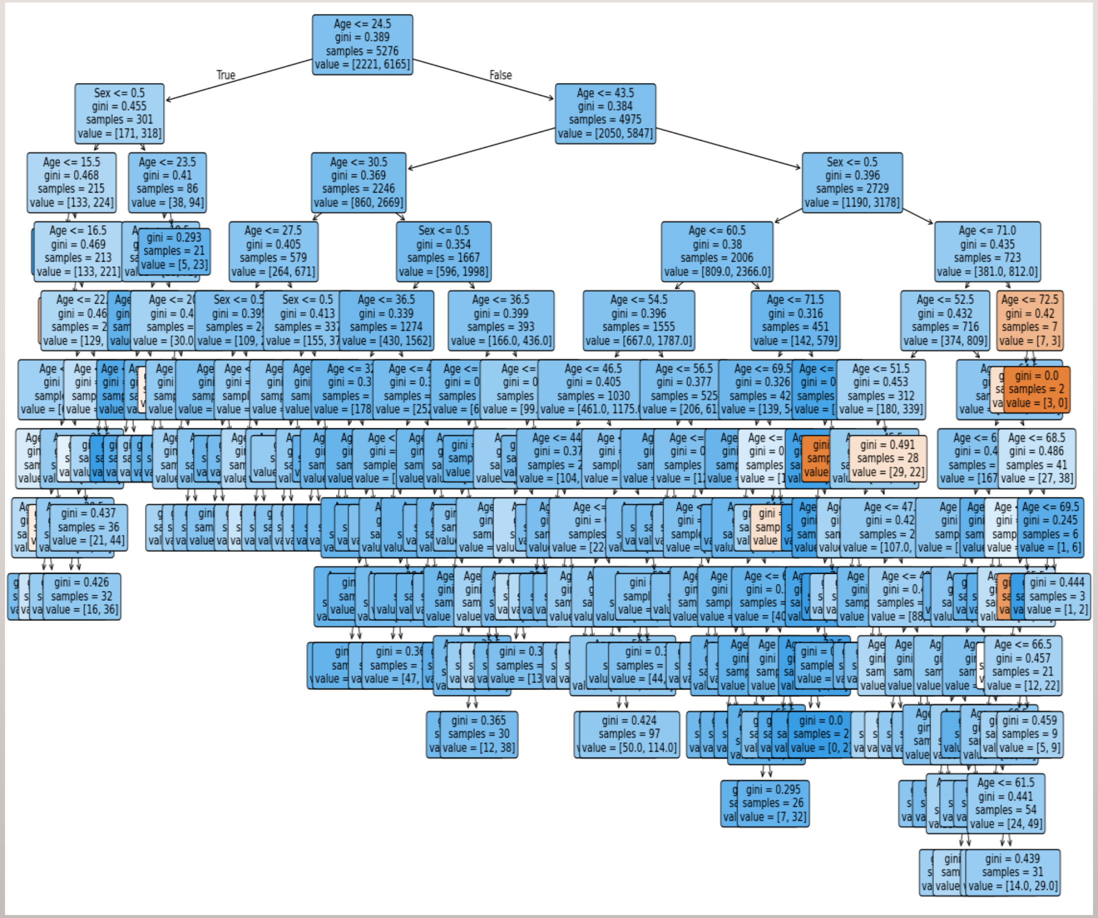


CODE & RESULTS(BEFORE DASK):



```
import pandas as pd
import time
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

[illegible]
```

GRAPH & INTERPRETATION (BEFORE DASK)



