COE305 MACHINE LEARNING

Final Project Report

Deadline: 04-01-2026

Team Members:

Student ID	Student Name	Section (as per OIS)
2309115620	Amro Akel	3
2309015120	Syed Husain Razavi	1
2309015473	Abdulrahman Bin Omran	3

Project Title: Satellite Collision Avoidance using machine learning models

Abstract:

The increasing congestion of Low Earth Orbit (LEO) has made collision avoidance a routine and resource-intensive task in satellite operations. Space agencies process hundreds of weekly conjunction alerts per satellite, delivered as Conjunction Data Messages (CDMs), of which only a small fraction require detailed human analysis and potential avoidance manoeuvres. Accurate prediction of collision risk is therefore critical to ensure spacecraft safety while minimizing unnecessary operational actions.

This project investigates the use of machine learning techniques to predict the final collision risk estimate between an operational satellite and a potentially hazardous space object using real-world CDM data provided by the European Space Agency. The dataset contains orbital parameters, uncertainty covariances, relative geometry, and temporal information describing close-approach events. After data preprocessing, feature engineering, and handling of missing or noisy measurements, multiple regression and ensemble-based models are trained to learn the relationship between CDM attributes and the final assessed collision probability.

Model performance is evaluated using appropriate regression metrics, with emphasis on accuracy, robustness, and generalization across different conjunction scenarios. The results demonstrate that machine learning models can effectively approximate expert-validated collision risk estimates and provide consistent predictions. Such models have the potential to support operational decision-making by prioritizing high-risk events, reducing analyst workload, and improving the efficiency of satellite collision avoidance processes.

Introduction

- Background of the problem

With the rise of “new space” and the deployment of massive satellite constellations, it is no longer an occasional task but a routine operational necessity. Currently space agencies rely on a stream of conjunction data messages to monitor close encounters. However the scale of data is shocking and overwhelming a single satellite can trigger hunderds of alerts weekly. even after automated filtering human analysts must manually intervene for approximately two actionable alerts per spacecraft every week, leading to frequent costly avoidance maneuvers.

- Motivation for using Machine Learning

The primary motivation for applying Machine Learning (ML) in this domain is the limitation of classical probabilistic modeling under high uncertainty. Traditional methods often require manual "fine-tuning" by experts to distinguish between false alarms and genuine threats, and as the number of satellites grows humans in the loop analysis cannot scale.

- Societal / practical relevance

The sustainability of the orbital environment is a global priority. A single high-speed collision can generate thousands of pieces of lethal debris a phenomenon known as the Kessler Syndrome which could render specific orbits unusable for generations.

Problem Statement

Clearly define:

- The prediction / classification task

  Today, active collision avoidance among orbiting satellites has become a routine task in space operations, For a typical satellite in Low Earth Orbit, hundreds of alerts issued every week corresponding to possible close encounters between a satellite and another space object (in the form of conjunction data messages, CDMs. As of estimations done in January 2019, more than 34,000 objects with a size larger than 10cm are orbiting in our planet. Of these, 22,300 are tracked by the Space Surveillance Network, and their position released in the form of a globally shared catalogue, which makes 11,400 objects not under surveillance or monitored resulting in a higher chance of collision between objects in space. making accurate and early maneuvers decisions for the avoidance of collisions lays on the uncertainty in the data.

- Why the problem is important?

  As we mentioned earlier, More than a dozen spacecraft of partner agencies and commercial operators require assistance, Unnecessary avoidance maneuvers which are resulted from data uncertainty consume resources therefore shortening the operational life of expensive satellites and increasing the cost plus inaccurate early warning pose a critical risk to spacecrafts leading to wasting resources or a catastrophic if a true high risk event is missed.

- Limitations of existing solutions

  On average, at the European Space Agency, more than one collision avoidance maneuvers is

performed per satellite. In this challenge, we have to build a model that makes use of the CDMs recorded up to 2 days prior to the closest approach to predict the final risk.

Objectives

1- To create a feature set that accurately shows the event state at the two-day decision deadline by preprocessing the time-series of Conjunction Data Messages (CDMs) for each unique close approach event.

2- To build and train a regression model that can accurately predict the final collision risk for each event (Log10(PoC)).

