```
import pandas as pd
df = pd.read_csv('cybersecurity_intrusion_data.csv')
```

```
df.head()
```

	network_packet_size	login_attempts	session_duration	ip_reputation_score	failed_logins	unusual_time_access	attack_detected	prot
0	599	4	492.983263	0.606818	1	0	1	
1	472	3	1557.996461	0.301569	0	0	0	
2	629	3	75.044262	0.739164	2	0	1	
3	804	4	601.248835	0.123267	0	0	1	
4	453	5	532.540888	0.054874	1	0	0	

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9537 entries, 0 to 9536
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   network_packet_size   9537 non-null   int64
 1   login_attempts        9537 non-null   int64
 2   session_duration      9537 non-null   float64
 3   ip_reputation_score   9537 non-null   float64
 4   failed_logins         9537 non-null   int64
 5   unusual_time_access   9537 non-null   int64
 6   attack_detected       9537 non-null   int64
 7   protocol_type_TCP     9537 non-null   bool
 8   protocol_type_UDP     9537 non-null   bool
 9   encryption_used_DES   9537 non-null   bool
 10  browser_type_Edge     9537 non-null   bool
 11  browser_type_Firefox  9537 non-null   bool
 12  browser_type_Safari   9537 non-null   bool
 13  browser_type_Unknown  9537 non-null   bool
dtypes: bool(7), float64(2), int64(5)
memory usage: 586.9 KB
```

```
df.describe()
```

	network_packet_size	login_attempts	session_duration	ip_reputation_score	failed_logins	unusual_time_access	attack_detected
count	9537.000000	9537.000000	9537.000000	9537.000000	9537.000000	9537.000000	9537.000000
mean	500.430639	4.032086	792.745312	0.331338	1.517773	0.149942	0.447101
std	198.379364	1.963012	786.560144	0.177175	1.033988	0.357034	0.497220
min	64.000000	1.000000	0.500000	0.002497	0.000000	0.000000	0.000000
25%	365.000000	3.000000	231.953006	0.191946	1.000000	0.000000	0.000000
50%	499.000000	4.000000	556.277457	0.314778	1.000000	0.000000	0.000000
75%	635.000000	5.000000	1105.380602	0.453388	2.000000	0.000000	1.000000
max	1285.000000	13.000000	7190.392213	0.924299	5.000000	1.000000	1.000000

```
df.shape
```

```
(9537, 14)
```

```
df = pd.get_dummies(df, drop_first=True)
```

```
# Check for missing values
print(df.isnull().sum())
# If any:
df.fillna(df.mean(), inplace=True)
```

```
network_packet_size     0
login_attempts          0
session_duration        0
ip_reputation_score     0
failed_logins           0
unusual_time_access     0
attack_detected         0
protocol_type_TCP       0
protocol_type_UDP       0
encryption_used_DES     0
browser_type_Edge       0
browser_type_Firefox    0
browser_type_Safari     0
browser_type_Unknown    0
dtype: int64
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
X = df.drop('attack_detected', axis=1)
y = df['attack_detected']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

```
# Logistic Regression
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
lr_pred = lr.predict(X_test)
```

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
rf_pred = rf.predict(X_test)
```

```
# SVM
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
```

```
# KNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
knn_pred = knn.predict(X_test)
```

```
# Decision Tree
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
dt_pred = dt.predict(X_test)
```

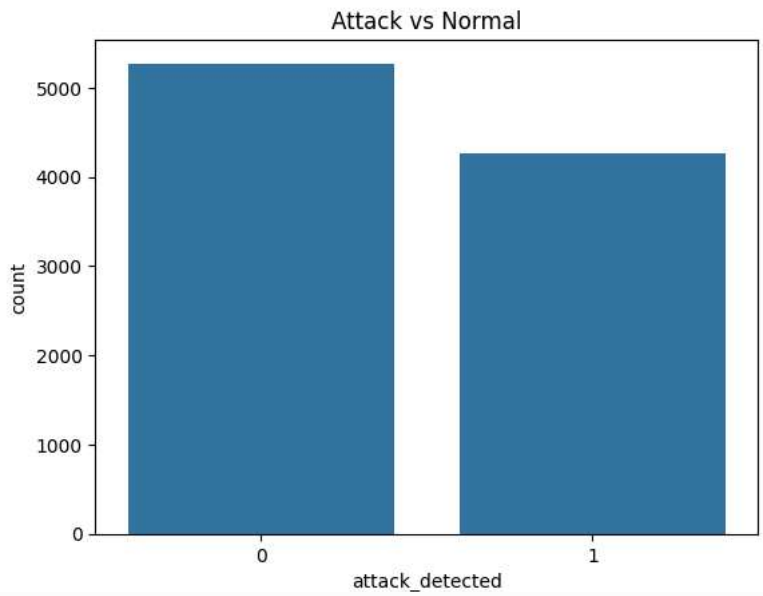
```
from sklearn.metrics import accuracy_score, precision_score
print('Logistic Regression:', accuracy_score(y_test, lr_pred), precision_score(y_test, lr_pred))
print('Random Forest:', accuracy_score(y_test, rf_pred), precision_score(y_test, rf_pred))
print('SVM:', accuracy_score(y_test, svm_pred), precision_score(y_test, svm_pred))
print('KNN:', accuracy_score(y_test, knn_pred), precision_score(y_test, knn_pred))
print('Decision Tree:', accuracy_score(y_test, dt_pred), precision_score(y_test, dt_pred))
```

