Astrophysical Dataset Analysis: A Comprehensive Exploration of Orbital Characteristics and Predictive Insights

1. Introduction

- **Objective:** This analysis aims to employ data analytics and machine learning techniques to estimate the risk of an asteroid being hazardous to Earth, based on the provided features.

- **Dataset Overview**: The dataset contains structured data about various asteroids that may approach Earth. This dataset is essential for analyzing the potential hazards posed by asteroids and supports ongoing planetary defense efforts.
    - Type of Data**:** The dataset includes both numerical and categorical data related to asteroids' physical and orbital characteristics.
    - Key Features:
        - Name: Designation of the asteroid.
        - Semi Major Axis: Average distance from the Sun.
        - Aphelion Distance: Farthest distance from the Sun in its orbit.
        - Miss Distance: Closest approach distance to Earth.
        - Eccentricity: Measure of orbit deviation from a circle.
        - Velocity: Speed relative to Earth.
        - Approach Date**:** Date of closest approach.

2. Data Preprocessing

- **Overview of Missing Data Handling:** In this analysis, missing data was addressed through systematic identification and imputation techniques to ensure data integrity. Initially, missing values were detected using pandas functions, followed by imputation strategies. For numerical features, missing values were filled using *backward* fill *(*bfill*)* and *forward fill (*ffill*)* methods, along with interpolation where necessary. Categorical features had missing values replaced with the *mode* to preserve their distribution. Context-specific handling was also applied to certain features, such as approach dates and distances, using relevant derived calculations. Finally, the dataset was re-evaluated to confirm that all missing values were resolved, enhancing the robustness of the analysis.

- **Feature Conversion:** Feature conversion was performed to enhance the dataset's usability and analytical capabilities. Key conversions included transforming distance measurements into consistent units, such as converting

miles and kilometers to astronomical units (AU). Additionally, orbital parameters, like the semi-major axis and perihelion distance, were converted from AU to meters for more precise calculations. Temporal features, such as the approach date, were formatted as datetime objects to facilitate time-based analyses. These conversions ensured that all features were standardized, making the dataset more suitable for further analysis and modeling.

- **Data Type Adjustments:** Data type adjustments were made to ensure that each feature in the dataset was correctly formatted for analysis. Numerical features were verified and converted to their respective types, such as integers and floats, to facilitate mathematical operations. Categorical variables were changed to the appropriate categorical data type to optimize memory usage and improve performance. These adjustments were crucial for enhancing data accuracy and enabling efficient processing in subsequent analytical tasks.

- **Handling Date Columns:** Date-related data was processed to create new features that enhance the analytical depth of the dataset. The approach date, originally in separate year, month, and day columns, was combined and transformed into a single datetime object. ‘Time Until Approach' feature represented the number of days until the asteroid's close approach to Earth. By calculating the difference between the current date and the approach date, this new feature provides valuable insights into the timing of potential asteroid encounters, facilitating further analysis of their implications.

3. Exploratory Data Analysis (EDA)

- **Descriptive Statistics:** The dataset includes key statistics for various features related to asteroid close approaches. The count for most variables ranges between 2,800 and 4,000 entries, with some missing values. The mean relative velocity is approximately 50,516 km/hr, with a standard deviation of 26,530 km/hr, indicating a significant variation in the speed of approaching asteroids. The minimum recorded velocity is around 1,207 km/hr, while the maximum reaches 160,681 km/hr. Miss distances show an average of 0.258 astronomical units (AU), ranging from 0.000178 to 0.499884 AU, while distances in kilometers and miles exhibit similar variability. The "Jupiter Tisserand Invariant" has a mean of 5.126, with values spanning from 2.367 to 9.025, indicating different asteroid orbital dynamics. The semi-major axis averages at 1.358 AU with variability reflected by a standard deviation of 0.465. Temporal features like the approach year cover a range from 1995 to 2016, with a median around 2008, while the approach day and month exhibit consistent statistics across the dataset. The semi-major axis, mean anomaly, and other orbital parameters display standard deviations that suggest diversity in orbital characteristics.