3- To make sure that the model's predictions are a reliable early warning, the amount of prediction error for the most dangerous events must be kept to a minimum.

4- To find out which CDM attributes and/or engineered features have the biggest effect on the final risk prediction.

Dataset Description

- **Dataset Source:** Collision Avoidance Challenge
- **Data Type:** (Structured / Image / Text / Time-series): Structured
- **Number of Samples:** 162634
- **Number of Features:**104
- **Target Variable:** risk
- **Link for the Dataset:** Collision Avoidance Challenge

DATA PREPROCESSING & EDA

Data Pre-processing

- Handling Missing Values:-

Identify Missing Values

Description: To identify missing values, We will calculate the count and percentage of null values for each column, and then create a dataframe to display it and then sort it.

The code:

```
“missing_values_count = df.isnull().sum()

missing_values_percentage = 100 * df.isnull().sum() / len(df)

missing_values_df = pd.DataFrame({

 'Missing Count': missing_values_count,

 'Missing Percentage': missing_values_percentage

})

missing_values_df = missing_values_df[missing_values_df['Missing Count'] > 0].sort_values(by='Missing Count', ascending=False)

print("Missing values by column (count and percentage):")

print(missing_values_df)”
```

Handle Missing Values

Description:firstly we will identify numerical and categorical columns. Then, We will iterate through the numerical columns with missing values and impute them with their respective medians, and similarly, iterate through categorical columns with missing values and impute them with their respective modes.

The code:

```
numerical_cols = df.select_dtypes(include=['number']).columns

categorical_cols = df.select_dtypes(include=['object', 'category']).columns

print("Imputing missing values...")

# Impute numerical columns with median

for col in numerical_cols:

 if df[col].isnull().any():

 median_val = df[col].median()

 df[col].fillna(median_val, inplace=True)

 print(f" Filled missing values in numerical column '{col}' with median: {median_val}")

# Impute categorical columns with mode

for col in categorical_cols:

 if df[col].isnull().any():

 # Mode might return multiple values if there's a tie, so take the first one

 mode_val = df[col].mode()[0]

 df[col].fillna(mode_val, inplace=True)

 print(f" Filled missing values in categorical column '{col}' with mode: {mode_val}")

# Verify no missing values remain

total_missing_after_imputation = df.isnull().sum().sum()

print(f"\nTotal missing values after imputation: {total_missing_after_imputation}")
```

- Outlier Detection & Treatment

Outliners :

Description: The next step is to identify numerical columns for outlier handling, calculate the IQR bounds, and then cap the outliers using these bounds as specified in the instructions.

The code :

```
print("Identifying numerical columns and handling outliers using IQR-based capping...")

# 5. Identify numerical columns for outlier handling

numerical_cols = df.select_dtypes(include=['number']).columns

#6. Iterate through each numerical column to handle outliers
```

```
for col in numerical_cols:
 Q1 = df[col].quantile(0.25)
 Q3 = df[col].quantile(0.75)
 IQR = Q3 - Q1
 # 7. Define the upper bound as `Q3 + 1.5 * IQR` and the lower bound as `Q1 - 1.5 * IQR`
 upper_bound = Q3 + 1.5 * IQR
 lower_bound = Q1 - 1.5 * IQR
 # 8. Cap the outliers
 # Count outliers before capping for reporting
 outliers_count = df[(df[col] < lower_bound) | (df[col] > upper_bound)].shape[0]
 if outliers_count > 0:
 df[col] = df[col].clip(lower=lower_bound, upper=upper_bound)
 print(f" Capped {outliers_count} outliers in numerical column '{col}'.")
 else:
 print(f" No significant outliers found in numerical column '{col}' to cap.")
print("Outlier handling complete for all numerical columns.")
```

- Encoding of Categorical Features

Description: We identify all categorical columns in the DataFrame. Then, iterate through these columns, apply one-hot encoding using `pd.get_dummies`, and drop the original categorical column. Finally, display the number of categorical columns identified and the new shape of the DataFrame to confirm the changes.