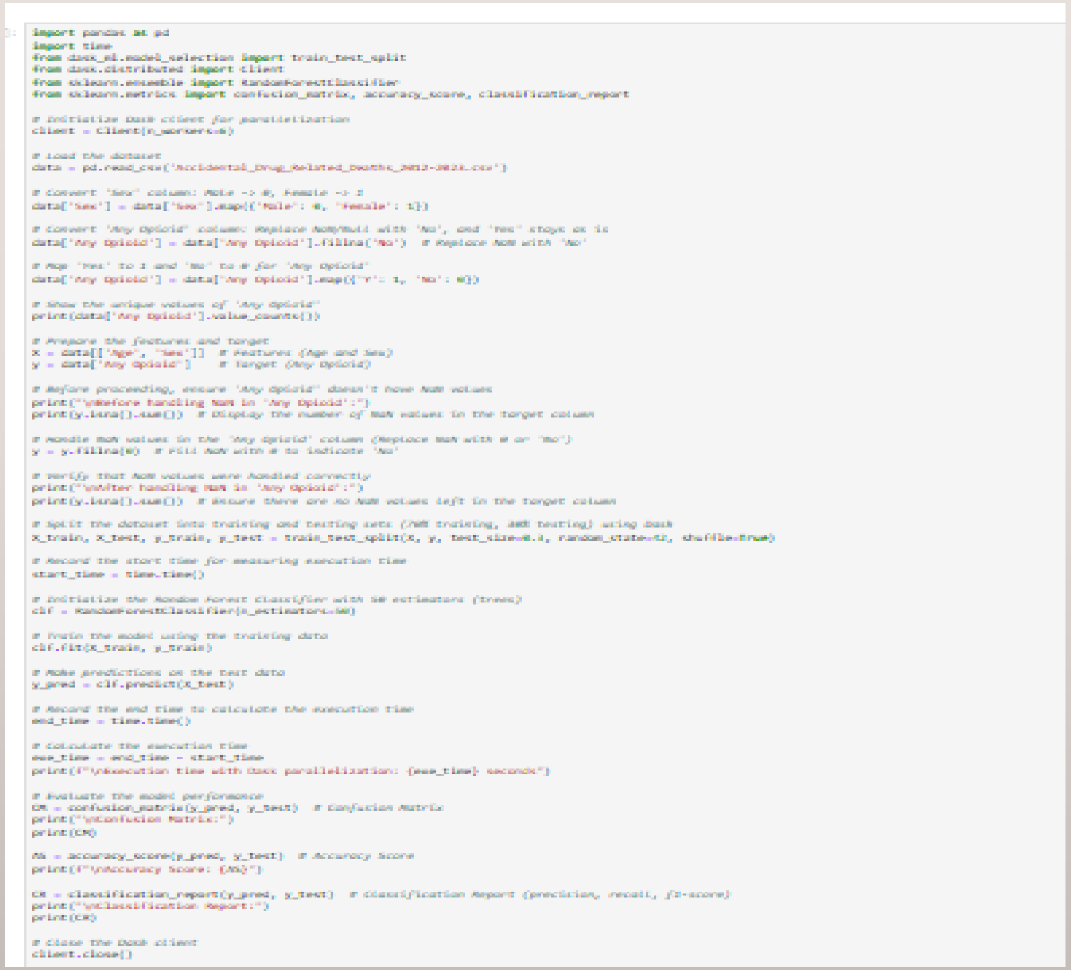
Execution Time & Accuracy: Model took 0.289 seconds with an accuracy of 72.74%.

Confusion Matrix: TN = 4, FP = 2, FN = 978, TP = 2611, showing a severe class imbalance.