```
Logistic Regression: 0.7468553459119497 0.7477360931435963
Random Forest: 0.8930817610062893 0.9896449704142012
SVM: 0.8721174004192872 0.9380281690140845
KNN: 0.7971698113207547 0.8537666174298375
Decision Tree: 0.8265199161425576 0.8050171037628279
```

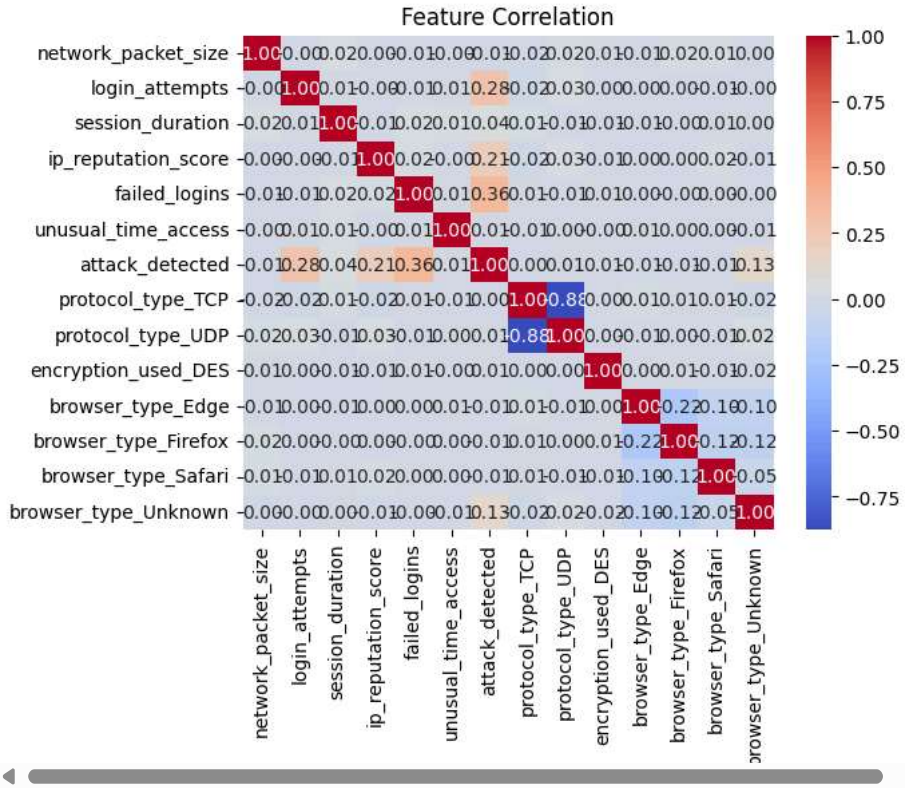
```
sample = X_test[5].reshape(1, -1)
print('Prediction:', rf.predict(sample))
```

```
Prediction: [0]
```

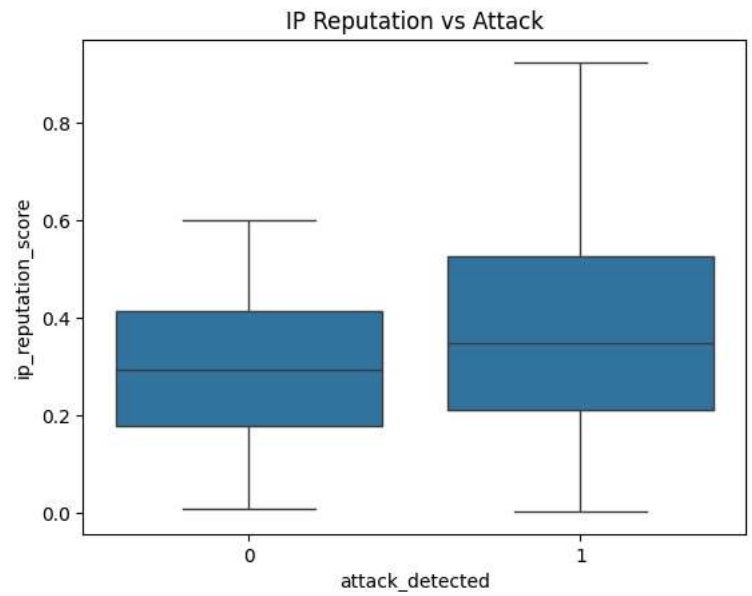
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(x='attack_detected', data=df)
plt.title('Attack vs Normal')
plt.show()
```



