In [1]:
```
import pandas as pd
```

In [2]:
```
# Importing Datasets
dataset=pd.read_csv("C:/Users/RANJANA/Downloads/alert management dataset final.csv")
```

In [3]:
```
dataset.head()
```

Out[3]:

	Alert ID	Alert Time	Date	Alert Type	Alert Description	Alert Source	Status	Severity	Reponse Time
0	6324	12:31:52	31-05-2023	Hardware Issue	Memory module failure	Source2	Open	High	16
1	7908	12:31:52	01-06-2023	Server Down	Server not responding	Source10	Open	High	17
2	6516	12:31:52	01-06-2023	Server Down	Server not responding	Source7	Open	High	19
3	4191	12:31:52	05-06-2023	Database Issue	Index corruption	Source4	Closed	High	17
4	8100	12:31:52	06-06-2023	Database Issue	Database connection timeout	Source7	Closed	Low	24

In [4]:
```
# Metrics Calculations
dataset.describe()
```

Out[4]:

	Alert ID	Reponse Time
count	1999.000000	1999.000000
mean	5390.144572	20.183092
std	2580.664217	3.122698
min	1000.000000	15.000000
25%	3203.000000	17.000000
50%	5271.000000	20.000000
75%	7573.000000	23.000000
max	9998.000000	25.000000

In [5]:
```
import matplotlib.pyplot as plt
```

In [6]:
```
# Unique values
for column in dataset.columns:
    unique=dataset[column].unique()
print(f"{column}:{unique} \n")
```

```
Reponse Time:[16 17 19 24 15 20 22 25 21 23 18]
```

In [7]:
```
# Text Preprocessing
import pandas as pd
dataset=pd.read_csv("C:/Users/RANJANA/Downloads/alert management dataset final.csv")
unique_categories = dataset['Alert Type'].unique()
Alert_Type_dict = {}

for i,Alert_Type in enumerate(unique_categories):
    Alert_Type_dict[Alert_Type] = i
dataset['Alert Type']=dataset['Alert Type'].map(Alert_Type_dict)
```

In [8]:
```
# conversions of object into int
import re
dataset['Severity_']=dataset['Severity'].replace({"Low":1,"Medium":2,"High":3})
dataset['Status_']=dataset['Status'].replace({"Open":1,"Closed":2})
dataset['Alert Source_']=dataset['Alert Source'].replace({"Source1":1,"Source2":2,"Source3":3,"Source4":4,"Source5":5,"Source6":6,"Source7":7,"Source8":8,"Sou
dataset['Alert Type_']=dataset['Alert Type'].replace({"Hardware_Issue":1,"Server_Down":2,"Database_Issue":3,"Network_Issue":4,"Data_Corruption":5,"Power_Outag
dataset['Alert Description_']=dataset['Alert Description'].replace({"Memory module failure":0,"Server not responding":1,"Index corruption":2,"Database connect
                                                                    "Complete power loss":5,"UPS failure":6,"Power supply failure":7,"CPU fan failure":8,"Slow
                                                                    "Packet loss on network interface":11,"Application crashed unexpectedly":12,"Disk read/wri
                                                                    "Network connectivity failure":15,"Critical software bug detected":16,"Router configuratio
                                                                    "File system integrity check failed":19,"Slow network performance":20})

print(dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1999 entries, 0 to 1998
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Alert ID            1999 non-null   int64
 1   Alert Time          1999 non-null   object
 2   Date                1999 non-null   object
 3   Alert Type          1999 non-null   int64
 4   Alert Description   1999 non-null   object
 5   Alert Source        1999 non-null   object
 6   Status              1999 non-null   object
 7   Severity            1999 non-null   object
 8   Reponse Time        1999 non-null   int64
 9   Severity_           1999 non-null   int64
 10  Status_             1999 non-null   int64
 11  Alert Source_       1999 non-null   int64
 12  Alert Type_         1999 non-null   int64
 13  Alert Description_  1999 non-null   int64
dtypes: int64(8), object(6)
memory usage: 218.8+ KB
None
```

