Name: Uwase Ketsia Deborah
ID: 26244

PART 1: PROBLEM DEFINITION & PLANNING

I. Sector Selection

Cybersecurity

II. Problem Statement

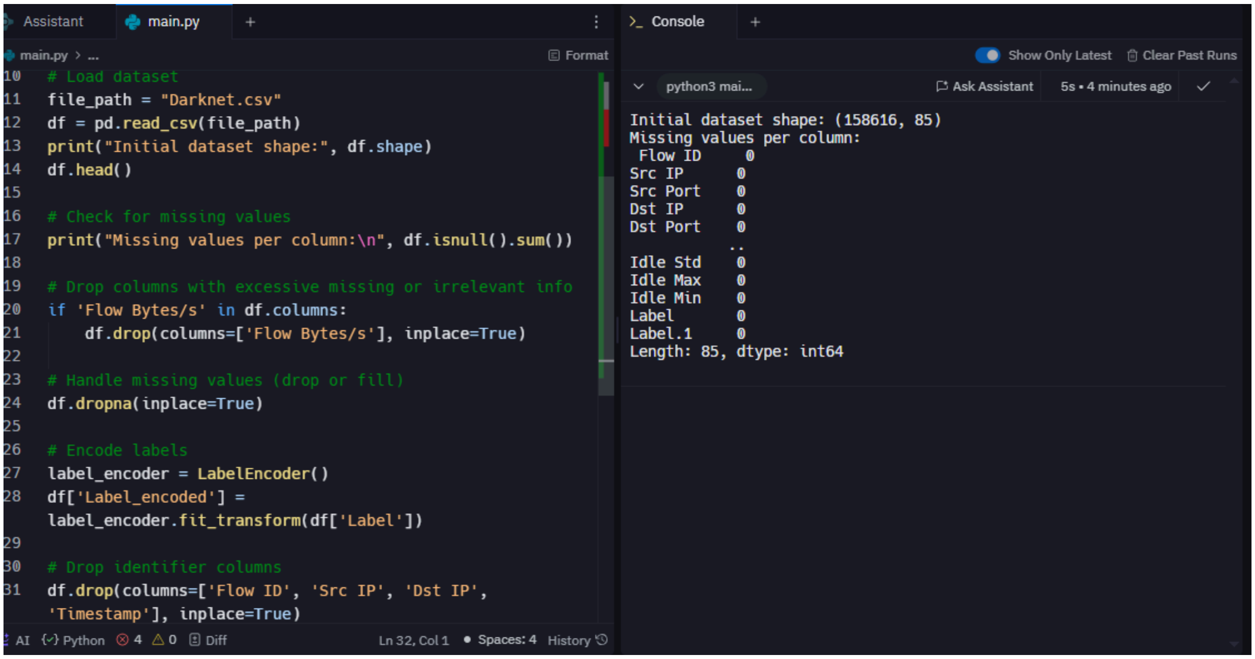
How can we use flow-based network data to accurately detect and differentiate between Tor and Non-Tor traffic in darknet environments?

III. 3. Dataset Identification

Dataset Title: Darknet
Source Link: [DarkNet]
Number of Rows and Columns: 158616, 85
Data Structure: csv file
Data Status: needs preprocessing.