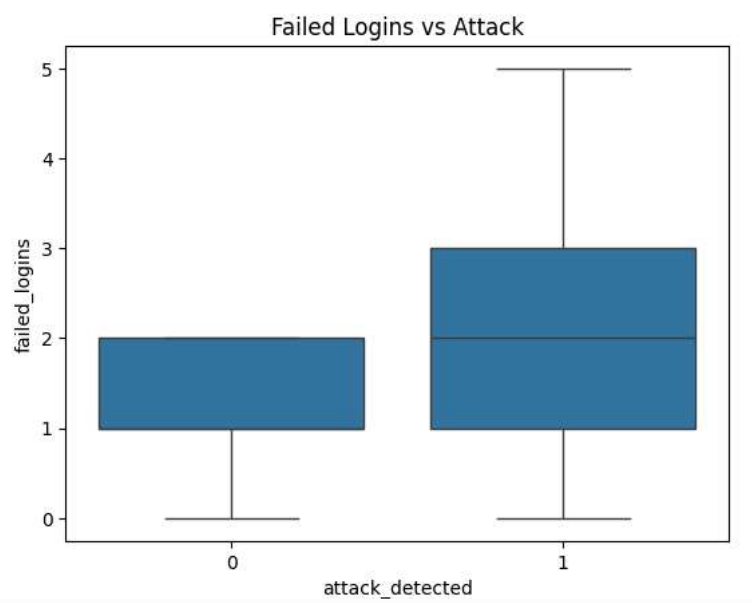

```
sns.heatmap(df.corr(), annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Feature Correlation')
plt.show()
```



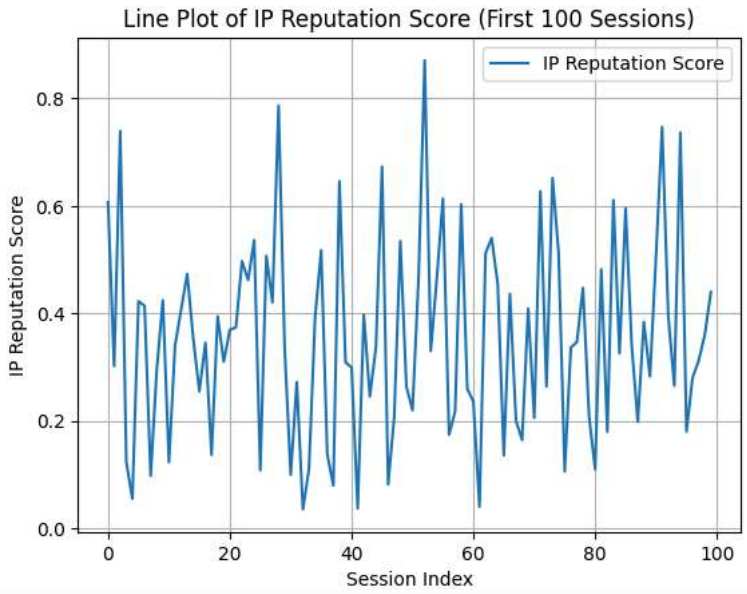

```
sns.boxplot(x='attack_detected', y='ip_reputation_score', data=df)
plt.title('IP Reputation vs Attack')
plt.show()
```




```
sns.boxplot(x='attack_detected', y='failed_logins', data=df)
plt.title('Failed Logins vs Attack')
plt.show()
```