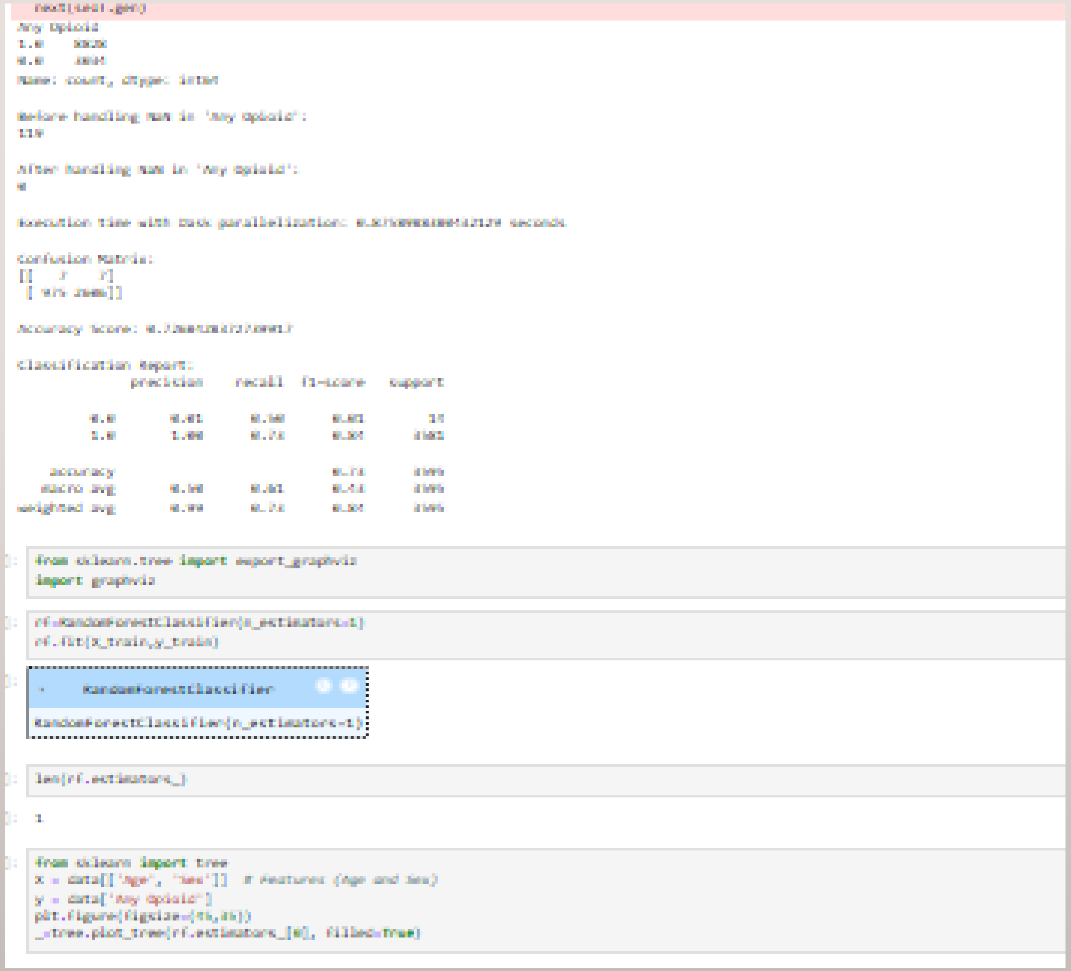
Precision & Recall: Poor classification for No Opioid Use (0.00 precision) but high precision for Opioid Use (1.00); recall for opioid use is 0.73 (73% correctly identified).

Graph Interpretation: Root node splits at Age ≤ 24.5 (moderate Gini impurity), further refined at Age ≤ 43.5, improving impurity reduction; mixed samples show higher Gini values, while pure classifications appear in terminal nodes.