PART 2: PYTHON ANALYTICS TASKS

1.Clean the Dataset

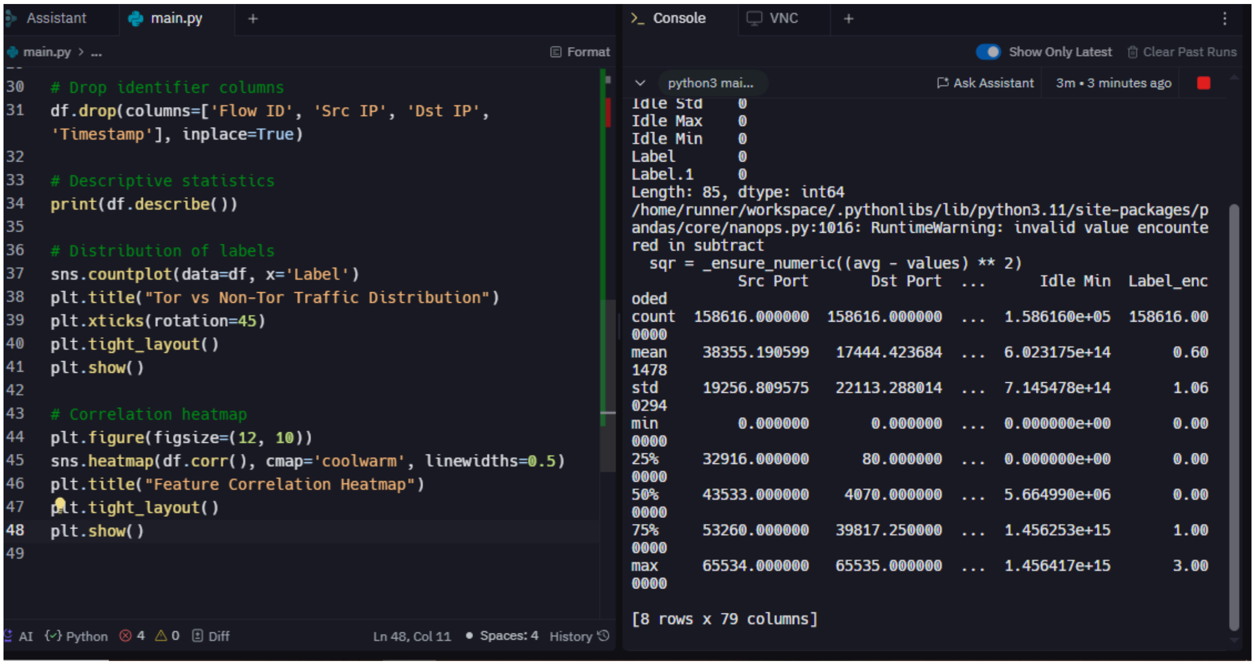


```python
# Load dataset
file_path = "Darknet.csv"
df = pd.read_csv(file_path)
print("Initial dataset shape:", df.shape)
df.head()

# Check for missing values
print("Missing values per column:\n", df.isnull().sum())

# Drop columns with excessive missing or irrelevant info
if 'Flow Bytes/s' in df.columns:
    df.drop(columns=['Flow Bytes/s'], inplace=True)

# Handle missing values (drop or fill)
df.dropna(inplace=True)

# Encode labels
label_encoder = LabelEncoder()
df['Label_encoded'] =
label_encoder.fit_transform(df['Label'])

# Drop identifier columns
df.drop(columns=['Flow ID', 'Src IP', 'Dst IP',
'Timestamp'], inplace=True)
```

```
Initial dataset shape: (158616, 85)
Missing values per column:
 Flow ID     0
Src IP      0
Src Port    0
Dst IP      0
Dst Port    0
           ..
Idle Std    0
Idle Max    0
Idle Min    0
Label       0
Label.1     0
Length: 85, dtype: int64
```

The code loads the **Darknet dataset** (158,616 rows × 85 columns), checks for missing values (none found), removes an irrelevant `"Flow Bytes/s"` column, drops any NaN rows, encodes the `"Label"` column into numbers, and deletes identifier columns like IP addresses and timestamps to keep only useful features for analysis or modeling.

