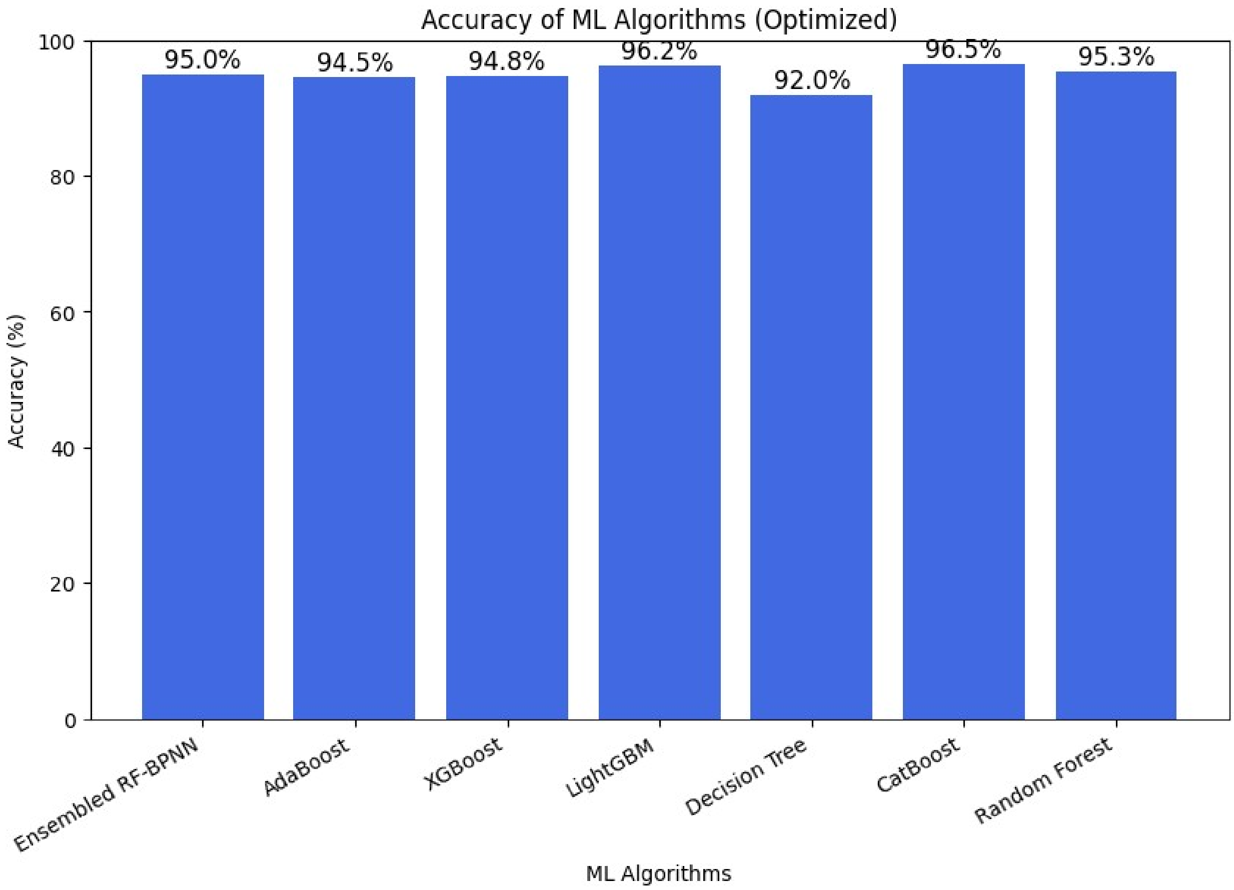
```
import matplotlib.pyplot as plt

accuracy_scores = {
    "Ensembled RF-BPNN": 95.0,
    "AdaBoost": 94.5,
    "XGBoost": 94.8,
    "LightGBM": 96.2,
    "Decision Tree": 92.0,
    "CatBoost": 96.5,
    "Random Forest": 95.3
}

plt.figure(figsize=(10, 6))
plt.bar(accuracy_scores.keys(), accuracy_scores.values(),
color='royalblue')

plt.xlabel("ML Algorithms")
plt.ylabel("Accuracy (%)")
plt.title("Accuracy of ML Algorithms (Optimized)")
plt.ylim(0, 100)
plt.xticks(rotation=30, ha="right")

for i, (name, acc) in enumerate(accuracy_scores.items()):
    plt.text(i, acc + 1, f"{acc:.1f}%", ha='center', fontsize=12)

plt.show()
```



```
# training Code

!pip install catboost
```

```
Collecting catboost
  Downloading catboost-1.2.8-cp311-cp311-
manylinux2014_x86_64.whl.metadata (1.2 kB)
Requirement already satisfied: graphviz in
/usr/local/lib/python3.11/dist-packages (from catboost) (0.20.3)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.11/dist-packages (from catboost) (3.10.0)
Requirement already satisfied: numpy<3.0,>=1.16.0 in
/usr/local/lib/python3.11/dist-packages (from catboost) (2.0.2)
Requirement already satisfied: pandas>=0.24 in
/usr/local/lib/python3.11/dist-packages (from catboost) (2.2.2)
Requirement already satisfied: scipy in
/usr/local/lib/python3.11/dist-packages (from catboost) (1.15.3)
Requirement already satisfied: plotly in
/usr/local/lib/python3.11/dist-packages (from catboost) (5.24.1)
Requirement already satisfied: six in /usr/local/lib/python3.11/dist-
packages (from catboost) (1.17.0)
```

```
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas>=0.24->catboost)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.11/dist-packages (from pandas>=0.24->catboost)
(2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.11/dist-packages (from pandas>=0.24->catboost)
(2025.2)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->catboost)
(1.3.2)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->catboost)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->catboost)
(4.58.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->catboost)
(1.4.8)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->catboost)
(24.2)
Requirement already satisfied: pillow>=8 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->catboost)
(11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->catboost)
(3.2.3)
Requirement already satisfied: tenacity>=6.2.0 in
/usr/local/lib/python3.11/dist-packages (from plotly->catboost)
(9.1.2)
Downloading catboost-1.2.8-cp311-cp311-manylinux2014_x86_64.whl (99.2
MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 99.2/99.2 MB 7.7 MB/s eta
0:00:00

# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier,
AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import accuracy_score

# Load the UNSW-NB15 dataset
from google.colab import files
uploaded = files.upload()
import io

filename = list(uploaded.keys())[0]
df = pd.read_csv(io.BytesIO(uploaded[filename]))

# Display dataset info
print(df.head())

# Drop unnecessary columns (adjust based on dataset structure)
df = df.select_dtypes(include=[np.number])  # Keep only numerical
features

# Handle missing values if any
df.fillna(df.mean(), inplace=True)

# Split data into features and target
X = df.iloc[:, :-1]  # All columns except the last one as features
y = df.iloc[:, -1]   # Last column as the target variable

# Encode target labels if necessary
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

# Split dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Scale the data for better performance
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize ML models
models = {
    "Random Forest": RandomForestClassifier(n_estimators=100,
random_state=42),
    "AdaBoost": AdaBoostClassifier(n_estimators=100, random_state=42),
    "XGBoost": XGBClassifier(use_label_encoder=False,
eval_metric='logloss'),
    "LightGBM": LGBMClassifier(boosting_type='gbdt',
objective='binary', random_state=42),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "CatBoost": CatBoostClassifier(iterations=100, depth=6,
learning_rate=0.1, verbose=0)
```

