In [5]:

```
import numpy as np
import pandas as pd
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow.keras import regularizers
import xgboost as xgb
from sklearn.decomposition import PCA
from sklearn import tree
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import RobustScaler
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import metrics
pd.set_option('display.max_columns',None)
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [6]:

```
df = pd.read_csv(r"C:\Users\user\Downloads\KDDTest+.txt\KDDTest+.txt")
```

In [7]:

```
# Check data
df.head()
```

Out[7]:

	0	tcp	private	REJ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.10	0.11	0.12	0.13
0	0	tcp	private	REJ	0	0	0	0	0	0	0	0	0	0	0	0	0
1	2	tcp	ftp_data	SF	12983	0	0	0	0	0	0	0	0	0	0	0	0
2	0	icmp	eco_i	SF	20	0	0	0	0	0	0	0	0	0	0	0	0
3	1	tcp	telnet	RSTO	0	15	0	0	0	0	0	0	0	0	0	0	0
4	0	tcp	http	SF	267	14515	0	0	0	0	0	1	0	0	0	0	0

In [8]:

```
columns = (['duration','protocol_type','service','flag','src_bytes','dst_b
,'num_failed_logins','logged_in','num_compromised','root_shell','su_attemp
,'num_shells','num_access_files','num_outbound_cmds','is_host_login','is_g
,'srv_serror_rate','rerror_rate','srv_rerror_rate','same_srv_rate','diff_s
,'dst_host_same_srv_rate','dst_host_diff_srv_rate','dst_host_same_src_port
,'dst_host_srv_serror_rate','dst_host_rerror_rate','dst_host_srv_rerror_ra
```

In [9]:

```
# Assign name for columns
df.columns = columns
```

In [10]:

```
df.head()
```

Out[10]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent
0	0	tcp	private	REJ	0	0	0	0	0
1	2	tcp	ftp_data	SF	12983	0	0	0	0
2	0	icmp	eco_i	SF	20	0	0	0	0
3	1	tcp	telnet	RSTO	0	15	0	0	0
4	0	tcp	http	SF	267	14515	0	0	0

In [11]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22543 entries, 0 to 22542
Data columns (total 43 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   duration                     22543 non-null  int64  
 1   protocol_type                22543 non-null  object 
 2   service                      22543 non-null  object 
 3   flag                         22543 non-null  object 
 4   src_bytes                    22543 non-null  int64  
 5   dst_bytes                    22543 non-null  int64  
 6   land                         22543 non-null  int64  
 7   wrong_fragment               22543 non-null  int64  
 8   urgent                       22543 non-null  int64  
 9   hot                          22543 non-null  int64  
 10  num_failed_logins            22543 non-null  int64  
 11  logged_in                    22543 non-null  int64  
 12  num_compromised              22543 non-null  int64  
 13  root_shell                   22543 non-null  int64  
 14  su_attempted                 22543 non-null  int64  
 15  num_root                     22543 non-null  int64  
 16  num_file_creations           22543 non-null  int64  
 17  num_shells                   22543 non-null  int64  
 18  num_access_files             22543 non-null  int64  
 19  num_outbound_cmds            22543 non-null  int64  
 20  is_host_login                22543 non-null  int64  
 21  is_guest_login               22543 non-null  int64  
 22  count                        22543 non-null  int64  
 23  srv_count                    22543 non-null  int64  
 24  serror_rate                  22543 non-null  float64
 25  srv_serror_rate              22543 non-null  float64
 26  rerror_rate                  22543 non-null  float64
 27  srv_rerror_rate              22543 non-null  float64
 28  same_srv_rate                22543 non-null  float64
 29  diff_srv_rate                22543 non-null  float64
 30  srv_diff_host_rate           22543 non-null  float64
 31  dst_host_count               22543 non-null  int64  
 32  dst_host_srv_count           22543 non-null  int64  
 33  dst_host_same_srv_rate       22543 non-null  float64
 34  dst_host_diff_srv_rate       22543 non-null  float64
 35  dst_host_same_src_port_rate  22543 non-null  float64
 36  dst_host_srv_diff_host_rate  22543 non-null  float64
 37  dst_host_serror_rate         22543 non-null  float64
 38  dst_host_srv_serror_rate     22543 non-null  float64
 39  dst_host_rerror_rate         22543 non-null  float64
 40  dst_host_srv_rerror_rate     22543 non-null  float64
 41  outcome                      22543 non-null  object 
 42  level                        22543 non-null  int64  
dtypes: float64(15), int64(24), object(4)
memory usage: 7.4+ MB
```