CODES &RESULTS(AFTER DASK)



```
import pandas as pd
import time
from dask_ml.model_selection import train_test_split
from dask.distributed import Client
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

# Initialize Dask client for parallelization
client = Client(n_workers=4)

# Load the dataset
data = pd.read_csv('Accidental_Drug_Related_Deaths_2012-2023.csv')

# Convert 'Sex' column: Male -> 0, Female -> 1
data['Sex'] = data['Sex'].map({'Male': 0, 'Female': 1})

# Convert 'Any Opioid' column: Replace NaN/Null with 'No', and 'Yes' stays as is
data['Any Opioid'] = data['Any Opioid'].fillna('No')  # Replace NaN with 'No'

# Map 'Yes' to 1 and 'No' to 0 for 'Any Opioid'
data['Any Opioid'] = data['Any Opioid'].map({'Y': 1, 'No': 0})

# Show the unique values of 'Any Opioid'
print(data['Any Opioid'].value_counts())

# Prepare the features and target
X = data[['Age', 'Sex']]  # Features (Age and Sex)
y = data['Any Opioid']    # Target (Any Opioid)

# Before proceeding, ensure 'Any Opioid' doesn't have NaN values
print("\nBefore handling NaN in 'Any Opioid':")
print(y.isna().sum())  # Display the number of NaN values in the target column

# Handle NaN values in the 'Any Opioid' column (Replace NaN with 0 or 'No')
y = y.fillna(0)  # Fill NaN with 0 to indicate 'No'

# Verify that NaN values were handled correctly
print("\nAfter handling NaN in 'Any Opioid':")
print(y.isna().sum())  # Ensure there are no NaN values left in the target column

# Split the dataset into training and testing sets (70% training, 30% testing) using Dask
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, shuffle=True)

# Record the start time for measuring execution time
start_time = time.time()

# Initialize the Random Forest Classifier with 50 estimators (trees)
clf = RandomForestClassifier(n_estimators=50)

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

# Make predictions on the test data
y_pred = clf.predict(X_test)

# Record the end time to calculate the execution time
end_time = time.time()

# Calculate the execution time
exe_time = end_time - start_time
print(f"\nExecution time with Dask parallelization: {exe_time} seconds")

# Evaluate the model performance
CM = confusion_matrix(y_pred, y_test)  # Confusion Matrix
print("\nConfusion Matrix:")
print(CM)

AS = accuracy_score(y_pred, y_test)  # Accuracy Score
print(f"\nAccuracy Score: {AS}")

CR = classification_report(y_pred, y_test)  # Classification Report (precision, recall, f1-score)
print("\nClassification Report:")
print(CR)

# Close the Dask client
client.close()
```



```
    next(self.gen)
Any Opioid
1.0    [illegible]
0.0    [illegible]
Name: count, dtype: int64

Before handling NaN in 'Any Opioid':
119

After handling NaN in 'Any Opioid':
0

Execution time with Dask parallelization: [illegible] seconds

Confusion Matrix:
[[   7    7]
 [ [illegible] [illegible]]]

Accuracy Score: [illegible]

Classification Report:
              precision    recall  f1-score   support

         0.0       0.01      0.50      0.01        15
         1.0       1.00      0.72      0.83      [illegible]

    accuracy                           0.72      [illegible]
   macro avg       0.50      0.61      0.42      [illegible]
weighted avg       0.99      0.72      0.83      [illegible]
```

```
from sklearn.tree import export_graphviz
import graphviz
```

```
rf=RandomForestClassifier(n_estimators=1)
rf.fit(X_train,y_train)
```

