spoof_detect

May 16, 2025

[152]:
```
import numpy as np
import pandas as pd
from geopy.distance import geodesic
import matplotlib.pyplot as plt
```

[153]:
```
def generate_gps_data(n_points=100, spoof_point=70, spoof_distance=0.01):
    lat, lon = 12.9716, 77.5946  # Starting location
    gps_data = []
    for i in range(n_points):
        if i == spoof_point:
            lat += spoof_distance  # Simulate spoofing with a big jump
            lon += spoof_distance
            true_spoofed = 1
        else:
            lat += np.random.uniform(-0.001, 0.001)
            lon += np.random.uniform(-0.001, 0.001)
            true_spoofed = 0
        gps_data.append((lat, lon, true_spoofed))
    print(gps_data)
    return pd.DataFrame(gps_data, columns=["latitude", "longitude",␣
↪"true_spoofed"])
```

[154]:
```
gps_df = generate_gps_data()
```

```
[(12.972580454684794, 77.59378776174349, 0), (12.972355718516281,
77.59440038011087, 0), (12.971509031495332, 77.5939114513095, 0),
(12.9716383667206, 77.5936387085275, 0), (12.971613655122264, 77.59350806090016,
0), (12.971850674616016, 77.59261643058515, 0), (12.972643855924556,
77.59315438634682, 0), (12.973057248905688, 77.59320159372623, 0),
(12.973143484390187, 77.59309801969766, 0), (12.974105822443544,
77.5923292537293, 0), (12.975052785103196, 77.59331961593556, 0),
(12.974559945142122, 77.59232041782798, 0), (12.97421058331563,
77.5920882812875, 0), (12.974164013156111, 77.59307435652842, 0),
(12.975082144934504, 77.59377718953014, 0), (12.97486544108787,
77.5931536537532, 0), (12.974387242943148, 77.59398389186441, 0),
(12.975204369153218, 77.59402193006966, 0), (12.975045789346703,
77.59340597397332, 0), (12.974241718206626, 77.59356687451759, 0),
(12.973669137977017, 77.5929869710244, 0), (12.973765625756757,
```

77.59347846477311, 0), (12.97394238750122, 77.5935688836293, 0),
(12.974863483878089, 77.59407716348713, 0), (12.974559497559586,
77.59403957720264, 0), (12.973970111214504, 77.5932290171285, 0),
(12.973630140894846, 77.59282822317944, 0), (12.97373219907638,
77.59282912272822, 0), (12.973309136992697, 77.59308084148404, 0),
(12.97420683313353, 77.59400880529252, 0), (12.97455045749641,
77.59364246983618, 0), (12.975330315027811, 77.59342116069382, 0),
(12.975414251401919, 77.59427140031754, 0), (12.976060475975348,
77.59367923217071, 0), (12.975168090022532, 77.59363151376382, 0),
(12.975618788971488, 77.59352260661231, 0), (12.975909680991887,
77.59376547686153, 0), (12.975872832301741, 77.59338292306484, 0),
(12.97541867236852, 77.59378633163224, 0), (12.975559179606867, 77.592952402805,
0), (12.975611229673913, 77.59363621022, 0), (12.975343613485867,
77.59431394991, 0), (12.974377681830108, 77.59369687065029, 0),
(12.97351686354564, 77.59295630444063, 0), (12.973790825419028,
77.59357531175795, 0), (12.972809271706952, 77.59278558505906, 0),
(12.973016638197072, 77.59270238141976, 0), (12.972262498093832,
77.59177239394333, 0), (12.97273604174913, 77.59092038715642, 0),
(12.973244737411752, 77.59074432072832, 0), (12.97411707684583,
77.5898631018108, 0), (12.973602936371671, 77.58958839744555, 0),
(12.972782224414729, 77.59048744222802, 0), (12.972472656958992,
77.59008998697662, 0), (12.972429693068376, 77.58975880061435, 0),
(12.972367503162525, 77.5902204675336, 0), (12.972272665329909,
77.58960679239036, 0), (12.971599965475935, 77.5892304819557, 0),
(12.972445996975786, 77.590213816759, 0), (12.9734349849828, 77.5907528307887,
0), (12.974424241596745, 77.59140737209171, 0), (12.974962243570525,
77.59057035228727, 0), (12.975491716404168, 77.59042747242859, 0),
(12.97480161516678, 77.58959203952439, 0), (12.973857150976075,
77.59035722698165, 0), (12.974573008746198, 77.589639166815, 0),
(12.975529469260362, 77.58979446097929, 0), (12.976303775990818,
77.59024090418232, 0), (12.977123955346515, 77.58975699244614, 0),
(12.977150669679235, 77.58891354605119, 0), (12.987150669679234,
77.5989135460512, 1), (12.987690175310107, 77.59832680066214, 0),
(12.988102999072003, 77.59897038246811, 0), (12.987506678754945,
77.5986803016533, 0), (12.987249450833968, 77.5982044529633, 0),
(12.986471402642055, 77.59888642075751, 0), (12.98595474546518,
77.59873284109032, 0), (12.985295520985877, 77.59819029739326, 0),
(12.985056146922837, 77.5989167080133, 0), (12.985973214041076,
77.59847113324336, 0), (12.986783338875586, 77.59827198039395, 0),
(12.987258530749896, 77.59926618184382, 0), (12.987940282606123,
77.59925764431773, 0), (12.988113240828106, 77.59836030765294, 0),
(12.987741503037633, 77.5981330601593, 0), (12.988442619898475,
77.5982972880111, 0), (12.9881468615222, 77.59862507645579, 0),
(12.988313156266033, 77.5985404499447, 0), (12.989189452036385,
77.59841713095038, 0), (12.988896998815033, 77.59802791778661, 0),
(12.988125034196216, 77.59805700053509, 0), (12.98888322729946,
77.59793779219704, 0), (12.98927776863363, 77.59803231824962, 0),
(12.98939177782206, 77.59832662712967, 0), (12.989059348062398,

```
77.59855552117124, 0), (12.989028333620247, 77.5988412599887, 0),
(12.989387868205105, 77.59789282846555, 0), (12.989783385525264,
77.59752015138186, 0), (12.990694276519966, 77.59677958503036, 0),
(12.990257807370277, 77.59737397940061, 0)]
```

[155]:
```
gps_df.head()
```

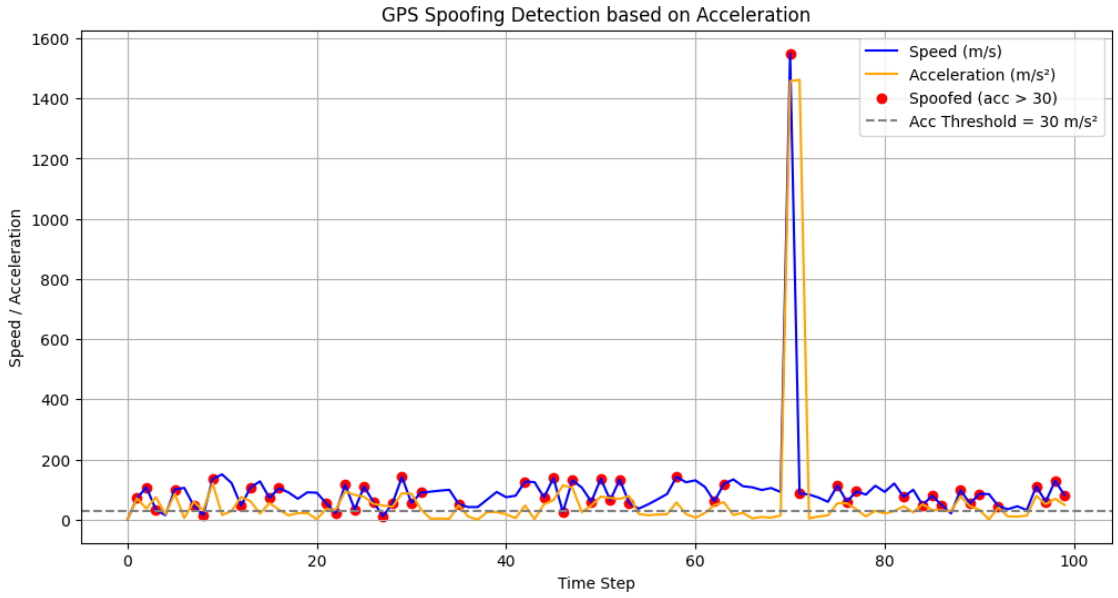
[155]:

	latitude	longitude	true_spoofed
0	12.972580	77.593788	0
1	12.972356	77.594400	0
2	12.971509	77.593911	0
3	12.971638	77.593639	0
4	12.971614	77.593508	0

[156]:
```
gps_df = compute_speed(gps_df)
gps_df = compute_acceleration(gps_df)
gps_df = detect_spoofing(gps_df, acc_threshold=30)
```

[157]:
```
plt.figure(figsize=(12, 6))
plt.plot(gps_df["speed_mps"], label="Speed (m/s)", color="blue")
plt.plot(gps_df["acceleration_mps2"], label="Acceleration (m/s²)",␣
 ↪color="orange")
plt.scatter(gps_df[gps_df["spoofed"] == 1].index,
            gps_df[gps_df["spoofed"] == 1]["speed_mps"],
            color="red", label="Spoofed (acc > 30)")

plt.axhline(30, color="gray", linestyle="--", label="Acc Threshold = 30 m/s²")
plt.xlabel("Time Step")
plt.ylabel("Speed / Acceleration")
plt.title("GPS Spoofing Detection based on Acceleration")
plt.legend()
plt.grid(True)
plt.show()
```



Effectiveness of Model

[158]:
```
from sklearn.metrics import classification_report, confusion_matrix

y_true = gps_df["true_spoofed"]
y_pred = gps_df["spoofed"]

print("Confusion Matrix:")
print(confusion_matrix(y_true, y_pred))

print("\nClassification Report:")
print(classification_report(y_true, y_pred, target_names=["Normal", "Spoofed"]))
```

```
Confusion Matrix:
[[47 52]
 [ 0  1]]

Classification Report:
              precision    recall  f1-score   support

      Normal       1.00      0.47      0.64        99
     Spoofed       0.02      1.00      0.04         1

    accuracy                           0.48       100
   macro avg       0.51      0.74      0.34       100
weighted avg       0.99      0.48      0.64       100
```