In [12]:

```
processed_data_bat_full = pd.DataFrame(df)

# Specify the path where you want to save the CSV file
csv_file_path = 'processed_data_bat_full.csv'

# Save the DataFrame to a CSV file
processed_data_bat_full.to_csv(csv_file_path, index=False)

# Print a message indicating successful saving
print(f"DataFrame saved to '{csv_file_path}'")
```

```
DataFrame saved to 'processed_data_bat_full.csv'
```

In [13]:

```
import mysql.connector
from sqlalchemy import create_engine
import pandas as pd

# Replace 'your_username', 'your_password', 'your_database', and 'your_hos
db_connection = mysql.connector.connect(
    user='root',
    password='12345',
    host='localhost',
    database='bat_algo_database'
)

# Path to your CSV file
csv_file_path = r'C:\Users\user\BAT_ALGORITHM_FULL\processed_data_bat_full

# Read CSV file into a pandas DataFrame
df = pd.read_csv(csv_file_path)

# Define the table name (avoid spaces)
table_name = 'data_collected'

# Create SQLAlchemy engine
engine = create_engine('mysql+mysqlconnector://root:12345@localhost/bat_al

try:
    # Create MySQL table based on DataFrame structure
    df[:0].to_sql(table_name, con=engine, index=False, if_exists='replace'

    # Load data into MySQL table
    df.to_sql(table_name, con=engine, index=False, if_exists='append')

    print(f'Data has been successfully loaded into the {table_name} table.

except Exception as e:
    print(f'Error: {str(e)}')

finally:
    # Close the database connection
    db_connection.close()

```

```
Data has been successfully loaded into the data_collected table.
```

In [14]:

```
table_name = 'data_collected'

# Create SQLAlchemy engine
engine = create_engine('mysql+mysqlconnector://root:12345@localhost/bat_al

# Query data from MySQL table into a DataFrame
query = f"SELECT * FROM {table_name}"
df = pd.read_sql(query, con=engine)

# Define the path to save the CSV file
csv_file_path = r'C:\Users\user\BAT_ALGORITHM_FULL\exported_data.csv'

# Write the DataFrame to CSV
df.to_csv(csv_file_path, index=False)

print(f'Data from the {table_name} table has been successfully exported tc
```

```
Data from the data_collected table has been successfully exported to C:\User
s\user\BAT_ALGORITHM_FULL\exported_data.csv.
```

In [16]:

```
db_connection.close()
```

In [30]:

```
print(df)
data_train=df
```

```
      protocol_type   service  flag  land  logged_in  is_host_login  \
0               tcp   private   REJ     0          0              0   
1               tcp  ftp_data    SF     0          0              0   
2              icmp     eco_i    SF     0          0              0   
3               tcp    telnet  RSTO     0          0              0   
4               tcp      http    SF     0          1              0   
...             ...       ...   ...   ...        ...            ...   
22538           tcp      smtp    SF     0          1              0   
22539           tcp      http    SF     0          1              0   
22540           tcp      http    SF     0          1              0   
22541           udp  domain_u    SF     0          0              0   
22542           tcp    sunrpc   REJ     0          0              0   

       is_guest_login outcome  level  duration   src_bytes  dst_bytes  \
0                   0       1     21       0.0   -0.188153  -0.076539   
1                   0       0     21       2.0   45.048780  -0.076539   
2                   0       1     15       0.0   -0.118467  -0.076539   
3                   0       1     11       1.0   -0.188153  -0.051581   
4                   0       0     21       0.0    0.742160  24.074875   
```

In [31]:

```
data_train.describe().style.background_gradient(cmap='Blues').set_properti
```

Out[31]:

	land	logged_in	is_host_login	is_guest_login	level	duration	
count	22543.000000	22543.000000	22543.000000	22543.000000	22543.000000	22543.000000	22
mean	0.000311	0.442222	0.000488	0.028435	18.017833	218.868784	
std	0.017619	0.496661	0.022085	0.166214	4.270409	1407.207069	1
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	17.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	20.000000	0.000000	
75%	0.000000	1.000000	0.000000	0.000000	21.000000	0.000000	
max	1.000000	1.000000	1.000000	1.000000	21.000000	57715.000000	218

