DIGANTARA ASSIGNMENT

Autonomous Detection of Orbital Maneuvers using orbital data

Problem Statement

To create a method to autonomously detect Orbital maneuvers using orbital data, in our case Semi Major Axis Variation with respect to date and time and to explain the choice of methodology.

Methodology

Identifying orbital maneuvers resembles detecting anomalies, where maneuvers are treated as deviations detectable even with limited data. While machine learning provides high accuracy and adaptability in maneuver detection, heuristic methods offer simplicity and interpretability but may not cover all anomaly types effectively. Despite the accuracy of machine learning predictions, they can tend to overfit due to imbalanced data. Therefore, establishing statistical thresholds based on practical experience or theoretical insights remains essential for reliable maneuver detection.

CODE

This approach introduces "SMA_diff_abs" feature to measure significant changes in orbit, crucial for maneuver detection, with a threshold of 0.2 based on historical analysis. X features SMA_diff_abs for its direct insight into orbital dynamics, simplifying model interpretation, while y (Maneuver) guides the model in learning maneuver patterns. The Random Forest Regressor is chosen for its robustness against overfitting and ability to handle complex relationships, ensuring accurate detection. Avoiding sampling in imbalanced data prevents bias, ensuring all maneuver instances contribute to training for effective orbital anomaly detection.

1. **Data Preprocessing**: Computes SMA differences and absolute differences as features.
2. **Maneuver Detection**: Uses Random Forest Regressor to predict and classify SMA variations exceeding a threshold (0.2) as maneuvers.
3. **Model Evaluation**: Compares predicted maneuver dates against known maneuver dates to compute accuracy metrics.
4. **Visualization**: Plots SMA over time, marking detected maneuvers.

Assumptions:

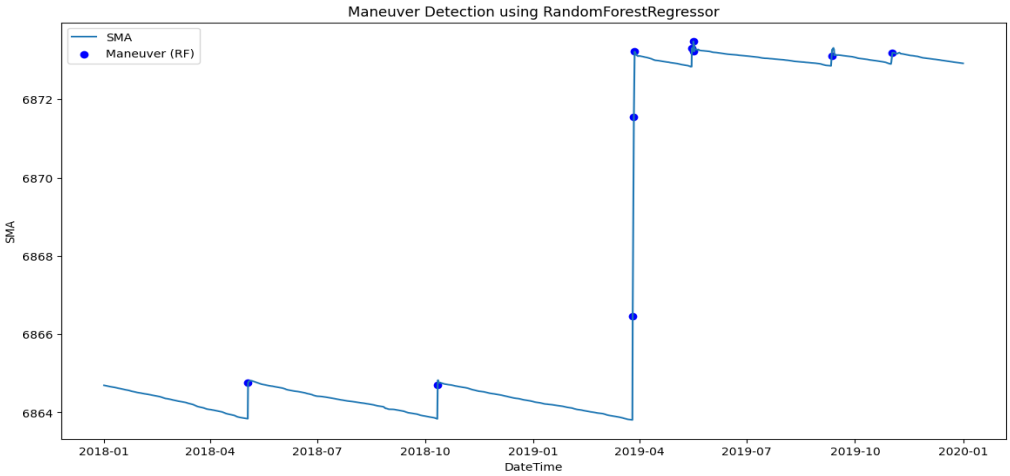
- **Thresholding**: Assumes maneuvers are identified by SMA differences exceeding 0.2.
- **Model Effectiveness**: Random Forest Regressor does its best to be efficient to learn and predict maneuvers from the limited SMA data which is also a severely imbalanced dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split

# Data preprocessing
url = 'https://drive.google.com/uc?id=1BvQRvWXdbpXpOaSahEMovATmKQuQilQa'
data = pd.read_csv(url, parse_dates=['Datetime'])
data['SMA_diff'] = data['SMA'].diff().fillna(0)
data['SMA_diff_abs'] = data['SMA_diff'].abs()

# Define labels for Maneuver detection (1 for Maneuver, 0 for normal)
data['Maneuver'] = np.where(data['SMA_diff_abs'] > 0.2, 1, 0) #Threshold for Maneuver = 0.2
```

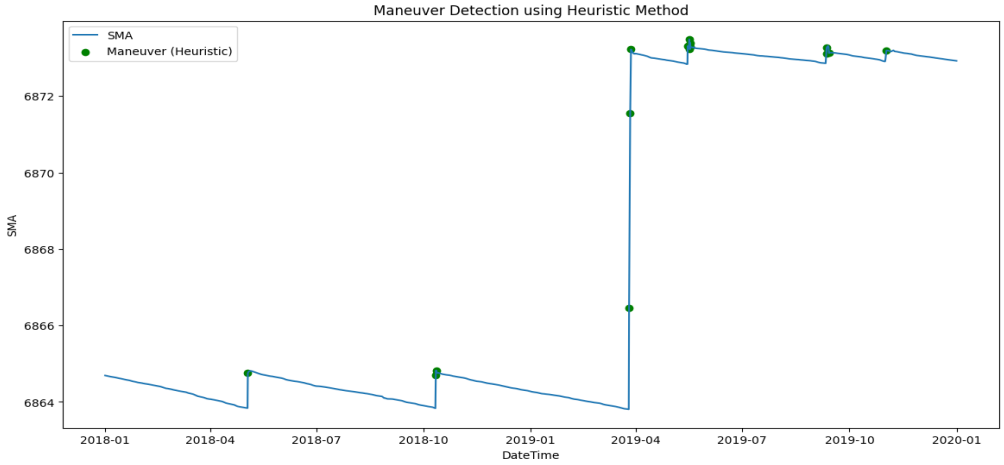
```
# Define features and target
X = data[['SMA_diff_abs']].values
y = data['Maneuver'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Random Forest regressor Model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X)