- **Visualizations:**

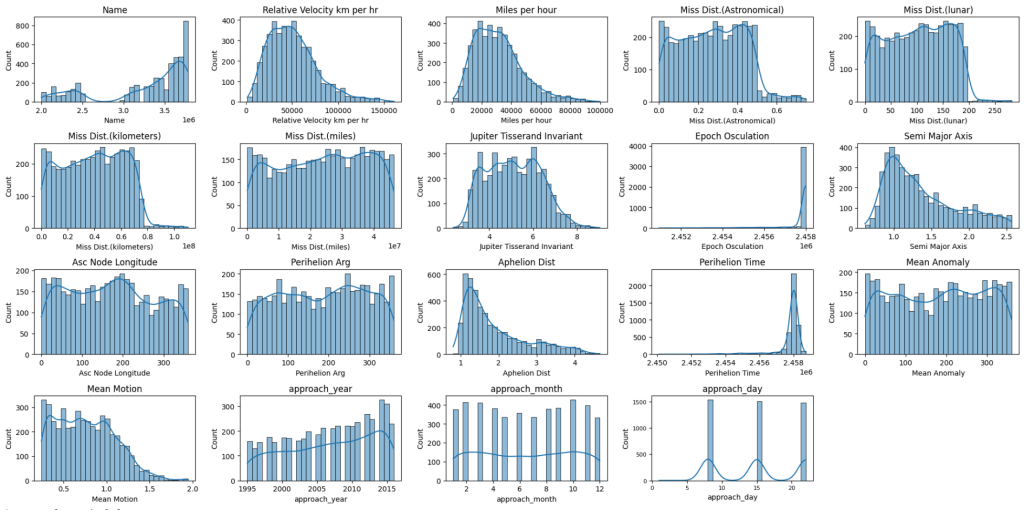
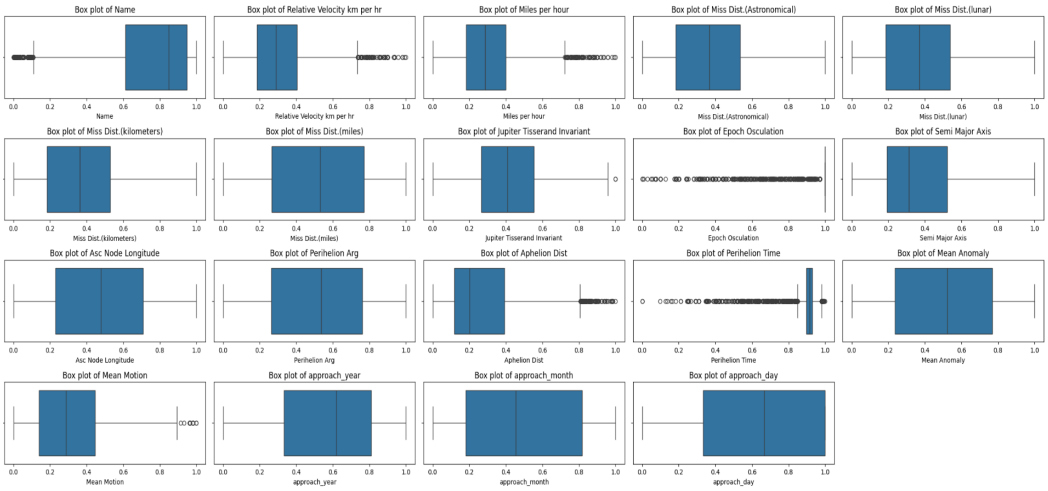


Figure 1. Histograms

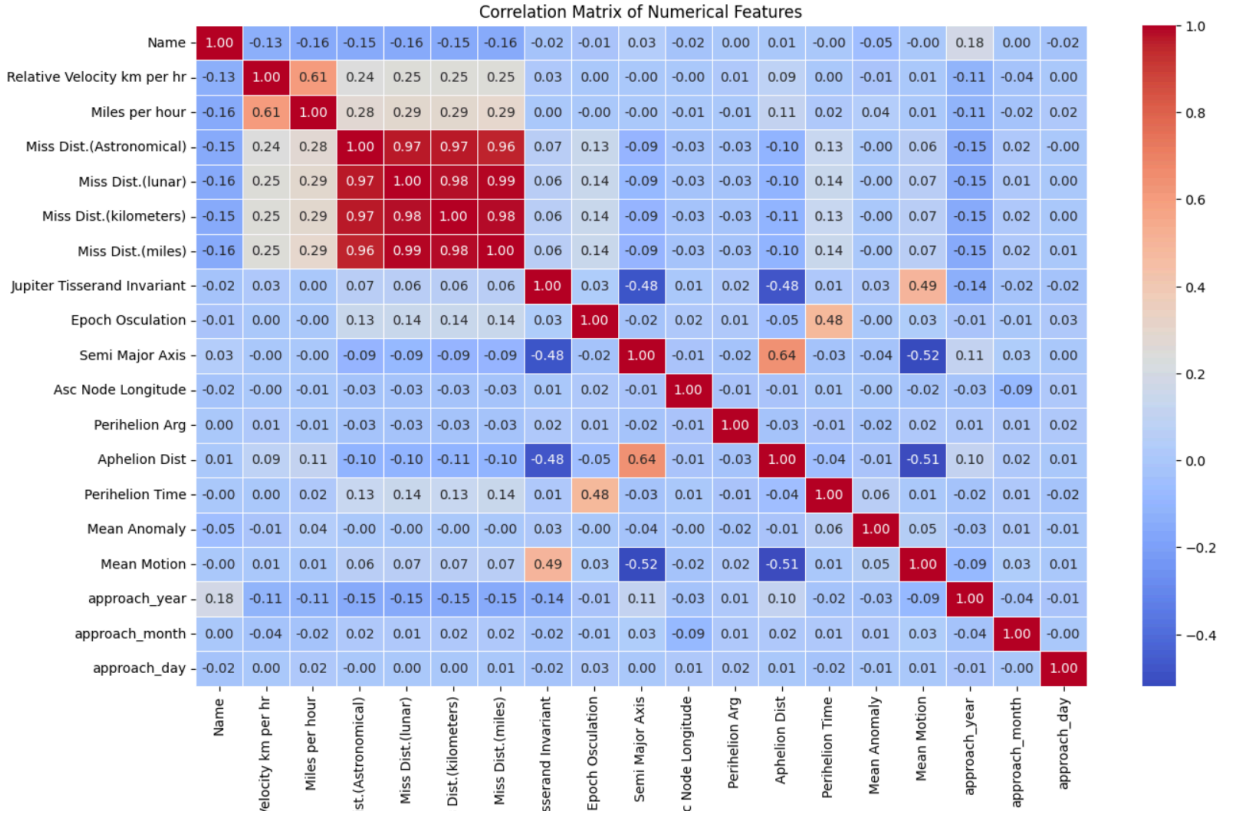
The histograms reveal several insights:

1. Some features, like "Relative Velocity km per hr" and "Jupiter Tisserand Invariant," show clustering, suggesting there are common characteristics shared by the majority of asteroids.
2. There is a noticeable increase in the number of observations in more recent years, likely due to advancements in detection technology.
3. The distribution of orbital characteristics, including "Semi-Major Axis" indicates that while some asteroids have typical near-Earth orbits, there is also significant variation among the dataset.
4. Additionally, some features show the presence of extreme values, pointing to potential outliers that could be explored further.