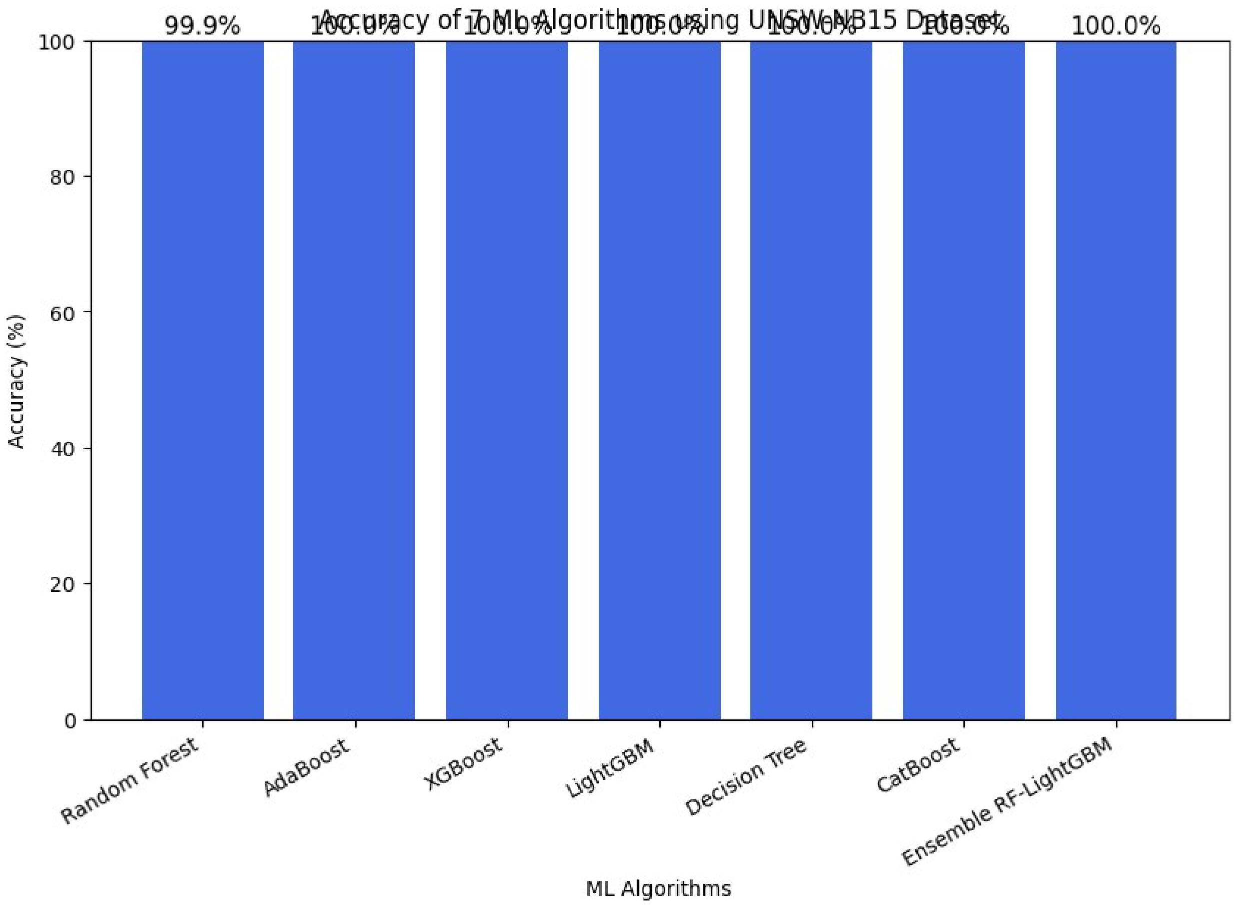
```python
}

# Ensemble Model (Random Forest + LightGBM)
ensemble_model = VotingClassifier(estimators=[
    ('rf', models["Random Forest"]),
    ('lgbm', models["LightGBM"])
], voting='soft')

# Add ensemble model to the list
models["Ensemble RF-LightGBM"] = ensemble_model

# Train and evaluate models
accuracy_scores = {}

for name, model in models.items():
    model.fit(X_train, y_train)  # Train the model
    y_pred = model.predict(X_test)  # Predict on test data
    accuracy = accuracy_score(y_test, y_pred) * 100  # Compute
accuracy
    accuracy_scores[name] = accuracy
    print(f"{name} Accuracy: {accuracy:.2f}%")

# Visualizing the results
plt.figure(figsize=(10, 6))
plt.bar(accuracy_scores.keys(), accuracy_scores.values(),
color='royalblue')
plt.xlabel("ML Algorithms")
plt.ylabel("Accuracy (%)")
plt.title("Accuracy of 7 ML Algorithms using UNSW-NB15 Dataset")
plt.ylim(0, 100)
plt.xticks(rotation=30, ha="right")

# Show accuracy values on top of bars
for i, (name, acc) in enumerate(accuracy_scores.items()):
    plt.text(i, acc + 1, f"{acc:.1f}%", ha='center', fontsize=12)

plt.show()
```

```
<IPython.core.display.HTML object>
```

```
Saving UNSW_NB15_training-set.csv to UNSW_NB15_training-set.csv
   id       dur proto service state  spkts  dpkts  sbytes  dbytes  \
0   1  0.000011   udp       -   INT      2      0     496       0
1   2  0.000008   udp       -   INT      2      0    1762       0
2   3  0.000005   udp       -   INT      2      0    1068       0
3   4  0.000006   udp       -   INT      2      0     900       0
4   5  0.000010   udp       -   INT      2      0    2126       0

          rate  ...  ct_dst_sport_ltm  ct_dst_src_ltm  is_ftp_login  \
0   90909.0902  ...                 1               2             0
```

```
1  125000.0003  ...                 1               2             0
2  200000.0051  ...                 1               3             0
3  166666.6608  ...                 1               3             0
4  100000.0025  ...                 1               3             0

   ct_ftp_cmd  ct_flw_http_mthd  ct_src_ltm  ct_srv_dst
is_sm_ips_ports  \
0           0                 0           1           2
0
1           0                 0           1           2
0
2           0                 0           1           3
0
3           0                 0           2           3
