FINAL PROJECT

SOURCE CODE:

```
import pandas as pd
import numpy as np
import gzip
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Load the dataset from corrected.gz
with gzip.open('corrected.gz', 'rt') as f:
    df = pd.read_csv(f, header=None)

# Step 2: Assign column names
columns = [
    "duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes",
    "land", "wrong_fragment", "urgent", "hot", "num_failed_logins",
    "logged_in", "num_compromised", "root_shell", "su_attempted", "num_root",
    "num_file_creations", "num_shells", "num_access_files", "num_outbound_cmds",
    "is_host_login", "is_guest_login", "count", "srv_count", "serror_rate",
    "srv_serror_rate", "rerror_rate", "srv_rerror_rate", "same_srv_rate",
    "diff_srv_rate", "srv_diff_host_rate", "dst_host_count",
    "dst_host_srv_count", "dst_host_same_srv_rate", "dst_host_diff_srv_rate",
    "dst_host_same_src_port_rate", "dst_host_srv_diff_host_rate",
    "dst_host_serror_rate", "dst_host_srv_serror_rate", "dst_host_rerror_rate",
    "dst_host_srv_rerror_rate", "target"
```

```
]
df.columns = columns

# Step 3: Convert target to binary (0 = normal, 1 = attack)
df['target'] = df['target'].apply(lambda x: 0 if x == 'normal.' else 1)

# Step 4: One-hot encode categorical columns
df = pd.get_dummies(df, columns=["protocol_type", "service", "flag"])

# Step 5: Feature scaling
X = df.drop("target", axis=1)
y = df["target"]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 6: Train Isolation Forest
model = IsolationForest(n_estimators=100, contamination=0.1, random_state=42)
model.fit(X_scaled)
y_pred = model.predict(X_scaled)
y_pred = np.where(y_pred == 1, 0, 1)  # Convert: 1 → 0 (normal), -1 → 1 (anomaly)

# Step 7: Evaluate
print("Confusion Matrix:\n", confusion_matrix(y, y_pred))
print("\nClassification Report:\n", classification_report(y, y_pred))

# Step 8: Plot confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(confusion_matrix(y, y_pred), annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
```

plt.title("Confusion Matrix")

plt.tight_layout()

plt.show()

IMPLEMENTATION:

```
[1] from google.colab import files
    files.upload()
```

```
Choose Files kaggle.json
• kaggle.json(application/json) - 65 bytes, last modified: 6/20/2025 - 100% done
Saving kaggle.json to kaggle.json
{'kaggle.json': b'{"username":"alexa0990","key":"a80a469d7793d26d86d5b83bd4baa4c6"}'}
```

```
[3] !mkdir -p ~/.kaggle
    !cp kaggle.json ~/.kaggle/
    !chmod 600 ~/.kaggle/kaggle.json
```

```
[4] !kaggle datasets download -d galaxyh/kdd-cup-1999-data
    !unzip kdd-cup-1999-data.zip
```

```
Dataset URL: https://www.kaggle.com/datasets/galaxyh/kdd-cup-1999-data
License(s): unknown
Downloading kdd-cup-1999-data.zip to /content
  0% 0.00/87.8M [00:00<?, ?B/s]
100% 87.8M/87.8M [00:00<00:00, 1.01GB/s]
Archive:  kdd-cup-1999-data.zip
  inflating: corrected.gz
  inflating: corrected/corrected
  inflating: kddcup.data.corrected
  inflating: kddcup.data.gz
  inflating: kddcup.data/kddcup.data
  inflating: kddcup.data_10_percent.gz
  inflating: kddcup.data_10_percent/kddcup.data_10_percent
  inflating: kddcup.data_10_percent_corrected
  inflating: kddcup.names
  inflating: kddcup.newtestdata_10_percent_unlabeled.gz
  inflating: kddcup.newtestdata_10_percent_unlabeled/kddcup.newtestdata_10_percent_unlabeled
  inflating: kddcup.testdata.unlabeled.gz
  inflating: kddcup.testdata.unlabeled/kddcup.testdata.unlabeled
  inflating: kddcup.testdata.unlabeled_10_percent.gz
  inflating: kddcup.testdata.unlabeled_10_percent/kddcup.testdata.unlabeled_10_percent
  inflating: training_attack_types
```

OUTPUT:

```
Confusion Matrix:
 [[ 53681   6912]
 [226245  24191]]

Classification Report:
              precision    recall  f1-score   support

           0       0.19      0.89      0.32     60593
           1       0.78      0.10      0.17    250436

    accuracy                           0.25    311029
   macro avg       0.48      0.49      0.24    311029
weighted avg       0.66      0.25      0.20    311029
```