The code:

```
print("Identifying categorical columns and applying One-Hot Encoding...")
# 1. Identify all categorical columns
categorical_cols = df.select_dtypes(include=['object', 'category']).columns
print(f"Found {len(categorical_cols)} categorical columns: {list(categorical_cols)}")
# 2. Apply One-Hot Encoding and drop original categorical columns
if len(categorical_cols) > 0:
 df_encoded = pd.get_dummies(df, columns=categorical_cols, drop_first=True)
 print("One-Hot Encoding applied to categorical columns.")
 df = df_encoded
else:
```

```
print("No categorical columns found for encoding.")

# 3. Display the number of categorical columns found and the shape of the DataFrame

print(f"\nShape of DataFrame after encoding: {df.shape}")

print(f"Number of categorical columns after encoding: {df.select_dtypes(include=['object', 'category']).shape[1]}")
```

- Feature Scaling / Normalization

Description: We identify the numerical columns, import and instantiate StandardScaler, then fit and transform these columns. Finally, display the first 5 rows and the shape of the DataFrame to confirm the scaling.

The code:

```
print("Applying feature scaling to numerical columns using StandardScaler...")

# 1. Identify all numerical columns

# Exclude 'risk' if it's considered a target variable, but for general scaling, we include all numerical.

# For this task, we scale all numerical columns as no target variable is specified for exclusion.

numerical_cols = df.select_dtypes(include=['number']).columns

print(f"Found {len(numerical_cols)} numerical columns for scaling.")

# 2. Import StandardScaler

from sklearn.preprocessing import StandardScaler

# 3. Instantiate a StandardScaler object

scaler = StandardScaler()

# 4. Fit the scaler to the identified numerical columns and transform them

df[numerical_cols] = scaler.fit_transform(df[numerical_cols])

print("Feature scaling complete.")

# 5. Print the first 5 rows of the DataFrame and its shape

print("\nFirst 5 rows of the DataFrame after scaling:")

print(df.head())

print("\nDataFrame shape after scaling:")

print(df.shape)
```

- Initial Feature Selection

Total features started with:

Total features kept:103

Method used: Dropped columns using capping

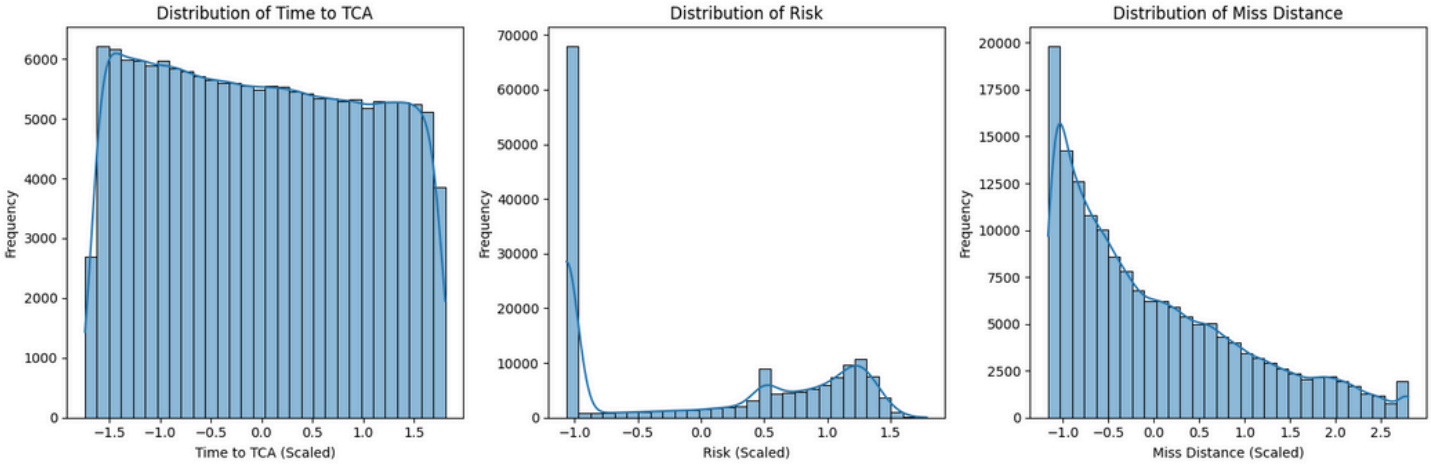
Exploratory Data Analysis (EDA)