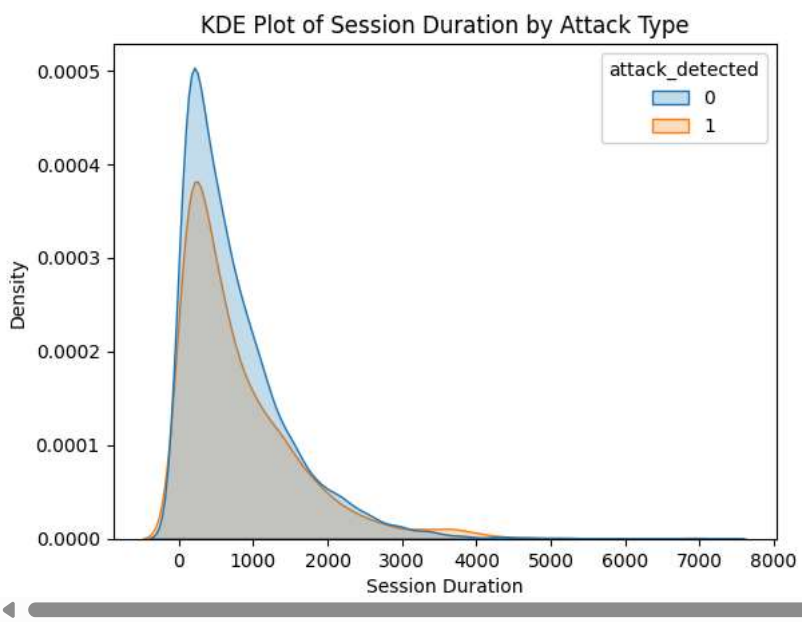



```
# Line plot for IP reputation scores of first 100 sessions
plt.plot(df['ip_reputation_score'][:100], label='IP Reputation Score')
plt.title('Line Plot of IP Reputation Score (First 100 Sessions)')
plt.xlabel('Session Index')
plt.ylabel('IP Reputation Score')
plt.legend()
plt.grid(True)
plt.show()
```

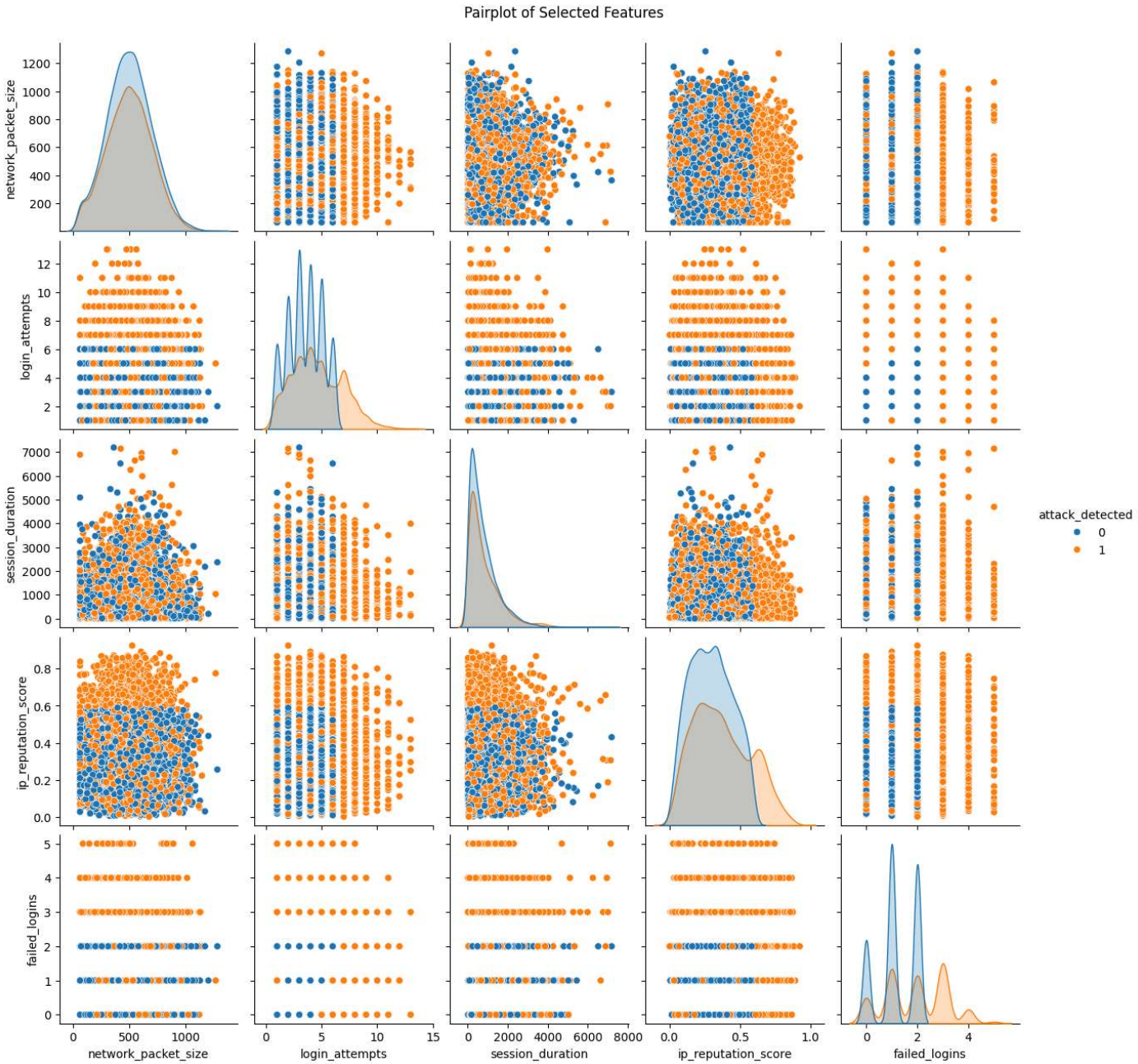



```
import seaborn as sns
import matplotlib.pyplot as plt

# KDE plot for session duration by attack class
sns.kdeplot(data=df, x='session_duration', hue='attack_detected', fill=True)
plt.title('KDE Plot of Session Duration by Attack Type')
plt.xlabel('Session Duration')
plt.ylabel('Density')
plt.show()
```