2. Conduct Exploratory Data Analysis (EDA)



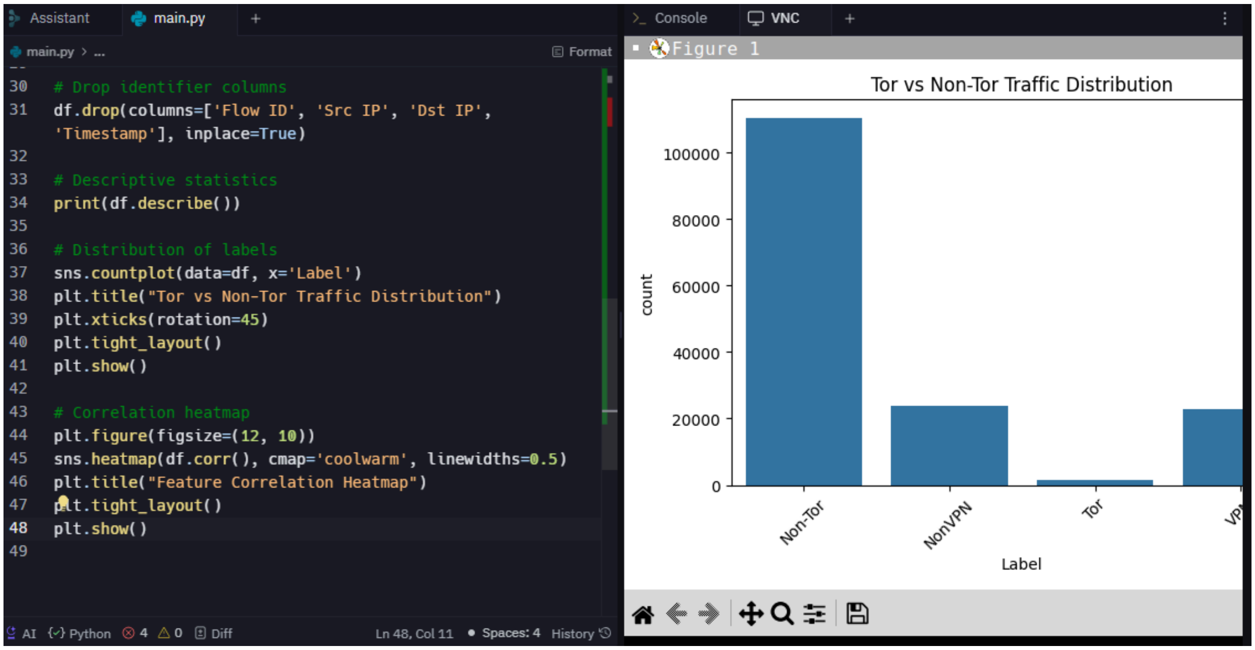
```python
# Drop identifier columns
df.drop(columns=['Flow ID', 'Src IP', 'Dst IP',
'Timestamp'], inplace=True)

# Descriptive statistics
print(df.describe())

# Distribution of labels
sns.countplot(data=df, x='Label')
plt.title("Tor vs Non-Tor Traffic Distribution")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Correlation heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(), cmap='coolwarm', linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.tight_layout()
plt.show()
```

```
Idle Std     0
Idle Max     0
Idle Min     0
Label        0
Label.1      0
Length: 85, dtype: int64
/home/runner/workspace/.pythonlibs/lib/python3.11/site-packages/p
andas/core/nanops.py:1016: RuntimeWarning: invalid value encounte
red in subtract
  sqr = _ensure_numeric((avg - values) ** 2)
            Src Port       Dst Port  ...      Idle Min  Label_enc
oded
count  158616.000000  158616.000000  ...  1.586160e+05  158616.00
0000
mean    38355.190599   17444.423684  ...  6.023175e+14       0.60
1478
std     19256.809575   22113.288014  ...  7.145478e+14       1.06
0294
min         0.000000       0.000000  ...  0.000000e+00       0.00
0000
25%     32916.000000      80.000000  ...  0.000000e+00       0.00
0000
50%     43533.000000    4070.000000  ...  5.664990e+06       0.00
0000
75%     53260.000000   39817.250000  ...  1.456253e+15       1.00
0000
max     65534.000000   65535.000000  ...  1.456417e+15       3.00
0000

[8 rows x 79 columns]
```

The code shows that after cleaning, the dataset has 79 usable columns and 158,616 records.