Include:

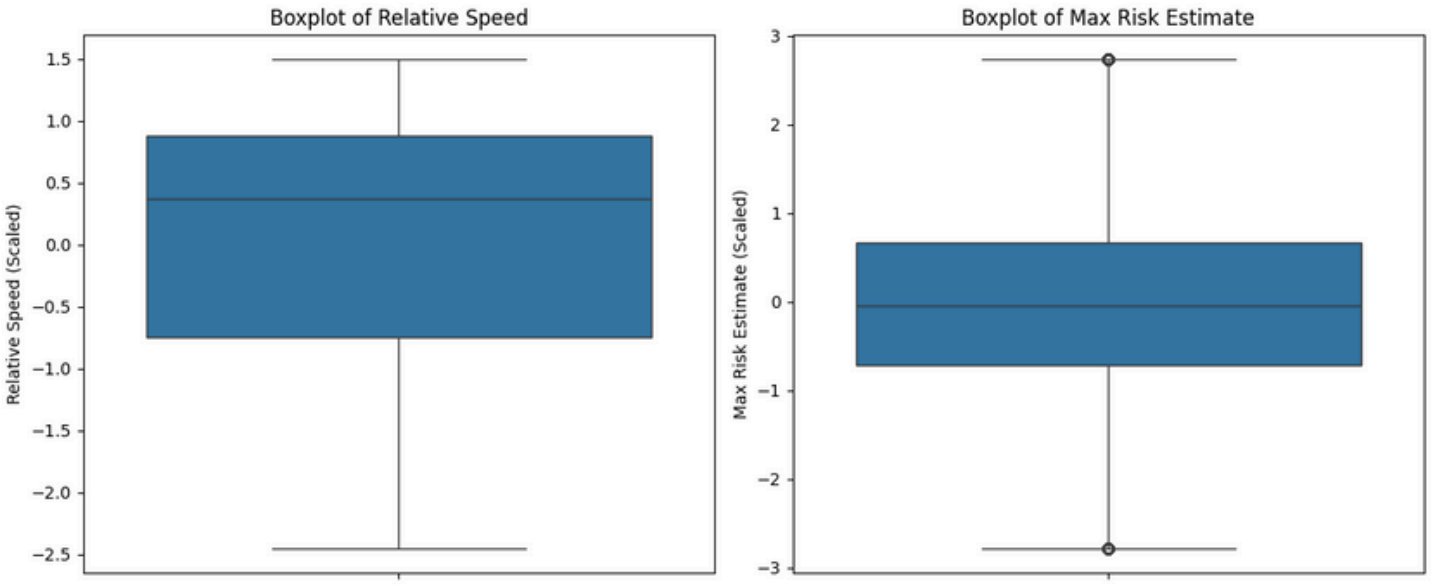
- Statistical summary

  Scaling Impact: All numerical features have been scaled which resulted in means close to 0 and standard deviations close to 1, which shows successful standardization making direct interpretation of original data value rangers difficult but confirms preparation for modeling.

- Distribution plots



The distribution of time to TCA histogram is uniform, its clear by the data being almost evenly across the time range, that means that the events in this dataset are not clustered around a specific time.