Here are some key insights from the boxplots:

1. Outliers: The Epoch Osculation and Perihelion Time features have many outliers, suggesting some unusual astronomical data that may require further analysis.
2. Skewness: Perihelion Time and Mean Motion show right-skewed distributions, indicating that while most data points are clustered, a few exhibit significantly higher values, worth investigating.
3. Variability: Features like Relative Velocity and Semi Major Axis show wide interquartile ranges, revealing diverse characteristics among the celestial objects.
4. Compactness: Features like Miss Distance (Astronomical) and Semi Major Axis show more uniformity, indicating most objects share similar proximity and orbital dimensions.



This heatmap shows the correlation matrix of numerical features in the dataset, with the color scale ranging from blue (negative correlation) to red (positive correlation). The diagonal reflects perfect correlations (1.0) for each feature with itself.

Strong positive correlations are seen between the various miss distance measures (kilometers, miles, lunar, astronomical), which is expected as they

represent the same metric in different units. Notably, Semi Major Axis and Aphelion Distance have a strong positive correlation, suggesting a direct relationship between these orbital elements.

Some negative correlations are observed, such as between Semi Major Axis and Mean Motion, reflecting Kepler's third law, where larger orbits are associated with slower motions. Similarly, Aphelion Distance negatively correlates with Mean Motion.

Many features show near-zero correlations, like between Epoch Osculation and other features, indicating independence or weak relationships. These insights highlight a mix of strongly coupled and independent features, giving a better understanding of the relationships between the celestial properties in the dataset.

4. Feature Engineering and Transformation

- **Feature Creation:**The date feature, created from year, month, and day columns, enabled the calculation of Time Until Approach, which offers a clear timeline for when an object will make its closest pass to Earth. The Miss Distance vs Semi-major Axis Ratio compares the miss distance to the object's orbit size, providing a measure of how close the approach is relative to the full orbital path.

  Eccentric Anomaly, solved using Kepler's equation, and True Anomaly offer detailed insights into the object's position in its orbit, particularly useful for non-circular paths. Heliocentric Distance calculates how far the object is from the Sun at any point in time, while Orbital Period determines how long it takes for the object to complete one full orbit around the Sun, helping with future predictions.

  Velocity at Perihelion/Aphelion measures the object's speed at its closest and farthest points from the Sun, indicating how gravitational forces affect velocity. Escape Velocity shows the speed needed to break free from the Sun's gravity, while Specific Orbital Energy helps evaluate the energy balance and stability of the orbit. The Velocity Change Indicator compares actual velocity to expected orbital velocity, highlighting deviations that may indicate external forces or perturbations in the object’s path.

- **Normalization:** After applying Min-Max scaling to the numerical features, the values in the dataset are normalized to a range between 0 and 1. This technique ensures that features with different scales, such as distances in miles or kilometers and velocities in km/s or km/h, are brought onto a common scale, preventing any single feature from dominating during analysis or model training.In this dataset, features like Miss Distance, Relative Velocity vary greatly in magnitude. Without normalization, algorithms like k-nearest neighbors or gradient descent-based models might give disproportionate weight to features with larger scales. By scaling, all features contribute equally, helping models converge faster and perform better.

The normalized values allow for better comparison between objects. For example, even though the Miss Distance (lunar) and Relative Velocity km per hr have different physical units and ranges, scaling them between 0 and 1 helps in identifying patterns and trends more clearly. Additionally, normalization improves model performance by avoiding numerical instability, especially in algorithms sensitive to feature scale.

- **Outlier Detection:** Outlier detection is crucial for ensuring the reliability of our analysis, as outliers can significantly influence the results of statistical and predictive models. In this study, we utilized the z-score method with a threshold of 3 to identify outliers, resulting in the detection of 341 rows. These outliers were primarily linked to features such as Relative Velocity km per sec*,* Miss Distance (kilometers)*,* and Jupiter Tisserand Invariant. Their occurrence across various close approach dates, from the early 1990s to 2016, indicates their representation of extreme values rather than isolated incidents.

  Recognizing the implications of these outliers is essential. While they may highlight critical cases, such as fast-moving asteroids significant for risk assessment, it is also important to consider the possibility of data collection errors.

- **Skewness Analysis:** Skewness analysis is an essential step in understanding the distribution of our dataset. Before feature engineering, several numerical features exhibited varying degrees of skewness, with notable values such as Epoch Osculation at -3.865882 and Aphelion Dist at 1.156590. After feature engineering, new features were introduced, including Orbital Period and Escape Velocity, which resulted in additional skewness values. For instance, the Heliocentric Distance recorded a skewness of 1.433490, indicating a rightward skew.

  This analysis serves as a preliminary assessment of the data distribution, informing subsequent transformations needed to enhance model performance. By addressing skewness, we can improve the robustness and accuracy of our predictive models, ultimately contributing to more reliable risk assessments.

5. Model and Training Method

We developed a binary classification model using a simple feedforward neural network and enhanced its performance evaluation through K-Fold Cross Validation and hyperparameter tuning.