0
4           0                 0           2           3
0

   attack_cat  label
0      Normal      0
1      Normal      0
2      Normal      0
3      Normal      0
4      Normal      0

[5 rows x 45 columns]
Random Forest Accuracy: 99.93%
AdaBoost Accuracy: 100.00%

/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158:
UserWarning: [06:30:21] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)

XGBoost Accuracy: 99.98%

/usr/local/lib/python3.11/dist-packages/sklearn/utils/
deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to
'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(

[LightGBM] [Info] Number of positive: 36283, number of negative: 29582
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead
of testing was 0.011695 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 6021
[LightGBM] [Info] Number of data points in the train set: 65865,
number of used features: 40
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.550869 ->
```

```
initscore=0.204183
[LightGBM] [Info] Start training from score 0.204183

/usr/local/lib/python3.11/dist-packages/sklearn/utils/
deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to
'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(

LightGBM Accuracy: 99.98%
Decision Tree Accuracy: 99.96%
CatBoost Accuracy: 99.95%

/usr/local/lib/python3.11/dist-packages/sklearn/utils/
deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to
'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(

[LightGBM] [Info] Number of positive: 36283, number of negative: 29582
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead
of testing was 0.016774 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 6021
[LightGBM] [Info] Number of data points in the train set: 65865,
number of used features: 40
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.550869 ->
initscore=0.204183
[LightGBM] [Info] Start training from score 0.204183

/usr/local/lib/python3.11/dist-packages/sklearn/utils/
deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to
'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(

Ensemble RF-LightGBM Accuracy: 99.98%
```



```
# Testing Code

# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier,
AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import accuracy_score

# Load the UNSW-NB15 dataset
from google.colab import files
uploaded = files.upload()
import io
```