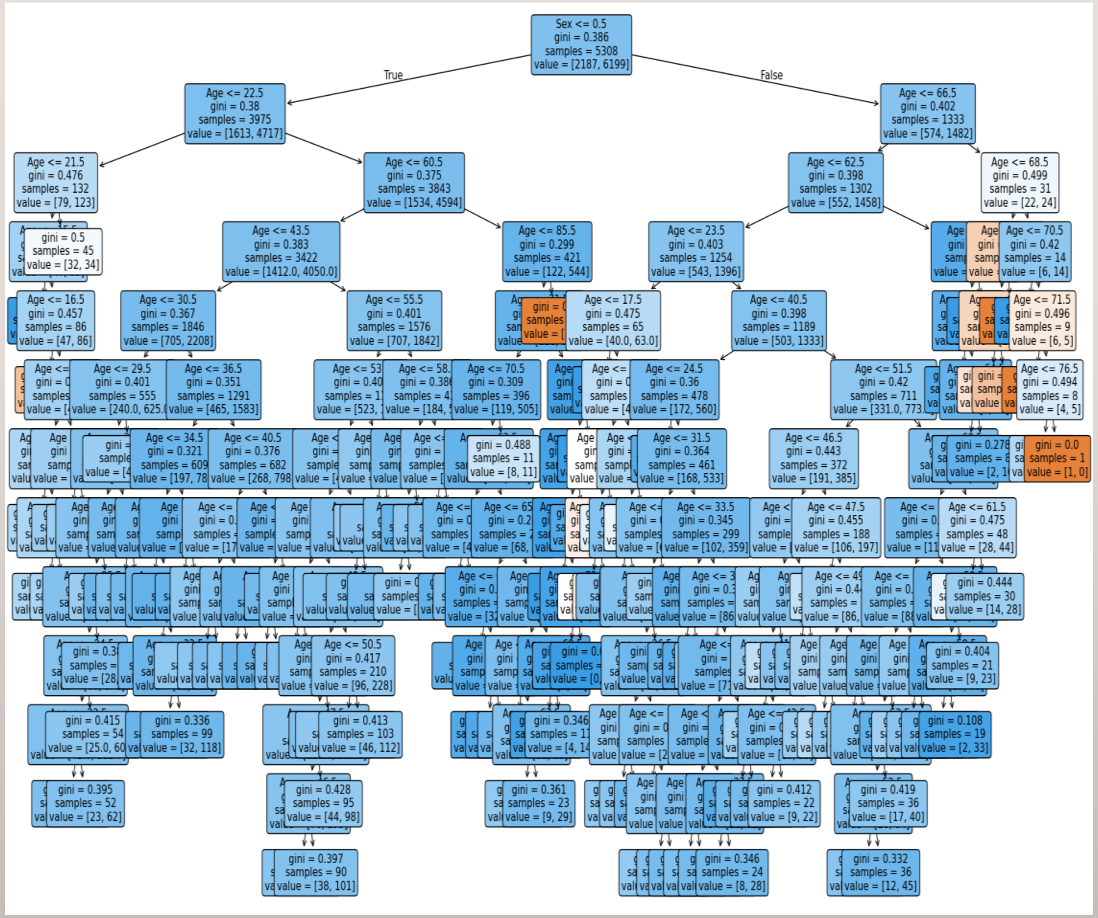
```
RandomForestClassifier
RandomForestClassifier(n_estimators=1)
```

```
len(rf.estimators_)
```

```
1
```

```
from sklearn import tree
X = data[['Age', 'Sex']]  # Features (Age and Sex)
y = data['Any Opioid']
plt.figure(figsize=(15,15))
_=tree.plot_tree(rf.estimators_[0], filled=True)
```

GRAPH & INTERPRETATION (AFTER DASK)



Execution Time & Accuracy: Model took 0.8759 seconds with an accuracy of 72.68%, slightly lower than before Dask.

Confusion Matrix: TN = 7, FP = 7, FN = 975, TP = 2606, showing minimal improvement in class imbalance.

Precision & Recall: Very poor classification for No Opioid Use (0.01 precision), while Opioid Use (Class 1) maintains high precision (1.00); recall for opioid use remains at 0.73 (73% correctly identified).

Graph Interpretation: The tree splits on Sex ≤ 0.5 (Gini = 0.386), with better impurity reduction at deeper levels, leading to some pure classifications (Gini = 0.0).

EVALUATING AND COMPARING RANDOM FOREST RESULTS BEFORE AND AFTER DASK

Before Dask: Execution time was 0.289 seconds with an accuracy of 72.74%. The model effectively identified opioid users (Class 1) but struggled with non-users due to class imbalance.

After Dask: Execution time increased to 0.8759 seconds with an accuracy of 72.68%. Dask introduced more refined tree splits, improving impurity reduction but not accuracy.

Graph Interpretation: Both models show similar splits, with slight improvements in impurity reduction after Dask, but no significant change in the model's ability to classify opioid use.

Which Method Was Faster? Before Dask was faster, as Dask's parallelization overhead slowed down execution for the small dataset.