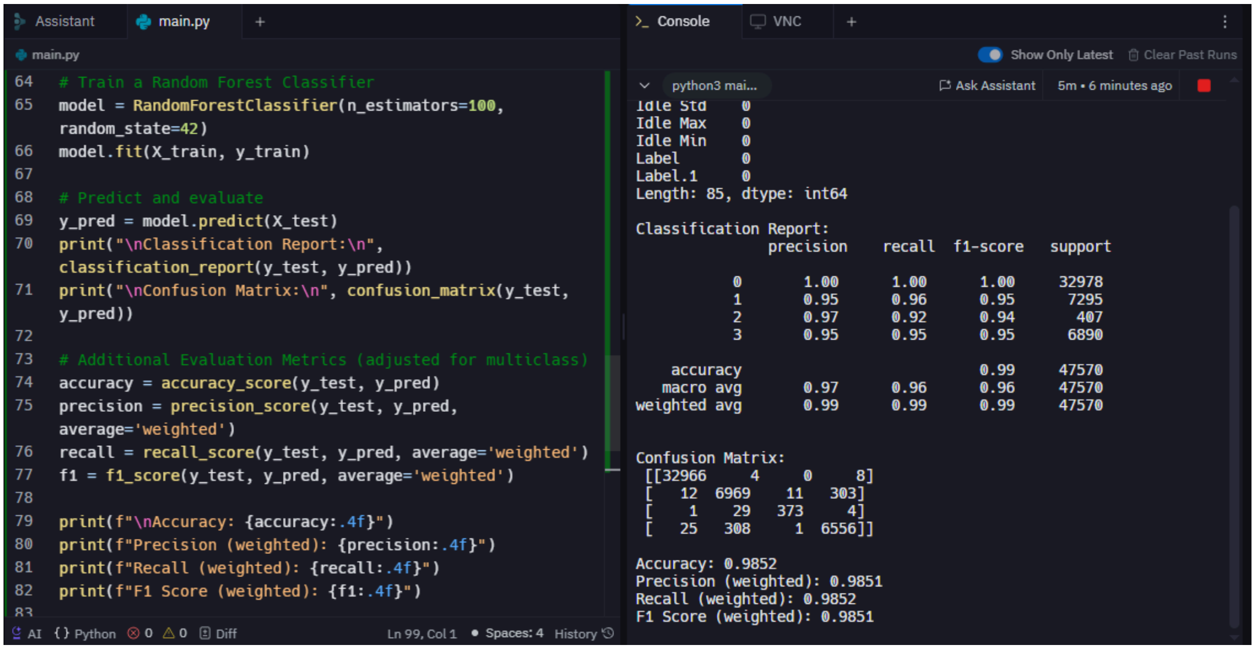
The numeric summary (`df.describe()`) gives ranges, averages, and variation for each feature, showing wide value differences between network traffic attributes.

The label distribution plot reveals how balanced Tor vs Non-Tor classes are.



```python
# Drop identifier columns
df.drop(columns=['Flow ID', 'Src IP', 'Dst IP', 'Timestamp'], inplace=True)

# Descriptive statistics
print(df.describe())

# Distribution of labels
sns.countplot(data=df, x='Label')
plt.title("Tor vs Non-Tor Traffic Distribution")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Correlation heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(), cmap='coolwarm', linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.tight_layout()
plt.show()
```

3. Machine learning model



```python
# Train a Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Additional Evaluation Metrics (adjusted for multiclass)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print(f"\nAccuracy: {accuracy:.4f}")
print(f"Precision (weighted): {precision:.4f}")
print(f"Recall (weighted): {recall:.4f}")
print(f"F1 Score (weighted): {f1:.4f}")
```

```
Idle Std      0
Idle Max      0
Idle Min      0
Label         0
Label.1       0
Length: 85, dtype: int64

Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00     32978
           1       0.95      0.96      0.95      7295
           2       0.97      0.92      0.94       407
           3       0.95      0.95      0.95      6890

    accuracy                           0.99     47570
   macro avg       0.97      0.96      0.96     47570
weighted avg       0.99      0.99      0.99     47570


Confusion Matrix:
 [[32966     4     0     8]
 [   12  6969    11   303]
 [    1    29   373     4]
 [   25   308     1  6556]]

Accuracy: 0.9852
Precision (weighted): 0.9851
Recall (weighted): 0.9852
F1 Score (weighted): 0.9851
```

The code trains a Random Forest Classifier on the darknet dataset to classify network traffic and evaluates its performance using a classification report, confusion matrix, and weighted metrics. The model achieved about **98.5% accuracy**, with high precision, recall, and F1-scores across all classes, showing it reliably distinguishes normal and potentially malicious traffic. The confusion matrix indicates very few misclassifications, confirming the model is highly effective for darknet traffic detection.