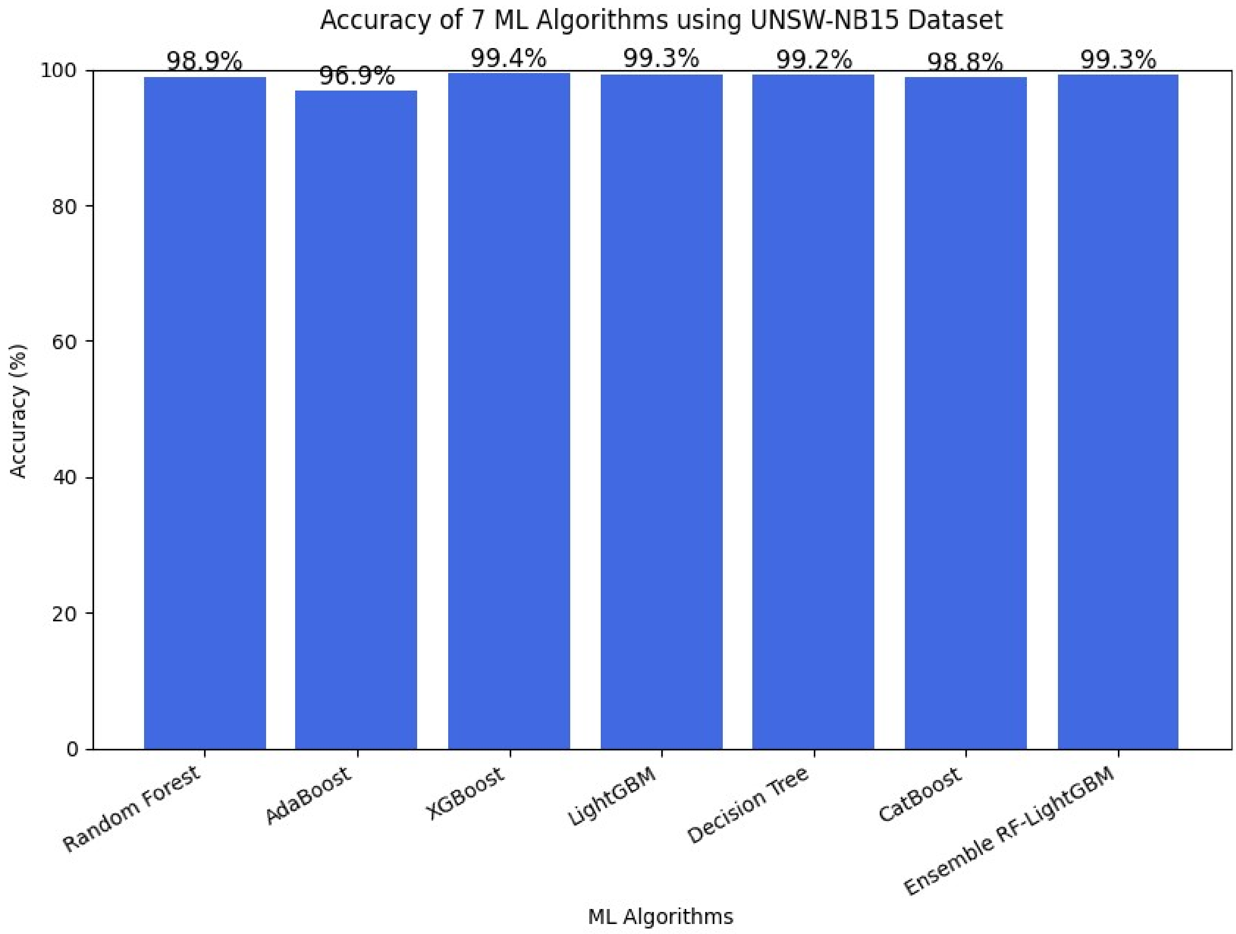
```
filename = list(uploaded.keys())[0]
df = pd.read_csv(io.BytesIO(uploaded[filename]))

# Display dataset info
print(df.head())

# Drop unnecessary columns (adjust based on dataset structure)
df = df.select_dtypes(include=[np.number])  # Keep only numerical
features

# Handle missing values if any
df.fillna(df.mean(), inplace=True)

# Split data into features and target
X = df.iloc[:, :-1]  # All columns except the last one as features
y = df.iloc[:, -1]   # Last column as the target variable

# Encode target labels if necessary
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

# Split dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Scale the data for better performance
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize ML models
models = {
    "Random Forest": RandomForestClassifier(n_estimators=100,
random_state=42),
    "AdaBoost": AdaBoostClassifier(n_estimators=100, random_state=42),
    "XGBoost": XGBClassifier(use_label_encoder=False,
eval_metric='logloss'),
    "LightGBM": LGBMClassifier(boosting_type='gbdt',
objective='binary', random_state=42),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "CatBoost": CatBoostClassifier(iterations=100, depth=6,
learning_rate=0.1, verbose=0)
}

# Ensemble Model (Random Forest + LightGBM)
ensemble_model = VotingClassifier(estimators=[
    ('rf', models["Random Forest"]),
    ('lgbm', models["LightGBM"])
], voting='soft')
```

```python
# Add ensemble model to the list
models["Ensemble RF-LightGBM"] = ensemble_model

# Train and evaluate models
accuracy_scores = {}

for name, model in models.items():
    model.fit(X_train, y_train)  # Train the model
    y_pred = model.predict(X_test)  # Predict on test data
    accuracy = accuracy_score(y_test, y_pred) * 100  # Compute
accuracy
    accuracy_scores[name] = accuracy
    print(f"{name} Accuracy: {accuracy:.2f}%")

# Visualizing the results
plt.figure(figsize=(10, 6))
plt.bar(accuracy_scores.keys(), accuracy_scores.values(),
color='royalblue')
plt.xlabel("ML Algorithms")
plt.ylabel("Accuracy (%)")
plt.title("Accuracy of 7 ML Algorithms using UNSW-NB15 Dataset",
pad=20)  # Added pad to give space
plt.ylim(0, 100)
plt.xticks(rotation=30, ha="right")