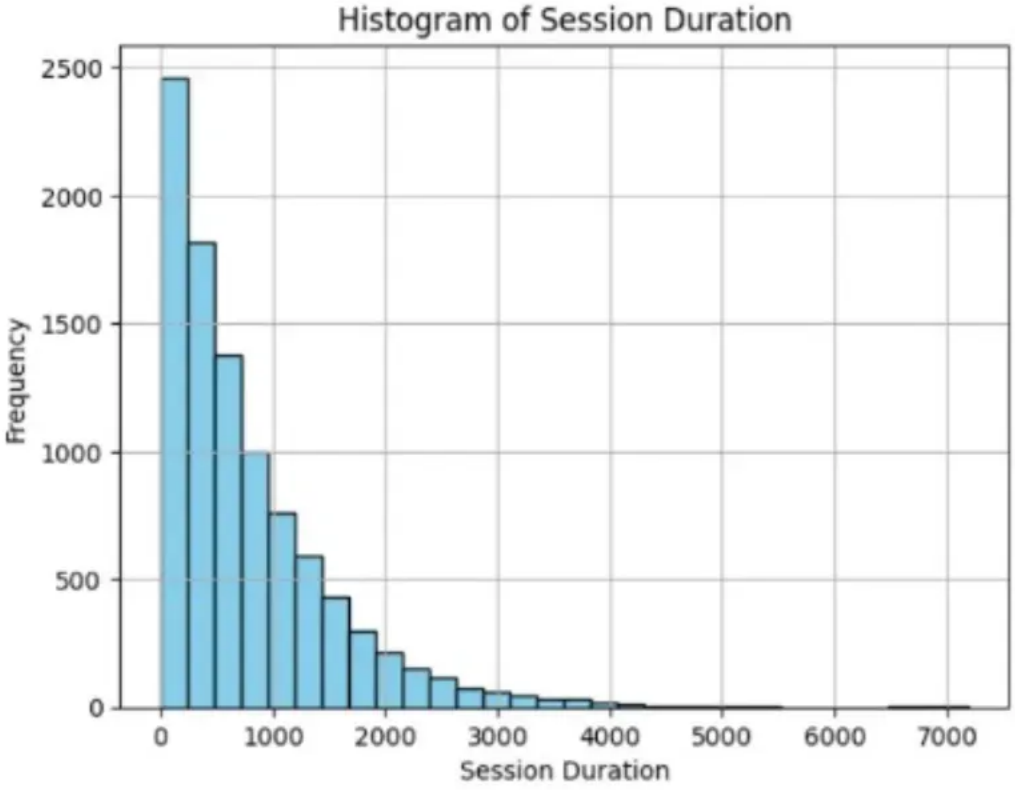



```
sns.pairplot(df[['network_packet_size', 'login_attempts', 'session_duration',
                 'ip_reputation_score', 'failed_logins', 'attack_detected']],
             hue='attack_detected')
plt.suptitle('Pairplot of Selected Features', y=1.02)
plt.show()
```




```
import matplotlib.pyplot as plt

plt.hist(df['session_duration'], bins=30, color='skyblue', edgecolor='black')
plt.title('Histogram of Session Duration')
plt.xlabel('Session Duration')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```


Histogram of Session Duration
Frequency
Session Duration
0
500
1000
1500
2000
2500
0
1000
2000
3000
4000
5000
6000
7000