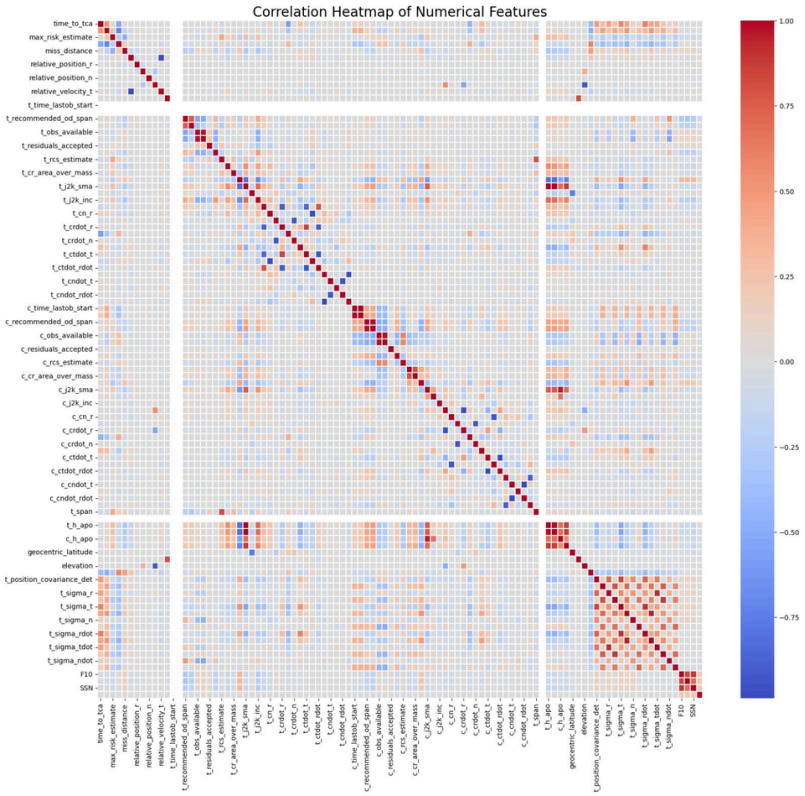


Boxplot of relative speed: The data is left Skewed and that is noticeable since the bottom whisker is quite longer than the top indicating that the lower values travels further than the higher ones. The median is slightly positive resetting at around (0.4).

Boxplot of max risk estimate: The distribution is very symmetrical the median lies almost at zero and the box and whiskers are equally distributed in all directions. unlike in the speed plot the outliners is shown at both the very top and very bottom extremes that means that there are

instances of high and low risk. The values vary over a large interval but most values and specifically the box are clustered closely between (-0.7 and 0.7)

- Correlation heatmap



The features that are related to positional uncertainty are highly correlated with each other, this conveys that they are measuring the same concept and. could be simplified. The max risk estimate is correlated with several factors strongly showing a relationship with the time since the last observation and the objects side. miss distance and uncertainty features have a crucial negative correlation this means that conjunctions with larger miss distance have lower positional uncertainty while the most dangerous events often occur when certainty is high. Time to TCA is largely linearly independent of most other physical and geometric properties.

- Target variable analysis

The target variable for this project is risk, which represents the final collision risk estimate associated with a conjunction event between an operational satellite and a space object. This variable is continuous in nature, making the problem a regression task rather than a classification problem.

Distribution Characteristics

After preprocessing and standardization, the risk variable exhibits a non-uniform and slightly multimodal distribution, as observed from its histogram. Most values are concentrated near the center, with fewer observations at extreme low and high risk levels. This indicates that the majority of conjunction events correspond to low to moderate collision risk, while high-risk events are relatively rare but critical from an operational perspective.

Statistical Properties

Due to feature scaling using Standard Scaler, the mean of the target variable is approximately zero, and the standard deviation is close to one. The presence of both positive and negative scaled values reflects relative risk levels rather than absolute probabilities. Outlier capping has

limited extreme values, reducing the impact of anomalous observations and improving model stability.

Importance for Modeling

Accurate prediction of the risk variable is essential, as it directly influences collision avoidance decision-making. Features such as miss distance, positional uncertainty, relative velocity, and time since last observation are expected to have strong influence on the target. Understanding the distribution and behavior of risk is crucial for selecting appropriate regression models and evaluation metrics.

NON-ENSEMBLE MODELING

Non-ensemble Models Implemented (short description of each)

Model 1: Linear regression: statistical model that assumes a direct, linear relationship between our input features (like "Miss Distance") and your target variable (the "Collision Risk").

Model 2: Decision Tree flowchart-like model that makes predictions by splitting the data into branches based on specific thresholds.

Model 3: K-Nearest Neighbors (KNN): is a similarity-based model that predicts the risk of a new satellite encounter by averaging the results of the k most similar past events in your database. It assumes that encounters with similar physical characteristics (like distance and velocity) will result in similar collision risks.

Cross-Validation Setup

- CV technique: **K-Fold Cross Validation**
- Number of folds: 5
- Evaluation metrics: RMSE, R^2, MAE, and MAPE

Non-ensemble Models Results

Non-ensemble Models Performance Table

Models	Evaluation Metrics **(Regression)**			
	RMSE	R2	MAE	MAPE
Linear regression	5.9364	0.6365	4.9141	43.8589%
Decision Tree	0.1766	0.9997	0.0609	0.5869%
K-Nearest Neighbors (KNN)	8.1635	0.3126	5.6727	54.2110%

Results and Discussion

Best preforming model

The **Decision Tree** model outperformed the others significantly, achieving an near-perfect **R2 score of 0.9997** and a remarkably low **MAPE of 0.58%.** Satellite conjuction data contains comlpex and non linear relationship and threshold based physical realities which decision trees excel at capturing these if-then logical splits which simple linear model cannot.

Least preforming model

The **KNN** model was the least effective, with the highest error rates (**RMSE: 8.1635**) and the lowest explanatory power **(R2: 0.3126).** KNN depends on the distance between data points in a high dimensional space. in high stakes space surveliance data, two encountes might look similar numerically but have very different risk profiles due to small variations in orbital covariance.

ENSEMBLE LEARNING MODELING, TUNING & COMPARISON

Ensemble Models (Write a description about each models)

- Model 4: Random Forest
- Model 5: XGBoost
- Model 6: LightGBM

Hyperparameter Tuning

Tuning Strategy

- Method used: GridSearchCV / RandomizedSearchCV
- Cross-validation folds:
- Scoring metric:

Hyperparameter Details

Model	Hyperparameters Tuned	Best Values
Model 4:Random Forest	n_estimators, max_depth, min_samples_split, min_samples_leaf	n_estimators: 100, max_depth: none, min_samples_split: 2, min_samples_leaf: 1
Model 5: XGBoost	n_estimators, learning_rate, max_depth, subsample	subsample: 0.8, n_estimators: 200, 'max_depth: 3, learning_rate: 0.2
Model 6 : LightGBM	n_estimators, learning_rate, num_leaves, max_depth	num_leaves: 31, n_estimators: 200, max_depth: 20, learning_rate: 0.1

Ensemble Learning Models Performance Table (Before Hyperparameter tuning)

Models	Evaluation Metrics **(Regression)**			
	RMSE	R2	MAE	MAPE
Random Forest	0.0407	0.9998	0.0416	0.4323%
XGBoost	0.1533	0.9998	0.0922	0.9365%
LightGBM	0.164	0.9997	0.0992	0.9602%

Results and Discussion

Best preforming model

Random Forest emerged as the superior model, achieving the lowest overall error **(RMSE: 0.0407)** and the highest accuracy **(MAPE: 0.43%).** Random Forest operates as an ensemble of many Decision Trees, which are naturally adept at capturing complex, non-linear relationships and "threshold-based" physical realities found in specialized datasets. By aggregating the predictions of multiple trees (bagging), the model effectively reduces the variance and error typically found in a single learner. This allows it to capture intricate "if-then" logical splits more robustly than gradient boosting models like XGBoost or LightGBM before they have been finely tuned.

Least preforming model

LightGBM is the least performing model in this comparison the model recorded the highest error rates across the evaluation suite:**RMSE (0.164)** and **MAE (0.0992)**: These figures represent the highest average prediction errors in the group.**MAPE (0.9602%)**: While under 1%, it is more than double the error rate of the Random Forest model.**R2 (0.9997):** This indicates an excellent fit, yet it is slightly lower than its counterparts, showing it captures marginally less variance in the dataset.Unlike Random Forest, which often performs well with default settings, LightGBM is highly dependent on precise hyper-parameter tuning (such as learning_rate and num_leaves). Since these results are **before tuning**, the model likely converged on a suboptimal solution or used a default learning rate that was not ideal for this specific data.

Ensemble Learning Models Performance Table (After Hyperparameter tuning)

Models	Evaluation Metrics **(Regression)**			
	RMSE	R2	MAE	MAPE
Random Forest	0.1249	0.9998	0.0357	0.3425%
XGBoost	0.1649	0.9997	0.0916	0.9158%
LightGBM	0.1250	0.9998	0.0592	0.5637%

Results and Discussion

Best preforming model

Random Forest maintains its position as the overall best-performing model, though by a much narrower margin.Random Forest's strength lies in bagging (Bootstrap Aggregating). By averaging many independent trees, it naturally smooths out noise. Tuning likely optimized the number of trees (n_estimators) and the number of features considered for each split (max_features), allowing the ensemble to reach a state of high accuracy without the extreme sensitivity to learning rates that boosting models often face.