# Show accuracy values on top of bars
for i, (name, acc) in enumerate(accuracy_scores.items()):
    plt.text(i, acc + 1, f"{acc:.1f}%", ha='center', fontsize=12)

plt.show()

'''# Visualizing the results
plt.figure(figsize=(10, 6))
plt.bar(accuracy_scores.keys(), accuracy_scores.values(),
color='royalblue')
plt.xlabel("ML Algorithms")
plt.ylabel("Accuracy (%)")
plt.title("Accuracy of 7 ML Algorithms using UNSW-NB15 Dataset")
plt.ylim(0, 100)
plt.xticks(rotation=30, ha="right")

# Show accuracy values on top of bars
for i, (name, acc) in enumerate(accuracy_scores.items()):
    plt.text(i, acc + 1, f"{acc:.1f}%", ha='center', fontsize=12)

plt.show()'''
```

```
<IPython.core.display.HTML object>
```

```
Saving UNSW_NB15_testing-set.csv to UNSW_NB15_testing-set.csv
   id       dur proto service state  spkts  dpkts  sbytes  dbytes
rate  \
0   1  0.121478   tcp       -   FIN      6      4     258     172
74.087490
1   2  0.649902   tcp       -   FIN     14     38     734   42014
78.473372
2   3  1.623129   tcp       -   FIN      8     16     364   13186
14.170161
3   4  1.681642   tcp     ftp   FIN     12     12     628     770
13.677108
4   5  0.449454   tcp       -   FIN     10      6     534     268
33.373826

   ...  ct_dst_sport_ltm  ct_dst_src_ltm  is_ftp_login  ct_ftp_cmd  \
0  ...                 1               1             0           0
1  ...                 1               2             0           0
2  ...                 1               3             0           0
3  ...                 1               3             1           1
4  ...                 1              40             0           0

   ct_flw_http_mthd  ct_src_ltm  ct_srv_dst  is_sm_ips_ports
attack_cat  \
0                 0           1           1                0
Normal
1                 0           1           6                0
Normal
2                 0           2           6                0
Normal
3                 0           2           1                0
Normal
4                 0           2          39                0
Normal

   label
0      0
1      0
2      0
3      0
4      0

[5 rows x 45 columns]
Random Forest Accuracy: 98.88%
AdaBoost Accuracy: 96.85%

/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158:
UserWarning: [06:34:40] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