- **Model Architecture:**

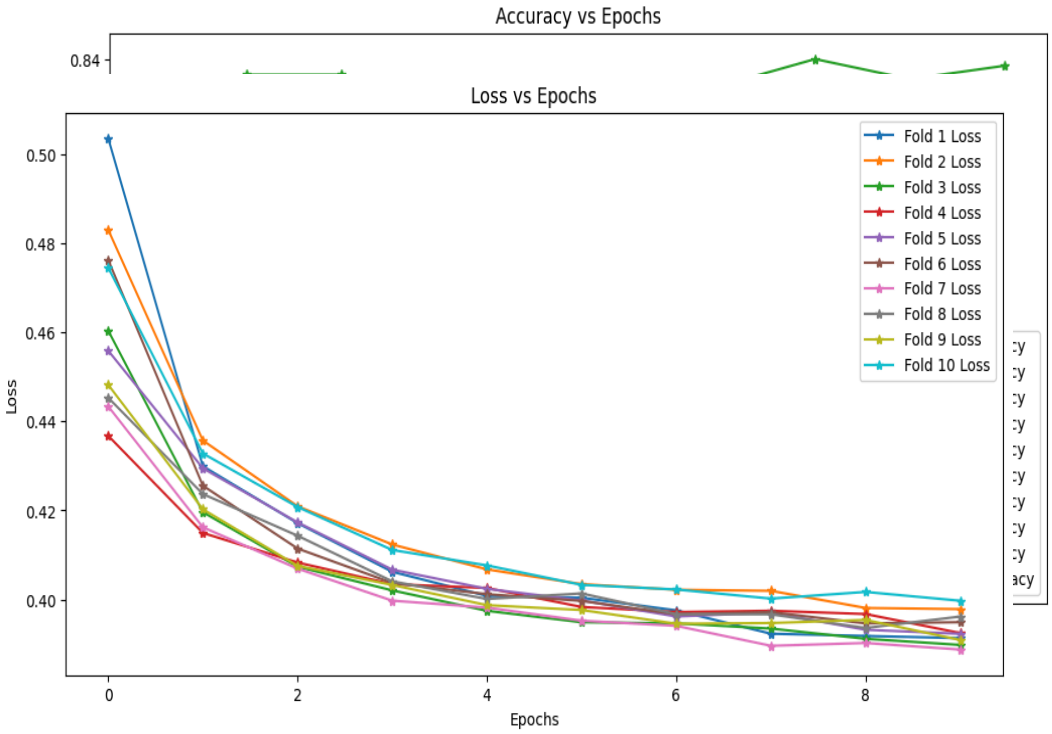
The model consists of:

- Input Layer: Adjusted dynamically based on the dataset's feature size.
- Hidden Layer 1: A fully connected layer with configurable units (default 32) and ReLU activation for non-linearity.
- Hidden Layer 2: Another dense layer with 16 units and `tanh` activation, useful for capturing complex relationships.
- Output Layer: A single unit with a sigmoid activation function, producing a probability for binary classification.

We compiled the model with the Adam optimizer and binary cross-entropy loss, suitable for binary classification tasks.

- **K-Fold Cross Validation:**

  We used K-Fold Cross Validation (k=10) to ensure the model's robustness by splitting the data into 10 folds, where the model was trained on 9 folds and validated on the remaining fold. This was repeated for all folds, allowing the model to be evaluated on different subsets of data.



- **Hyperparameter Tuning with Grid Search**

We applied Grid Search to optimize the model's learning rate and the number of units in the first hidden layer. By testing combinations of these parameters (learning rates: 0.001, 0.01, 0.1 and units: 16, 32, 64), we found the best configuration that maximized accuracy.

- **Conclusion**

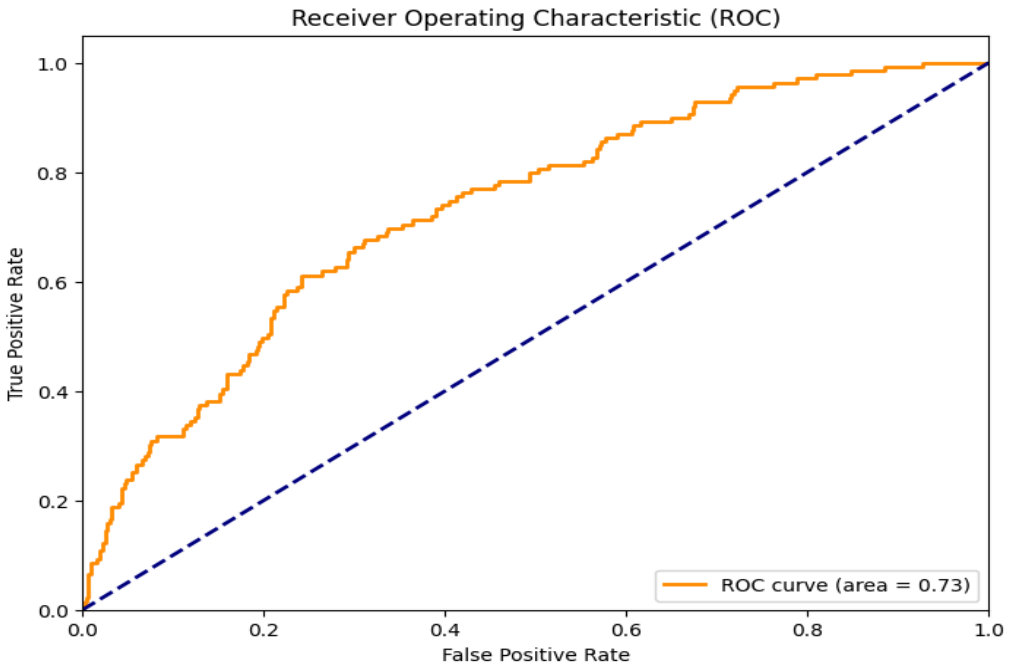
  By combining a well-structured neural network architecture, K-Fold Cross Validation for evaluation, and hyperparameter tuning using Grid Search, we ensured a model that generalizes well and achieves high accuracy on unseen data. We got an accuracy of around **0.83.**

