Predicting Hazardous Near-Earth-Objects

Project phase - 1

50461788 - Sonalira
50471077 - Khyathik

Near Earth Objects

The gravitational force of planets allows certain objects to enter the Earth's atmosphere. These objects are referred to as comets and asteroids, near earth objects(NEO).

1. Problem statement:

The objective of this project is to examine the dataset of near-Earth objects using data-driven analysis. This analysis aids in determining the likelihood of the incoming object, whether it is harmful and enters the earth's atmosphere.

a. When near-Earth objects enter our atmosphere, they could incur significant losses. Additionally, they might seriously harm our infrastructure and endanger our lives. This research enables us to address these issues.

b. Dataset has a variety of fields that enable us to track, detect, and analyze whether an object poses a threat to the earth. We can notify systems and save lives with the use of this knowledge.

2. Dataset:

We have collected our dataset from various sources with rows upto 40,000.
Datasource 1 :
https://www.kaggle.com/datasets/sameepvani/nasa-nearest-earth-objects?resource=download&select=neo.csv

Datasource 2 :
https://www.kaggle.com/code/elnahas/nasa-nearest-earth-objects/notebook

3. Data Cleaning / Preprocessing:

The several data cleaning operations that we have utilized are as follows:

- Missing values:

  Records with missing values can derail the analysis of the data.

```
In [23]: (missing/total_cells)*100
Out[23]: 0.1515909090909091
```

  In the dataset we chose, the percentage of missing values is small. We replaced the missing values with the corresponding column values in the next row.

```
In [24]: # replace missing values with what comes after it in the same column

In [25]: df_NEO['miss_distance'].fillna(method = 'bfill', inplace = True)

In [26]: df_NEO['relative_velocity'].fillna(method = 'bfill', inplace = True)

In [27]: df_NEO['absolute_magnitude'].fillna(method = 'bfill', inplace = True)

In [28]: df_NEO.isnull().sum()

Out[28]: id                       0
         name                     0
         est_diameter_min         0
         est_diameter_max         0
         orbiting_body            0
         sentry_object            0
         Close-Approach (CA) Date 0
         miss_distance            0
         relative_velocity        0
         absolute_magnitude       0
         hazardous                0
         dtype: int64
```

- Scaling:

  Scaling is done to handle the highly varying values or units. It is done to avoid the tendency of machine learning algorithms to weigh greater values, higher and smaller values as the lower values, regardless of the unit of the values.

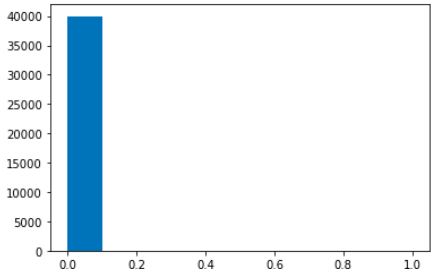
In [32]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data)
scaled_data = pd.DataFrame(scaled_data, columns=['est_diameter_min','relative_velocity','absolute_magnitude'])
scaled_data.head()
```

Out[32]:

	est_diameter_min	relative_velocity	absolute_magnitude
0	0.031607	0.100504	0.355956
1	0.006999	0.564248	0.511153
2	0.019039	0.878488	0.408163
3	0.002531	0.187003	0.615567
4	0.006714	0.325876	0.515425

In [33]:
```
# to detect anomalies in the data, first check if the column is normally distributed
```

In [34]:
```
from matplotlib import pyplot
pyplot.hist(scaled_data['est_diameter_min'])
pyplot.show()
```



- Date formatting:

  Just keeping the necessary data is crucial during the data cleansing procedure. There are values in the Date column of the dataset that are not required for prediction. The dates that were wrongly structured have been removed and replaced with valid dates in an effort to enhance our prediction.

```
from datetime import datetime

dateList = dfNeo['Close-Approach (CA) Date']

for index in range(len(dateList)):
    dateList[index] = datetime.strptime(dateList[index].split()[0], '%Y-%b-%d').date()

dfNeo = dfNeo.drop('Close-Approach (CA) Date',axis=1)
dfNeo['Close-Approach (CA) Date'] = dateList
```

Before:

Close-Approach (CA) Date
1900-Jan-04 22:25 ± 00:02
1900-Jan-11 01:07 ± 00:18
1900-Jan-12 23:07 ± 00:13
1900-Jan-29 18:13 ± 00:24
1900-Feb-04 03:50 ± 14:49

After:

Close-Approach (CA) Date
1900-01-04
1900-01-11
1900-01-12
1900-01-29
1900-02-04

- One hot encoding:

```
import sklearn
import sklearn.preprocessing
from sklearn.preprocessing import OneHotEncoder

encoding = OneHotEncoder()
hazardous_array = encoding.fit_transform(dfNeo[["hazardous"]]).toarray()
hazardous_labels = np.array(encoding.categories_).ravel()
dfNew = pd.concat([dfNeo,pd.DataFrame(hazardous_array,columns=hazardous_labels)],axis = 1)
dfNew
```

The category variable "Hazardous" has been subjected to one hot encoding. It is a process of converting categorical variable into numerical variables so as to improvise the prediction results.

Below is an example:

	id	name	est_diameter_min	est_diameter_max	orbiting_body	sentry_object	hazardous	miss_distance	relative_velocity	absolute_magnitude	Close-Approach (CA) Date	False	True
0	2162635	162635 (2000 SS164)	1.198271	2.679415	Earth	False	False	5.483974e+07	13569.249224	16.73	1900-01-04	1.0	0.0
1	2277475	277475 (2005 WK4)	0.265800	0.594347	Earth	False	True	6.143813e+07	73588.726663	20.00	1900-01-11	0.0	1.0
2	2512244	512244 (2015 YE18)	0.722030	1.614507	Earth	False	False	4.979872e+07	114258.692129	17.83	1900-01-12	1.0	0.0
3	3596030	(2012 BV13)	0.096506	0.215794	Earth	False	False	2.543497e+07	24764.303138	22.20	1900-01-29	1.0	0.0
4	3667127	(2014 GE35)	0.255009	0.570217	Earth	False	True	4.627557e+07	42737.733765	20.09	1900-02-04	0.0	1.0

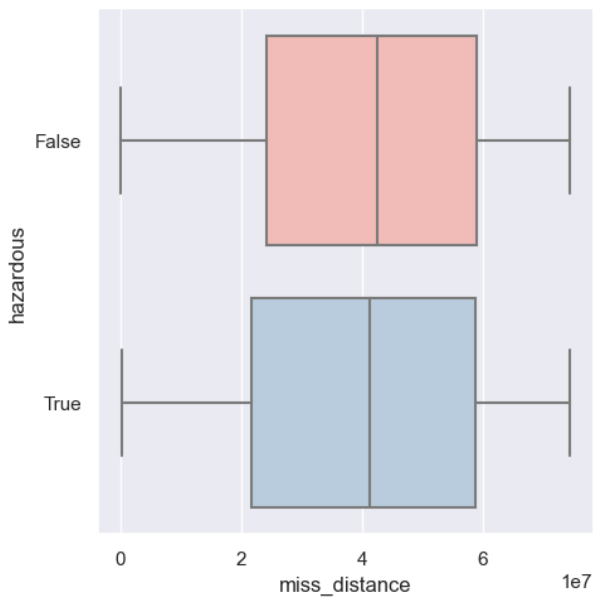
- Duplicate check:

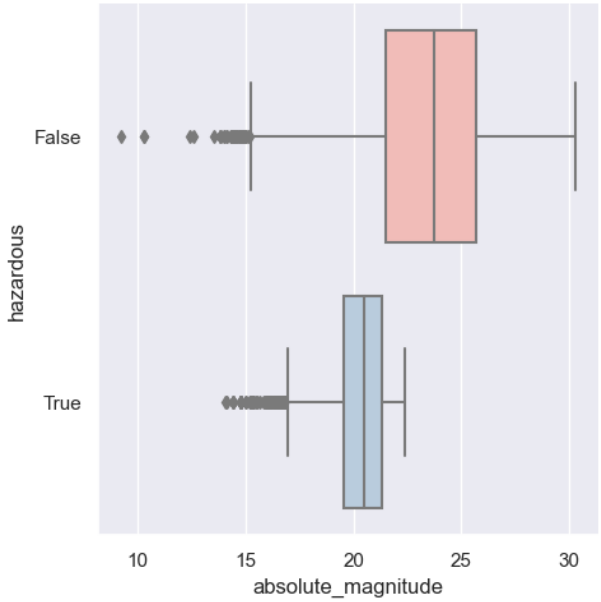
  Duplicated method returns a boolean value if we have any duplicate terms in the dataset. The code gives 0 as output, which means there are no duplicate terms in our dataset.

```
print(dfNeo.duplicated().sum())
✓ 0.8s
0
```

- Detect unwanted outliers:

  Outliers are the data points which differ significantly from other data points in the dataset. Outliers can be due to measurement error, data corruption or occasional extreme events. In the dataset we chose, the source of outliers is not clear. We performed a univariate analysis of data(box plot) to display the outliers.





- Merging datasets:

  Combining the close approach data (from another dataset) with our dataset greatly aids in prediction and also allows us to draw a number of intriguing conclusions.

  Data source of ‘Close approach Date’ : https://www.kaggle.com/code/elnahas/nasa-nearest-earth-objects/notebook

- Dropping unwanted columns:

  The values in the Sentry object column are consistent throughout the dataset’s objects. Therefore, when generating our estimates, this column doesn't really matter

```
dfNew = dfNew.drop('sentry_object', axis=1)
dfNew
```

✓ 0.3s Python

	id	name	est_diameter_min	est_diameter_max	orbiting_body	hazardous	miss_distance	relative_velocity	absolute_magnitude	Close-Approach (CA) Date	False	True	rarity
0	2162635	162635 (2000 SS164)	1.198271	2.679415	Earth	False	5.483974e+07	13569.249224	16.73	1900-01-04	1.0	0.0	2
1	2277475	277475 (2005 WK4)	0.265800	0.594347	Earth	True	6.143813e+07	73588.726663	20.00	1900-01-11	0.0	1.0	1
2	2512244	512244 (2015 YE18)	0.722030	1.614507	Earth	False	4.979872e+07	114258.692129	17.83	1900-01-12	1.0	0.0	2
3	3596030	(2012 BV13)	0.096506	0.215794	Earth	False	2.543497e+07	24764.303138	22.20	1900-01-29	1.0	0.0	1
4	3667127	(2014 GE35)	0.255009	0.570217	Earth	True	4.627557e+07	42737.733765	20.09	1900-02-04	0.0	1.0	1

- Checking unique values :

  Checking for the number of unique values in each column.

```
print(df_NEO.nunique())
```

```
id                        15719
name                      15719
est_diameter_min           1390
est_diameter_max           1390
orbiting_body                 1
sentry_object                 1
Close-Approach (CA) Date  31198
miss_distance             39695
relative_velocity         39690
absolute_magnitude         1359
hazardous                     2
dtype: int64
```

  Orbiting_body and sentry_object has only 1 unique variable.

4. Exploratory Data Analysis:

Examining the general characteristics of data to uncover underlying structure in the data:

	count	mean	std	min	25%	50%	75%	max
id	40000.0	1.129585e+07	1.846201e+07	2.000433e+06	3.299914e+06	3.621612e+06	3.827320e+06	5.427586e+07
est_diameter_min	40000.0	1.482702e-01	3.585850e-01	6.089126e-04	2.140696e-02	6.089126e-02	1.756123e-01	3.789265e+01
est_diameter_max	40000.0	3.315423e-01	8.018205e-01	1.361570e-03	4.786742e-02	1.361570e-01	3.926811e-01	8.473054e+01
miss_distance	39798.0	4.079501e+07	2.098205e+07	1.185167e+04	2.390998e+07	4.237677e+07	5.902371e+07	7.449941e+07
relative_velocity	39695.0	4.938502e+04	2.453765e+04	5.616956e+02	3.014307e+04	4.635959e+04	6.481951e+04	1.299852e+05
absolute_magnitude	39840.0	2.316739e+01	2.924832e+00	9.230000e+00	2.090000e+01	2.320000e+01	2.540000e+01	3.030000e+01

We observe that standard deviation for columns est_diameter_min and est_diameter_max is greater than other columns indicating that spread of data for these columns is higher than other columns.

We observe that est_diameter_min, est_diameter_max have deferred maximum value when compared to the 75% of values this indicates that there are significant outliers in the data.

Analyzing if each column is left skewed or right skewed:

We calculated mean and median of 3 columns and observed the following: Mean of est_diameter_min is 0.148 and median is 0.06 (mean > median) therefore this is a right skewed distribution. This also indicates that most of the detected NEOs have a smaller estimated diameter than the mean and therefore they are not hazardous(good news!!!)

Out[90]:

	count	mean	std	min	25%	50%	75%	max
id	40000.0	1.129585e+07	1.846201e+07	2.000433e+06	3.299914e+06	3.621612e+06	3.827320e+06	5.427586e+07
est_diameter_min	40000.0	1.482702e-01	3.585850e-01	6.089126e-04	2.140696e-02	6.089126e-02	1.756123e-01	3.789265e+01
est_diameter_max	40000.0	3.315423e-01	8.018205e-01	1.361570e-03	4.786742e-02	1.361570e-01	3.926811e-01	8.473054e+01
miss_distance	40000.0	4.080261e+07	2.097845e+07	1.185167e+04	2.392831e+07	4.238516e+07	5.902303e+07	7.449941e+07
relative_velocity	40000.0	4.938134e+04	2.454829e+04	5.616956e+02	3.013684e+04	4.635753e+04	6.481940e+04	1.299852e+05
absolute_magnitude	40000.0	2.316766e+01	2.926833e+00	9.230000e+00	2.090000e+01	2.320000e+01	2.540000e+01	3.030000e+01

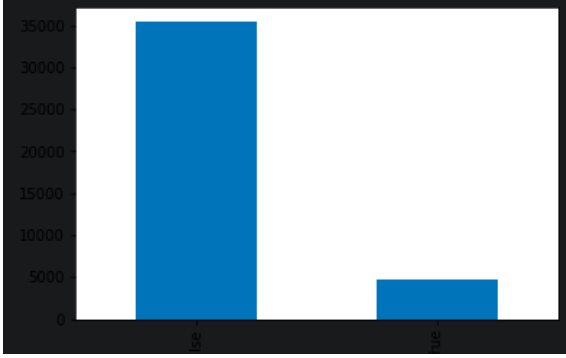
```
In [92]: df_NEO['est_diameter_min'].median()
Out[92]: 0.0608912622