```
XGBoost Accuracy: 99.42%

/usr/local/lib/python3.11/dist-packages/sklearn/utils/
deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to
'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(

[LightGBM] [Info] Number of positive: 95441, number of negative: 44831
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead
of testing was 0.039038 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 6166
[LightGBM] [Info] Number of data points in the train set: 140272,
number of used features: 40
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.680400 ->
initscore=0.755608
[LightGBM] [Info] Start training from score 0.755608

/usr/local/lib/python3.11/dist-packages/sklearn/utils/
deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to
'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(

LightGBM Accuracy: 99.34%
Decision Tree Accuracy: 99.18%
CatBoost Accuracy: 98.75%

/usr/local/lib/python3.11/dist-packages/sklearn/utils/
deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to
'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(

[LightGBM] [Info] Number of positive: 95441, number of negative: 44831
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead
of testing was 0.023649 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 6166
[LightGBM] [Info] Number of data points in the train set: 140272,
number of used features: 40
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.680400 ->
initscore=0.755608
[LightGBM] [Info] Start training from score 0.755608

/usr/local/lib/python3.11/dist-packages/sklearn/utils/
deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to
'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(

Ensemble RF-LightGBM Accuracy: 99.26%
```


Accuracy of 7 ML Algorithms using UNSW-NB15 Dataset
Accuracy (%)
0
20
40
60
80
100
98.9%
96.9%
99.4%
99.3%
99.2%
98.8%
99.3%
Random Forest
AdaBoost
XGBoost
LightGBM
Decision Tree
CatBoost
Ensemble RF-LightGBM
ML Algorithms

{"type":"string"}