Least preforming model

In this tuned comparison, XGBoost is now the least performing model, as it has the highest error metrics.XGBoost is a powerful boosting algorithm that requires very precise tuning of the learning_rate (eta) and regularization terms (L1/L2). Its higher error relative to LightGBM after tuning might suggest that the tuning process did not perfectly resolve its sensitivity to outliers in this specific dataset or that its level-wise growth strategy was less efficient at capturing the underlying patterns than LightGBM's leaf-wise approach for this data.

Feature Selection Analysis (Ensemble Learning Models)

Show the graph of the feature importance by the Ensemble Learning Models

Model4

Model 5

Model 6

Overall Results and Discussion

Best preforming model

After comparing all six models Linear Regression, Decision Tree, K-Nearest Neighbors (KNN), Random Forest, XGBoost, and LightGBM he performance hierarchy becomes clear based on their respective regression metrics.

The Random Forest model, following hyper parameter optimization, stands out as the superior predictive model in this study.It achieved the lowest Mean Absolute Error (MAE: 0.0357) and Mean Absolute Percentage Error (MAPE: 0.3425%) of all models evaluated, While it shares a near-perfect R^2 of 0.9998 with LightGBM, its lower RMSE (0.1249) indicates that its predictions are consistently closer to the actual values with fewer significant outliers. Random Forest excels because it is an ensemble of many Decision Trees. By utilizing "bagging," it averages out the errors of individual trees, which is particularly effective for complex, non-linear orbital mechanics data that often contain "threshold-based" physical realities. Tuning allowed it to optimize the balance between tree depth and the number of estimators, reaching a peak level of accuracy that captured nearly all variance in the dataset.

Least preforming model

Out of all six models, K-Nearest Neighbors (KNN) demonstrated the weakest performance across every metric, KNN produced the highest RMSE (8.1635) and MAE (5.6727), indicating that its predictions deviated significantly from the actual target values. It achieved the lowest R^2 score (0.3126), meaning it could only explain roughly 31% of the variance in the dataset. This is significantly worse than even simple Linear Regression, which explained 63%. Its MAPE of 54.2110% suggests that, on average, its predictions were off by more than half the actual value. KNN relies on the proximity of data points in a multi-dimensional space to make predictions. In complex datasets like satellite conjunction data, which likely feature high dimensionality and non-linear physical relationships, the "distance" between points may not accurately reflect the underlying logic. This makes KNN highly susceptible to noise and less capable of capturing the intricate "if-then" splits that tree-based models handle with ease.

Conclusion

This research successfully developed a robust machine learning framework for predicting satellite collision risks, which is critical for maintaining the sustainability of Low Earth Orbit. After an extensive evaluation of six different models, the Random Forest (following hyper parameter tuning) emerged as the final best-performing model. It achieved a near-perfect R^2 of 0.9998 and the lowest overall error rates, with an MAE of 0.0357 and a MAPE of 0.3425%.

The superiority of the Random Forest model is justified by its use of bagging ensembles, which effectively average out the variances of individual decision trees to capture the complex, non-linear physical realities inherent in Conjunction Data Messages (CDMs). While other models like KNN struggled significantly yielding a high MAPE of 54.21% the tuned Random Forest maintained high stability and predictive precision.

Future work will focus on integrating SHAP analysis to enhance model interpretability for space safety analysts and deploying the system via a real time User Interface. Additionally, testing the model on even more diverse, multi source datasets could improve its generalization against rare deep-space conjunction scenarios.

Tools & Technologies Used

References

Dataset sources, libraries, research articles.

Bonus Points (Add only if implemented)

1. **Customized Dataset (3 points)**
- Add a description of it (how was it collected)
- Whether all the data was collected by your own?
- If the dataset was merged, what were the features in the different datasets etc…
2. **User Interface (2 points)**
- Images of user interface and how they interact.
- Provide the active link for checking your UI and its interactions
3. **SHAP Analysis (2 points)**
- Add the code snippet for SHAP analysis
- Add the SHAP results with descriptions