- **ROC Curve:**

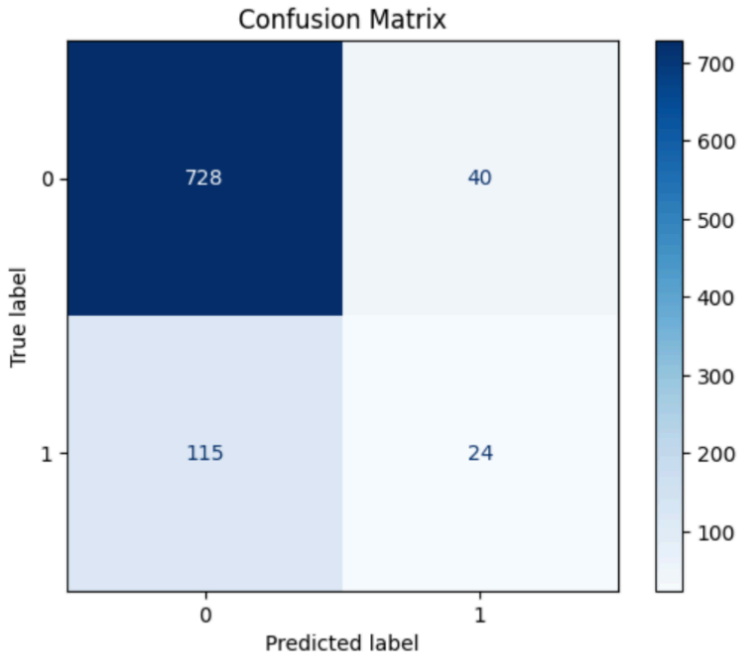
  The ROC (Receiver Operating Characteristic) curve provides insight into the trade-off between the true positive rate (sensitivity) and the false positive rate.

  - AUC Score: The area under the curve (AUC) is approximately 0.73, suggesting that the classifier is good at distinguishing between positive and negative classes. An AUC of 1 represents a perfect classifier, while an AUC of 0.5 represents a random guess.
  - Threshold Selection: The curve shows that as you decrease the classification threshold (moving right along the curve), the true positive rate increases but at the cost of a higher false positive rate.

  In summary, with an AUC of 0.73, the model shows good performance in differentiating between classes.



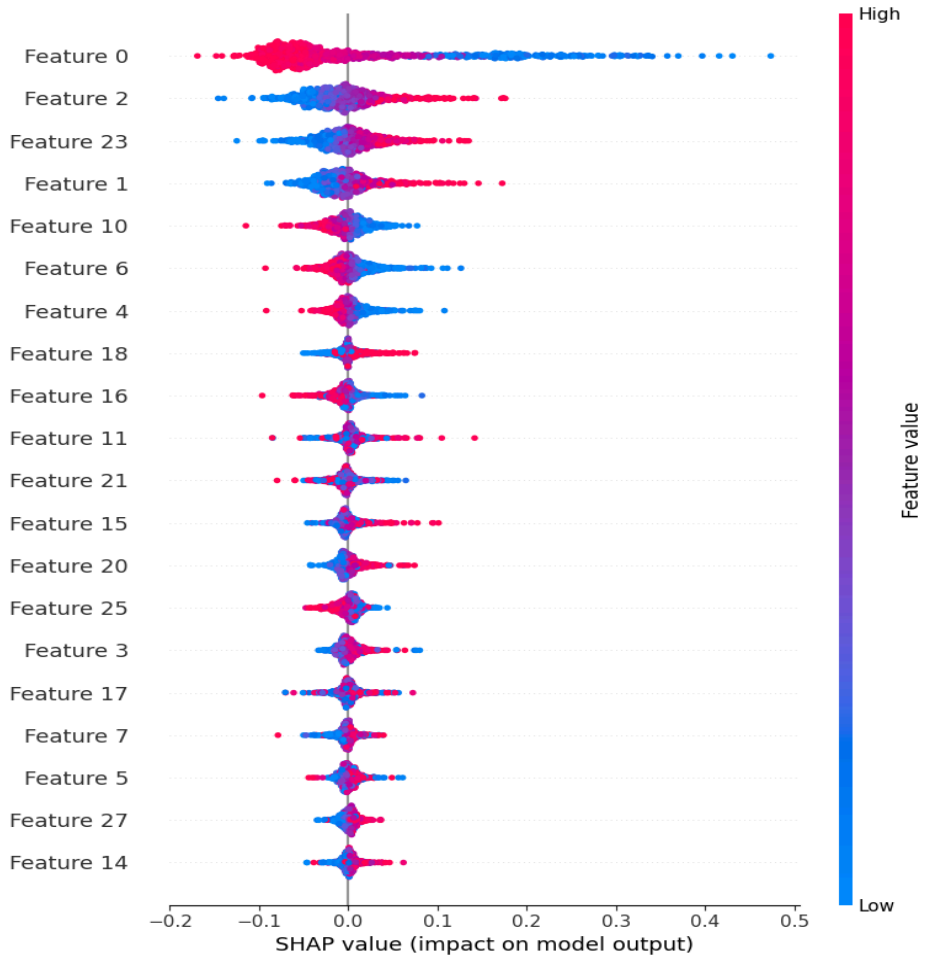
Confusion Matrix:



The confusion matrix gives a breakdown of the model's predictions into true positives, true negatives, false positives, and false negatives:

- True Negatives (TN): The model correctly classified 728 instances as negative (0).
- False Positives (FP): There are 40 instances where the model incorrectly classified a negative instance as positive.
- False Negatives (FN): The model missed 115 positive instances by classifying them as negative.
- True Positives (TP): 24 instances were correctly classified as positive.

This matrix suggests that the model is better at identifying negatives (TN) than positives (TP), as reflected in the Confusion matrix.



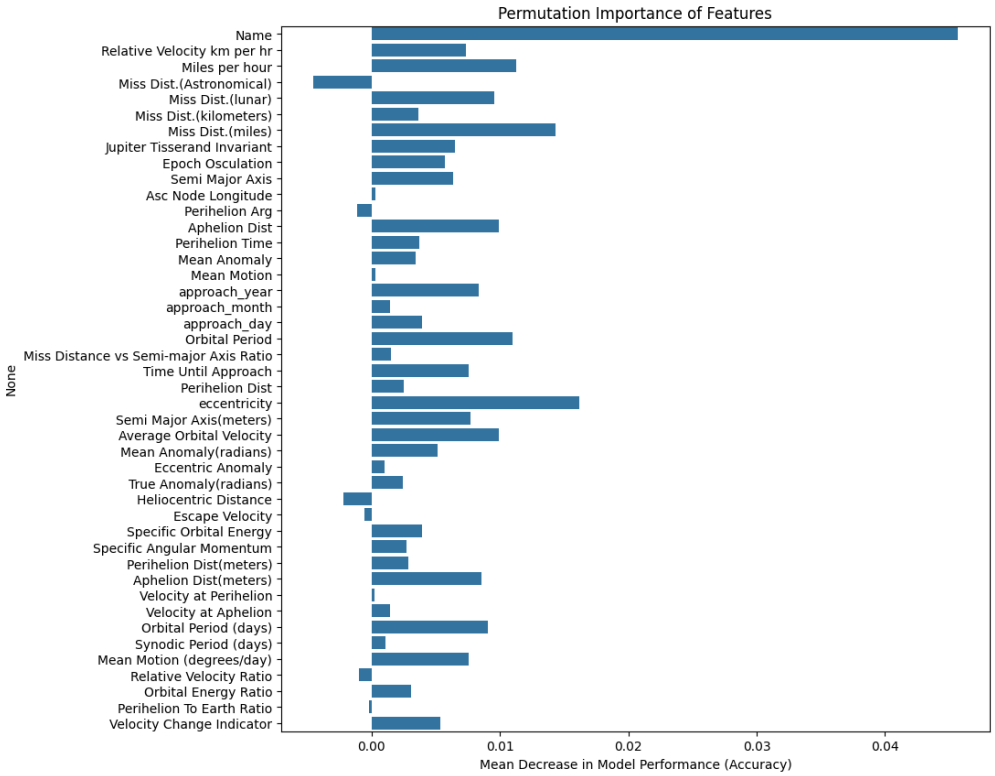
- **SHAP Plot (Feature Importance):**

  The SHAP (SHapley Additive exPlanations) plot demonstrates the importance of various features by showing their impact on the model's output.

  - Feature Importance: Feature 0 is the most influential feature, as evidenced by the broader SHAP values spread across both the positive and negative directions. This means changes in Feature 0 significantly impact model predictions.
  - Color Scheme: The color gradient represents the feature values, with high values in red and low values in blue. For some features, higher values increase the model output, while for others, lower values do.
  - Spread of SHAP Values: The wider spread of SHAP values indicates that a feature contributes to a larger range of prediction outputs. For example, Feature 0 has the widest spread, meaning it heavily influences predictions.

Overall, the plot highlights the importance of various features in model decision-making, with Feature 0 being the most significant.

- **Permutation Feature Importance:**



Permutation Feature Importance is a technique used to understand how each feature in a model affects its performance by observing the mean decrease in accuracy when the feature is randomly shuffled. This method helps identify the most important features contributing to the model's predictive power.

- **Key Inferences:**

The analysis of the model reveals that certain features significantly influence its performance. The Name feature shows the highest importance, with a notable decrease in accuracy when permuted, indicating its strong predictive value. Similarly, Miss Distance (in Astronomical Units) and Relative Velocity (km/hr) also play crucial roles in enhancing model accuracy.

On the other hand, several features, including Orbital Period, Heliocentric Distance, and Mean Anomaly (radians), exhibit minimal importance, suggesting they contribute little to the prediction task. Features like Semi Major Axis, Velocity at Perihelion, and Orbital Energy Ratio are similarly low in impact. Some features, such as Name, may serve primarily as identifiers rather than useful

predictors, while Velocity Change Indicator and Eccentric Anomaly could be excluded from the model without significant loss in performance.

In conclusion, the permutation importance analysis highlights that Name, Miss Distance (Astronomical), and Relative Velocity are the key variables driving the model's accuracy.

6. Anomaly Detection

- **Overview:** This analysis explores anomaly detection in the asteroid dataset using two methods: Isolation Forest, a machine-learning approach, and Z-Score analysis, a statistical method. The objective is to assess their effectiveness in identifying anomalies.

- **Feature Selection**: We focused on features crucial for detecting unusual asteroid behaviors, such as *Miss Distance (kilometers)*, *Relative Velocity (km/h)*, *Jupiter Tisserand Invariant*, *Aphelion Distance*, *Mean Motion*, and *Perihelion Time*. These features provide insights into the asteroid's trajectory, speed, and orbital characteristics, making them valuable indicators of anomalies.

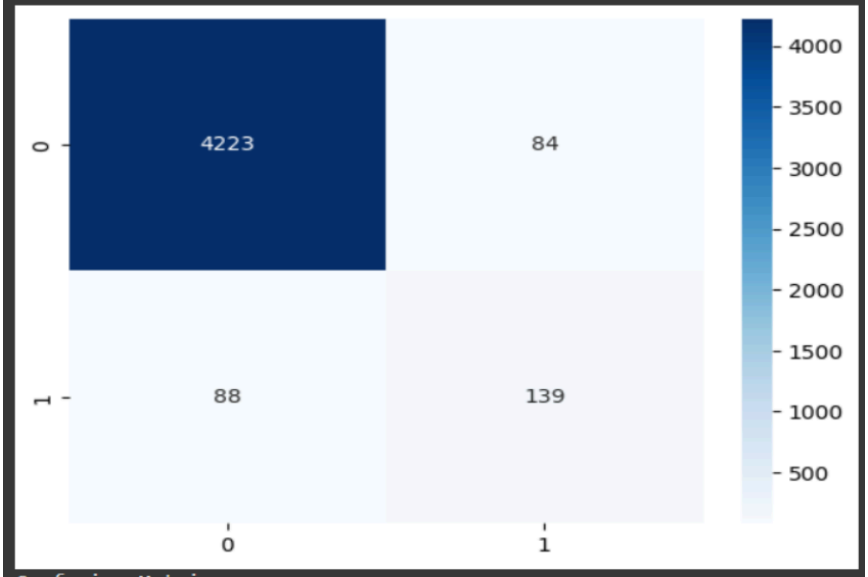
- **Isolation Forest Method**:
  The Isolation Forest algorithm isolates observations to identify anomalies, assuming that roughly 5% of the data are outliers (contamination level set at 0.05). Predictions were mapped into a new column, inbuilt_anomaly, where anomalies are marked as 1, and normal points as 0.

- **Z-Score Analysis**:
  This method calculates the deviation of each data point from the mean. A threshold of 3.05 was used, flagging data points with Z-Scores beyond this range as anomalies. These results were recorded in a custom_anomaly column.

- **Comparison of Methods**:
  To evaluate the methods' performance, a confusion matrix was generated, comparing their outputs. It highlights true positives (anomalies identified by both methods), true negatives (normal points agreed upon), false positives (anomalies only detected by Z-Score), and false negatives (anomalies only detected by Isolation Forest). A heatmap visualizes these results, providing a clear understanding of the methods' alignment.

- **Output Summary**:
  The findings include the number of anomalies detected by each method, the confusion matrix, and a visual heatmap, illustrating the overlap between machine-learning and statistical approaches. This comparison offers a deeper understanding of the dataset's anomalies and the strengths of each detection method.

```
Inbuilt Method Anomalies: 227
Custom Method Anomalies: 223
```




```
Confusion Matrix:
[[4223   84]
 [  88  139]]
Anomalies flagged by both methods: 139
```

Appendix

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Input

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.losses import BinaryCrossentropy

from sklearn.model_selection import KFold

from tensorflow.keras.optimizers import Adam

from scikeras.wrappers import KerasClassifier

from sklearn.model_selection import GridSearchCV

from sklearn.metrics import roc_curve, auc, confusion_matrix, ConfusionMatrixDisplay

from sklearn.inspection import permutation_importance, PartialDependenceDisplay

from sklearn.model_selection import TimeSeriesSplit

from sklearn.preprocessing import StandardScaler

import shap

from datetime import datetime

from scipy import stats

from sklearn.model_selection import train_test_split

import tensorflow as tf

from builtins import range
```

```
file_path = "/content/drive/MyDrive/nsscdataset.csv"

df = pd.read_csv(file_path)
```

EDA

Handling the missing values of some particular columns of the dataframe

```
df['Miss Dist.(Astronomical)'] = df['Miss Dist.(Astronomical)'].fillna(df['Miss Dist.(miles)'] * miles_to_au)
df['Miss Dist.(Astronomical)'] = df['Miss Dist.(Astronomical)'].fillna(df['Miss Dist.(kilometers)'] * km_to_au)
df['Miss Dist.(Astronomical)'] = df['Miss Dist.(Astronomical)'].fillna(df['Miss Dist.(lunar)'] * lunar_to_au)
df['Miss Dist.(miles)'] = df['Miss Dist.(miles)'].fillna(df['Miss Dist.(Astronomical)'] * au_to_miles)
df['Miss Dist.(kilometers)'] = df['Miss Dist.(kilometers)'].fillna(df['Miss Dist.(Astronomical)'] * au_to_km)
df['Miss Dist.(lunar)'] = df['Miss Dist.(lunar)'].fillna(df['Miss Dist.(Astronomical)'] * au_to_lunar)
df['Epoch Date Close Approach'] = df['Epoch Date Close Approach'].interpolate()
```

```
plt.figure(figsize=(20, 10))

# Plot histograms for each numerical feature
for i, feature in enumerate(numerical_features):
   plt.subplot(4, 5, i + 1)
   sns.histplot(df[feature], bins=30, kde=True)
   plt.title(f'{feature}')

plt.xticks(fontsize=8)
plt.yticks(fontsize=8)
plt.tight_layout()
plt.show()
skewness = df[numerical_features].skew()
print("Skewness of numerical features:\n", skewness)
```

```
plt.figure(figsize=(25, 10))

for i, feature in enumerate(numerical_features):
   plt.subplot(4, 5, i + 1)
   sns.boxplot(x=df[feature])
   plt.title(f'Box plot of {feature}')

plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()
plt.show()
```

```
# Calculate correlation matrix
correlation_matrix = df[numerical_features].corr()

# Plot heatmap
plt.figure(figsize=(20, 20))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix of Numerical Features')
plt.show()
```

Dataframe to numpy

```
timestamp_columns = df[numerical_features].select_dtypes(include=['datetime', 'datetime64']).columns
# Converting timestamps to Unix time
for col in timestamp_columns:
   df[col] = df[col].astype('int64') // 1e9

X = df[numerical_features].values # Now extract the numerical features again
X = X.astype(np.float32)
y = df['Hazardous'].values.astype(int).astype(np.float32)
```

Neural Network

```
def create_model(units=32, learning_rate=0.001):
   model = tf.keras.models.Sequential([
       tf.keras.layers.Dense(units, activation='relu', input_shape=(X_train.shape[1],)),
       tf.keras.layers.Dense(16, activation = 'tanh'),
       tf.keras.layers.Dense(1, activation='sigmoid')
   ])
   model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate), loss='binary_crossentropy',
metrics=['accuracy'])
   return model
```