In [9]:
```
dataset=dataset.drop(columns=['Alert ID','Alert Time'])
dataset.to_csv("Preprocess_data.csv")
print(dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1999 entries, 0 to 1998
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Date                1999 non-null   object
 1   Alert Type          1999 non-null   int64
 2   Alert Description   1999 non-null   object
 3   Alert Source        1999 non-null   object
 4   Status              1999 non-null   object
 5   Severity            1999 non-null   object
 6   Reponse Time        1999 non-null   int64
 7   Severity_           1999 non-null   int64
 8   Status_             1999 non-null   int64
 9   Alert Source_       1999 non-null   int64
 10  Alert Type_         1999 non-null   int64
 11  Alert Description_  1999 non-null   int64
dtypes: int64(7), object(5)
memory usage: 187.5+ KB
None
```

In [10]:
```
dataset.head()
```

Out[10]:

	Date	Alert Type	Alert Description	Alert Source	Status	Severity	Reponse Time	Severity_	Status_	Alert Source_	Alert Type_	Alert Description_
0	31-05-2023	0	Memory module failure	Source2	Open	High	16	3	1	2	0	0
1	01-06-2023	1	Server not responding	Source10	Open	High	17	3	1	10	1	1
2	01-06-2023	1	Server not responding	Source7	Open	High	19	3	1	7	1	1
3	05-06-2023	2	Index corruption	Source4	Closed	High	17	3	2	4	2	2
4	06-06-2023	2	Database connection timeout	Source7	Closed	Low	24	1	2	7	2	3

In [11]:
```
#RANDOM FOREST ALGORITHM
#X and y data
X=dataset.iloc[:,[9,11,8]]
y=dataset.iloc[:,10]
```

In [12]:
```
from sklearn.model_selection import train_test_split
X_train , X_test , y_train , y_test =train_test_split(X,y,test_size=0.3,random_state=50)
```

In [13]:
```
#Random Forest
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Create a Random Forest classifier
model = RandomForestClassifier(n_estimators=10,random_state=42)

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

# Predict on the test set
y_pred = model.predict(X_test)


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

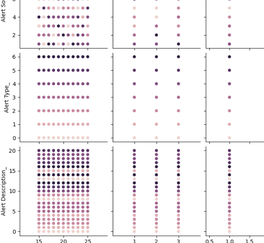
```
Accuracy: 0.98
```

In [14]:
```
# CROSS-VALIDATION
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.linear_model import LogisticRegression

# Create a KFold object
kf = KFold(n_splits=8, shuffle=True, random_state=42)

# Perform k-fold cross-validation
scores = cross_val_score(model, X_test, y_test, cv=kf)

# Print the accuracy scores for each fold
for fold, score in enumerate(scores):
    print(f"Fold {fold+1}: Accuracy = {score}")

# Calculate and print the average accuracy across all folds
average_accuracy = scores.mean()
print(f"Average Accuracy: {average_accuracy}")
```

```
Fold 1: Accuracy = 0.9333333333333333
Fold 2: Accuracy = 0.8933333333333333
Fold 3: Accuracy = 0.92
Fold 4: Accuracy = 0.84
Fold 5: Accuracy = 0.9333333333333333
Fold 6: Accuracy = 0.84
Fold 7: Accuracy = 0.9066666666666666
Fold 8: Accuracy = 0.92
Average Accuracy: 0.8983333333333334
```

In [15]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import sklearn.metrics as metrics
from sklearn.tree import export_graphviz
```

In [16]:
```
# Confusion Matrix
y_pred = model.predict(X_test)
confusion_matrix = metrics.confusion_matrix(y_test, y_pred)
sns.heatmap(confusion_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```



In [21]:
```
# Data visualization
sns.pairplot(dataset, hue='Alert Type')
sns
```

Out[21]:
```
<module 'seaborn' from 'C:\\ProgramData\\anaconda3\\lib\\site-packages\\seaborn\\__init__.py'>
```

In [ ]: