In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
train_df = pd.read_parquet("train.parquet")
test_df = pd.read_parquet("test.parquet")
sample_submission = pd.read_parquet("sample_submission.parquet")
```

In [3]:
```
train_df.head()
train_df.info()
train_df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1639424 entries, 0 to 1639423
Data columns (total 7 columns):
 #   Column  Non-Null Count    Dtype         
---  ------  --------------    -----         
 0   Date    1639424 non-null  datetime64[ns]
 1   X1      1639424 non-null  float64       
 2   X2      1639424 non-null  float64       
 3   X3      1639424 non-null  float64       
 4   X4      1639424 non-null  float64       
 5   X5      1639424 non-null  float64       
 6   target  1639424 non-null  object        
dtypes: datetime64[ns](1), float64(5), object(1)
memory usage: 87.6+ MB
```

Out[3]:

	Date	X1	X2	X3	X4	
count	1639424	1.639424e+06	1.639424e+06	1.639424e+06	1.639424e+06	1.6394
mean	2022-12-03 07:23:43.817145600	1.139258e+00	5.488189e+00	4.110388e+32	2.706323e+29	1.1872
min	2020-12-16 00:00:00	1.000000e+00	5.412539e+00	1.000000e+00	1.000000e+00	0.0000
25%	2021-12-10 00:00:00	1.049171e+00	5.480597e+00	1.000000e+00	1.000000e+00	0.0000
50%	2022-11-30 00:00:00	1.105171e+00	5.488979e+00	1.000000e+00	1.000000e+00	6.9314
75%	2023-11-23 00:00:00	1.214096e+00	5.496717e+00	1.000000e+00	2.718282e+00	2.8903
max	2024-12-11 00:00:00	4.014850e+00	5.541852e+00	1.651636e+38	5.540622e+34	3.4657
std	NaN	1.391992e-01	1.342811e-02	2.346156e+35	5.812988e+31	1.3048

In [4]:
```
train_df.isnull().sum()
```

Out[4]:
```
Date      0
X1        0
X2        0
X3        0
X4        0
X5        0
target    0
dtype: int64
```

In [5]:
```
sensor_cols = ["X1", "X2", "X3", "X4", "X5"]

imputer = SimpleImputer(strategy="median")
train_df[sensor_cols] = imputer.fit_transform(train_df[sensor_cols])
test_df[sensor_cols] = imputer.transform(test_df[sensor_cols])
```

In [6]:
```
for df in [train_df, test_df]:
    df["sensor_mean"] = df[sensor_cols].mean(axis=1)
    df["sensor_std"] = df[sensor_cols].std(axis=1)
    df["sensor_max"] = df[sensor_cols].max(axis=1)
    df["sensor_min"] = df[sensor_cols].min(axis=1)
```

In [7]:
```
X = train_df.drop("target", axis=1)
y = train_df["target"]

X_test = test_df.drop(columns=["ID"], errors="ignore")
```

In [8]:
```
# Feature engineering
for df in [train_df, test_df]:
```

```
    df["hour"] = df["Date"].dt.hour
    df["day"] = df["Date"].dt.day
    df["month"] = df["Date"].dt.month
    df["dayofweek"] = df["Date"].dt.dayofweek

# Drop Date AFTER feature extraction
train_df.drop(columns=["Date"], inplace=True)
test_df.drop(columns=["Date"], inplace=True)

# Split X and y
X = train_df.drop("target", axis=1)
y = train_df["target"]

X_test = test_df.drop(columns=["ID"], errors="ignore")

# Scale
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_test_scaled = scaler.transform(X_test)
```

In [9]:
```
X_train, X_val, y_train, y_val = train_test_split(
    X_scaled, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)
```

In [10]:
```
lr = LogisticRegression(max_iter=1000, n_jobs=-1)
lr.fit(X_train, y_train)
```

Out[10]:
```
LogisticRegression
LogisticRegression(max_iter=1000, n_jobs=-1)
```

In [11]:
```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
```

Out[11]:
```
KNeighborsClassifier
KNeighborsClassifier()
```

In [12]:
```
rf = RandomForestClassifier(
    n_estimators=300,
    max_depth=10,
    random_state=42,
    n_jobs=-1
)
rf.fit(X_train, y_train)
```

Out[12]:

RandomForestClassifier

```
RandomForestClassifier(max_depth=10, n_estimators=300, n_jobs=-1,
                       random_state=42)
```

In [13]:

```python
from sklearn.metrics import accuracy_score, f1_score

# Ensure labels are integers (safety check)
y_val = y_val.astype(int)

models = {
    "Logistic Regression": lr,
    "KNN": knn,
    "Random Forest": rf
}

for name, model in models.items():
    preds = model.predict(X_val)
    preds = preds.astype(int)  # safety conversion

    print(f"\n{name}")
    print("Accuracy:", accuracy_score(y_val, preds))
    print("F1 Score:", f1_score(y_val, preds, average="binary"))
```

```
Logistic Regression
Accuracy: 0.9916769599097245
F1 Score: 0.15116640746500778

KNN
Accuracy: 0.9938545526632814
F1 Score: 0.5450440279972907

Random Forest
Accuracy: 0.993445872790765
F1 Score: 0.3886201991465149
```

In [14]:

```python
test_preds = knn.predict(X_test_scaled)
```

In [15]:

```python
# Create submission with correct length
submission = pd.DataFrame({
    "ID": test_df["ID"],
    "target": test_preds.astype(int)
})

submission.to_csv("submission.csv", index=False)
```