```
def Kfold_cross_validation(X, y, k=5, epochs=10, units=32, learning_rate=0.001):

   losses = [[] for _ in range(k)]                                          # Initializing losses and
accuracies per fold using np.empty
   accuracies = [[] for _ in range(k)]

   X = X.astype(np.float32)
   y = y.astype(int).astype(np.float32)


   kf = KFold(n_splits=k, shuffle=True, random_state=42)                              # Initializing KFold

   fold_index = 0
   for train_index, test_index in kf.split(X):
       X_train, X_test = X[train_index], X[test_index]
       y_train, y_test = y[train_index], y[test_index]


       model = create_model(units=units, learning_rate=learning_rate)                            # Training the
model
       data = model.fit(X_train, y_train, epochs=epochs, validation_data=(X_test, y_test), verbose=0)


       losses[fold_index] = data.history['loss']                                        # Appending the loss
values
       accuracies[fold_index] = data.history['accuracy']                                # Appending the
accuracy values
       fold_index += 1

   # Ploting Loss vs Epochs for each fold
   plt.figure(figsize=(12, 6))
   for i in range(k):
       plt.plot(losses[i], label=f'Fold {i+1} Loss', marker='*')
   plt.title('Loss vs Epochs')
   plt.xlabel('Epochs')
   plt.ylabel('Loss')
   plt.legend()
   plt.show()

   # Ploting Accuracy vs Epochs for each fold
   plt.figure(figsize=(12, 6))
   for i in range(k):
       plt.plot(accuracies[i], label=f'Fold {i+1} Accuracy', marker='*')
   plt.title('Accuracy vs Epochs')
```

```
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()

    return losses, accuracies
```

```
def grid_search(X_train, y_train, X_test, y_test):
    param_grid = {
        'model__learning_rate': [0.001, 0.01, 0.1],
        'model__units': [16, 32, 64],
    }
    model_ = KerasClassifier(model=create_model, verbose=0)                          # Instiatiating
KerasClassifier with a model creation function


    grid = GridSearchCV(estimator=model_, param_grid=param_grid, n_jobs=-1, cv=3)     # Performing grid search
    grid_result = grid.fit(X_train, y_train)

    print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))

    best_model = grid_result.best_estimator_

    test_accuracy = best_model.score(X_test, y_test)                                   # Evaluating the best
model using the score method
    print(f"Test Accuracy: {test_accuracy}")
```

ROC curve

```
false_pos_r, true_pos_r, thresholds = roc_curve(y_test, y_probs)
roc_auc = auc(false_pos_r, true_pos_r)


plt.figure(figsize=(8, 6))
plt.plot(false_pos_r, true_pos_r, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
plt.show()
```

Confusion Matrix

```
from sklearn.metrics import roc_curve, auc, confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred)

# Ploting the Confusion Matrix
plt.figure(figsize=(6, 6))
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.show()
```

Shap Values

```
import shap
```

```
explainer = shap.Explainer(model, X_test)                                    # Fiting the SHAP explainer to the
model
shap_values = explainer(X_test)

# Ploting the SHAP summary to see feature importance
plt.figure(figsize=(10, 6))
shap.summary_plot(shap_values, X_test)
plt.show()
```

Permutation Importance

```
X_test_df = pd.DataFrame(X_test, columns=numerical_features)
plt.figure(figsize=(10, 10))                                  # Plotting the Permutation Importance graph
sns.barplot(x=perm_importance.importances_mean, y=X_test_df.columns)
plt.title("Permutation Importance of Features")
plt.xlabel("Mean Decrease in Model Performance (Accuracy)")
plt.show()
```

Anomaly Detection

```
def anomaly_detection():
    features = [
         'Miss Dist.(kilometers)','Relative Velocity km per hr', 'Jupiter Tisserand Invariant',
        'Aphelion Dist', 'Mean Motion', 'Perihelion Time'
    ]
    data_scaled = df[features]
```

```
    # Inbuilt Method: Isolation Forest
    iso_forest = IsolationForest(contamination=0.05, random_state=42)
    df['inbuilt_anomaly'] = iso_forest.fit_predict(data_scaled)
    df['inbuilt_anomaly'] = df['inbuilt_anomaly'].map({-1: 1, 1: 0})  # 1 = anomaly, 0 = normal
```

```
    # Custom Method: Z-Score
    z_scores = np.abs(stats.zscore(data_scaled))
    threshold = 3.05  # Standard threshold for Z-Score
    df['custom_anomaly'] = (z_scores > threshold).any(axis=1).astype(int)
```

```
    print("Inbuilt Method Anomalies:", df['inbuilt_anomaly'].sum())
    print("Custom Method Anomalies:", df['custom_anomaly'].sum())
```

```
    conf_matrix = confusion_matrix(df['inbuilt_anomaly'], df['custom_anomaly'])
    sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues")
    plt.show()
```

```
    # Print Confusion Matrix and Results
    print("Confusion Matrix:")
    print(conf_matrix)
    print("Anomalies flagged by both methods:", conf_matrix[1, 1])
```

```
# Run anomaly detection
anomaly_detection()
```

Handling Binned Values

```
velocity_mapping = {'Very Slow': 0, 'Slow': 1, 'Fast': 2, 'Very Fast': 3}
df['Relative Velocity km per sec'] = df['Relative Velocity km per sec'].map(velocity_mapping)
# Remove any leading or trailing spaces from column names
df.columns = df.columns.str.strip()

# Perform target encoding for 'Orbital Period'
target_encoding = df.groupby('Orbital Period')['Hazardous'].mean()
df['Orbital Period'] = df['Orbital Period'].map(target_encoding)
```

```
uncertainty_mapping = {'Low': 0, 'Medium': 1, 'High': 2}
df['Orbit Uncertainity'] = df['Orbit Uncertainity'].map(uncertainty_mapping)
```