In [32]:

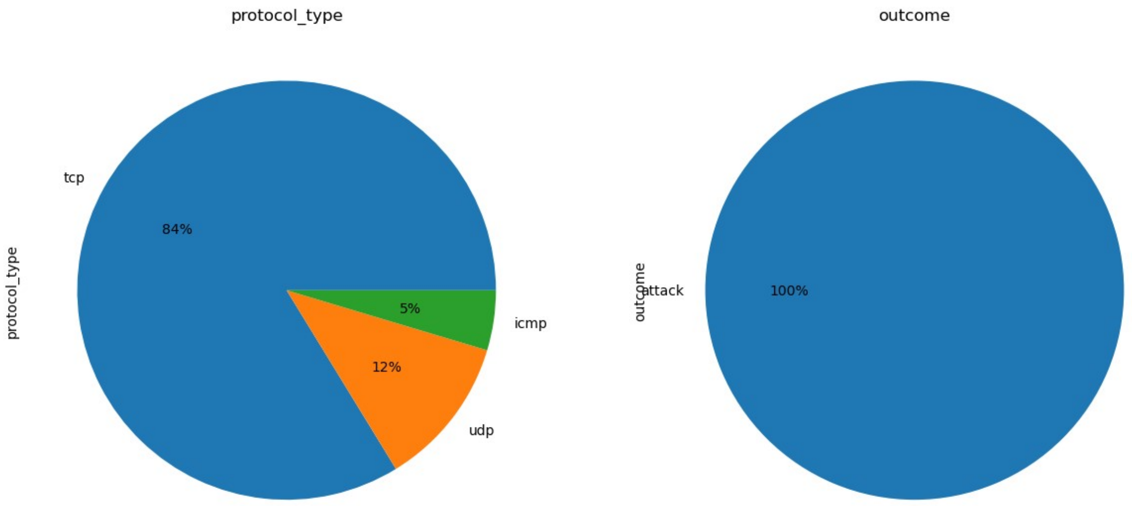
```
data_train.loc[data_train['outcome'] == "normal", "outcome"] = 'normal'
data_train.loc[data_train['outcome'] != 'normal', "outcome"] = 'attack'
```

In [33]:

```
def pie_plot(df, cols_list, rows, cols):
    fig, axes = plt.subplots(rows, cols)
    for ax, col in zip(axes.ravel(), cols_list):
        df[col].value_counts().plot(ax=ax, kind='pie', figsize=(15, 15), f
        ax.set_title(str(col), fontsize = 12)
    plt.show()
```

In [34]:

```
pie_plot(data_train, ['protocol_type', 'outcome'], 1, 2)
```



In [35]:

```
def Scaling(df_num, cols):
    std_scaler = RobustScaler()
    std_scaler_temp = std_scaler.fit_transform(df_num)
    std_df = pd.DataFrame(std_scaler_temp, columns =cols)
    return std_df
```

In [36]:

```
cat_cols = ['is_host_login','protocol_type','service','flag','land', 'logg
def preprocess(dataframe):
    df_num = dataframe.drop(cat_cols, axis=1)
    num_cols = df_num.columns
    scaled_df = Scaling(df_num, num_cols)

    dataframe.drop(labels=num_cols, axis="columns", inplace=True)
    dataframe[num_cols] = scaled_df[num_cols]

    dataframe.loc[dataframe['outcome'] == "normal", "outcome"] = 0
    dataframe.loc[dataframe['outcome'] != 0, "outcome"] = 1

    dataframe = pd.get_dummies(dataframe, columns = ['protocol_type', 'ser
    return dataframe
```

In [37]:

```
scaled_train = preprocess(data_train)
```

In [38]:

```
#Principal Component Analysis
x = scaled_train.drop(['outcome', 'level'] , axis = 1).values
y = scaled_train['outcome'].values
y_reg = scaled_train['level'].values

pca = PCA(n_components=20)
pca = pca.fit(x)
x_reduced = pca.transform(x)
print("Number of original features is {} and of reduced features is {}".fo

y = y.astype('int')
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, r
x_train_reduced, x_test_reduced, y_train_reduced, y_test_reduced = train_t
x_train_reg, x_test_reg, y_train_reg, y_test_reg = train_test_split(x, y_r
```

```
Number of original features is 116 and of reduced features is 20
```