threshold = 0.2 # Convert Maneuver scores to binary classification (Maneuver or not)
y_pred_binary = np.where(y_pred >= threshold, 1, 0)
Maneuver_indices = np.where(y_pred_binary == 1)[0]
Maneuver_dates = data.iloc[Maneuver_indices]['Datetime']

# Known Maneuver dates and corresponding SMA data
known_Maneuver_dates = [ '2018-05-03', '2018-10-11', '2019-03-27', '2019-05-17', '2019-09-11', '2019-11-01']

# Check accuracy against known maneuver dates
predicted_Maneuver_dates = Maneuver_dates.dt.strftime('%Y-%m-%d').tolist()

true_positives = len(set(predicted_Maneuver_dates) & set(known_Maneuver_dates))
false_positives = len(set(predicted_Maneuver_dates) - set(known_Maneuver_dates))
false_negatives = len(set(known_Maneuver_dates) - set(predicted_Maneuver_dates))

accuracy = true_positives / len(known_Maneuver_dates) * 100

# Calculate precision, recall, and F1 score based on known anomaly dates
precision = true_positives / (true_positives + false_positives) if (true_positives + false_positives) > 0 else 0
recall = true_positives / (true_positives + false_negatives) if (true_positives + false_negatives) > 0 else 0
f1 = 2 * (precision * recall) / (precision + recall) if (precision + recall) > 0 else 0

# Print dates with detected maneuvers
print("Dates with Detected Maneuvers:")
print(Maneuver_dates)

# Plotting to visualize maneuvers detected by RandomForestRegressor
plt.figure(figsize=(14, 7))
plt.plot(data['Datetime'], data['SMA'], label='SMA')
plt.scatter(Maneuver_dates, data.iloc[Maneuver_indices]['SMA'], marker='o', color='blue', label='Maneuver (RF)')

plt.xlabel('DateTime')
plt.ylabel('SMA')
plt.legend()
plt.title('Maneuver Detection using RandomForestRegressor')
plt.show()

# Print accuracy metrics
print("\nAccuracy Evaluation:")
print(f"True Positives (Correctly Detected Maneuvers): {true_positives}")
print(f"False Positives (Incorrectly Detected Maneuvers): {false_positives}")
print(f"False Negatives (Missed Maneuvers): {false_negatives}")
print(f"Accuracy: {accuracy:.2f}%")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
```

Note : Due to the constraints in the space and as it contradicts my interest to use a ML model, I'm just sharing the link for the heuristic method - https://colab.research.google.com/drive/16aguMkhpSprcJnFqQUhOFqqXxJafXllo?usp=sharing

Result



Accuracy Evaluation: (ML Model)
True Positives (Correctly Detected Maneuvers): 6
False Positives (Incorrectly Detected Maneuvers): 3
False Negatives (Missed Maneuvers): 0
Accuracy: 100.00%
Precision: 0.67
Recall: 1.00
F1 Score: 0.80



Accuracy Evaluation: (Heuristic Method)
True Positives (Correctly Detected Maneuvers): 6
False Positives (Incorrectly Detected Maneuvers): 4
False Negatives (Missed Maneuvers): 0
Accuracy: 100.00%
Precision: 0.60
Recall: 1.00
F1 Score: 0.75

Dates with Detected Maneuvers (by Random Forest Regressor)
2018-05-03 12:01:31.056960
2018-10-11 13:37:04.556640
2019-03-26 04:53:33.243936
2019-03-27 04:34:36.436800
2019-03-27 20:25:37.599168
2019-05-15 10:44:36.864096
2019-05-16 16:43:16.023072

2019-05-17 05:11:39.372288
2019-09-11 04:28:22.151136
2019-11-01 17:55:22.162656

Conclusion:

Both the methods have performed great in terms of accuracy, while ML model looks robust and Heuristic method seems less complicated, it again comes down to the quality of the data and how we could use them to the fullest and achieve our objective. The constraints in data shall favor the Heuristic method, as it doesn't really need a complex model to detect the maneuvers, by introducing more statistical models, for example, rolling mean and std, the heuristic method shall also show some necessary robustness, but again limited by the fact that it shall be applicable only in cases like these, dealing with the fewer features, here in our case only the SMA variation over time. So, when it comes to dealing with multiple factors (features), like Period, Inclination, eccentricity or even any other perturbations, which we will be encountering in real time, we will be challenged with a data set that requires robustness, so, it always comes down to Machine Learning method as it is proven to achieve accuracy when handling anomalies. Thus, a well-tuned and trained Machine Learning model shall outperform a Heuristic method, in any general case

p.s- i also made a few attempts to code to classify the type of maneuvers made, but I was not sure if they are valid, thus not including them here, such classification shall turn out to be useful analytics.