In [39]:

```python
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classificati

# Function to evaluate a solution (subset of features) using a classifier
def evaluate_solution(features, X_train, X_test, y_train, y_test):
    clf = RandomForestClassifier(random_state=42)
    clf.fit(X_train[:, features], y_train)
    y_pred = clf.predict(X_test[:, features])
    return accuracy_score(y_test, y_pred)

# Bat Algorithm for Feature Selection
def bat_algorithm(X_train, X_test, y_train, y_test, num_bats, max_iter, A,
    num_features = X_train.shape[1]
    best_solution = np.zeros(num_features, dtype=bool)
    best_score = 0

    # Initialization
    bats = np.random.rand(num_bats, num_features) < 0.5
    velocities = np.zeros_like(bats, dtype=float)

    for _ in range(max_iter):
        # Update bat positions and velocities
        frequencies = np.zeros(num_bats)
        for i in range(num_bats):
            frequencies[i] = alpha * np.exp(-gamma * np.linalg.norm(np.log
            velocities[i] += np.logical_xor(bats[i], best_solution).astype
            bats[i] = np.logical_xor(bats[i], (np.random.rand(num_features

        # Evaluate solutions and update the best solution
        scores = np.array([evaluate_solution(bat, X_train, X_test, y_train
        best_bat = np.argmax(scores)

        if scores[best_bat] > best_score:
            best_score = scores[best_bat]
            best_solution = np.copy(bats[best_bat])

        # Update velocities and positions
        velocities += A * np.logical_xor(bats, best_solution).astype(float
        bats = np.logical_xor(bats, velocities > np.random.rand(num_bats,

    return best_solution

# Generate some dummy data for demonstration purposes
X, y = np.random.rand(100, 10), np.random.randint(0, 2, 100)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, r

# Adjust num_bats, max_iter, A, alpha, gamma as needed
best_features = bat_algorithm(X_train, X_test, y_train, y_test, num_bats=1
print("Best features selected:", np.where(best_features)[0])

# Continue with the model creation and evaluation code
# Extract the best features for training and testing
X_train_selected = X_train[:, best_features]
```

```python
X_test_selected = X_test[:, best_features]

# Train a RandomForestClassifier using the selected features
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train_selected, y_train)

# Make predictions on the test set
y_pred = clf.predict(X_test_selected)

# Evaluate the performance of the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_report_result = classification_report(y_test, y_pred)

# Print the results
print("Accuracy on the test set:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", classification_report_result)

```

```
Best features selected: [0 5 8]
Accuracy on the test set: 0.7
Confusion Matrix:
 [[9 2]
 [4 5]]
Classification Report:
               precision    recall  f1-score   support

           0       0.69      0.82      0.75        11
           1       0.71      0.56      0.63         9

    accuracy                           0.70        20
   macro avg       0.70      0.69      0.69        20
weighted avg       0.70      0.70      0.69        20
```

In [40]:

```python
# Print important information after running the Bat Algorithm
print("Best features selected:", np.where(best_features)[0])

# Evaluate the performance of the selected features on the test set
selected_features = np.where(best_features)[0]
X_test_selected = X_test[:, selected_features]

# Train a classifier on the selected features
clf_selected = RandomForestClassifier(random_state=42)
clf_selected.fit(X_train[:, selected_features], y_train)

# Make predictions on the test set
y_pred_selected = clf_selected.predict(X_test_selected)

# Calculate and print accuracy
accuracy_selected = accuracy_score(y_test, y_pred_selected)
print("Accuracy on the test set with selected features:", accuracy_selecte

# Print confusion matrix and classification report
from sklearn.metrics import confusion_matrix, classification_report

conf_matrix_selected = confusion_matrix(y_test, y_pred_selected)
classification_report_selected = classification_report(y_test, y_pred_sele

print("Confusion Matrix with Selected Features:\n", conf_matrix_selected)
print("Classification Report with Selected Features:\n", classification_re

```

```
Best features selected: [0 5 8]
Accuracy on the test set with selected features: 0.7
Confusion Matrix with Selected Features:
 [[9 2]
 [4 5]]
Classification Report with Selected Features:
               precision    recall  f1-score   support

           0       0.69      0.82      0.75        11
           1       0.71      0.56      0.63         9

    accuracy                           0.70        20
   macro avg       0.70      0.69      0.69        20
weighted avg       0.70      0.70      0.69        20
```

In [ ]:

```python

```