In [93]: df_NEO['est_diameter_max'].median()
Out[93]: 0.1361570015

In [94]: df_NEO['miss_distance'].median()
Out[94]: 42385162.38054554

In [95]: df_NEO['absolute_magnitude'].median()
Out[95]: 23.2
```

Mean of miss_distance is 40802610 and median is 42385162 (mean ~ median) therefore this is normally distributed about the mean.
Mean of absolute_magnitude is 23.1 and median is 23.2, this feature is normally distributed.

Analyzing the Target Variable:

Trying to analyze the target variable. We observe that the target variable ‘hazardous’ has 88.4% False and 11.5% true values. This indicates of all the NEO orbiting earth every year only about 11% are hazardous.

```
df_NEO['hazardous'].value_counts().plot.bar()
<AxesSubplot:>
```



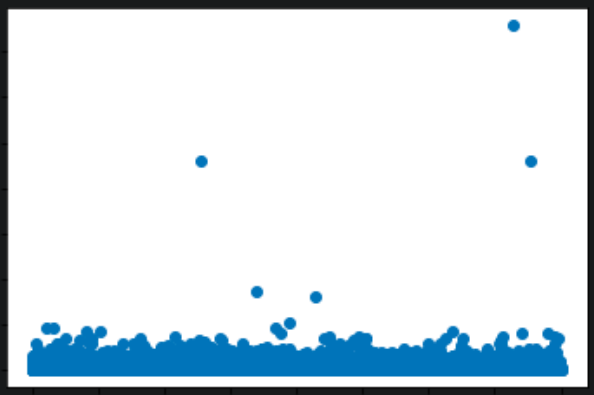
Univariate Graphical analysis:

Performing univariate analysis to comprehend descriptive summary and enumerative properties. This is performed to maximize insights into a dataset as mentioned in NIST publication.

Drawing a univariate scatter plot for est_diameter_min and est_diameter_max of the dataset:

```
plt.scatter(df_NEO.index, df_NEO['est_diameter_min'])
```

```
<matplotlib.collections.PathCollection at 0x7fb019c265b0>
```




```
plt.scatter(df_NEO.index, df_NEO['est_diameter_max'])
```

```
<matplotlib.collections.PathCollection at 0x7fb01a8e2fa0>
```

In the above scatter plots we observe that there are outliers in the data, these outliers must be detected and dealt with to avoid skewing the output when we apply a model.

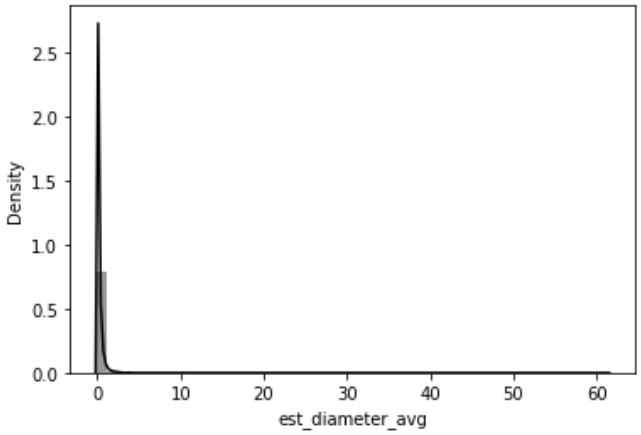
```
plt.scatter(df_NEO.index, df_NEO['relative_velocity'])
<matplotlib.collections.PathCollection at 0x7fb01a8c1e50>
```

This scatterplot is for relative_velocity, there are no outliers in the column relative_velocity.

Histogram:

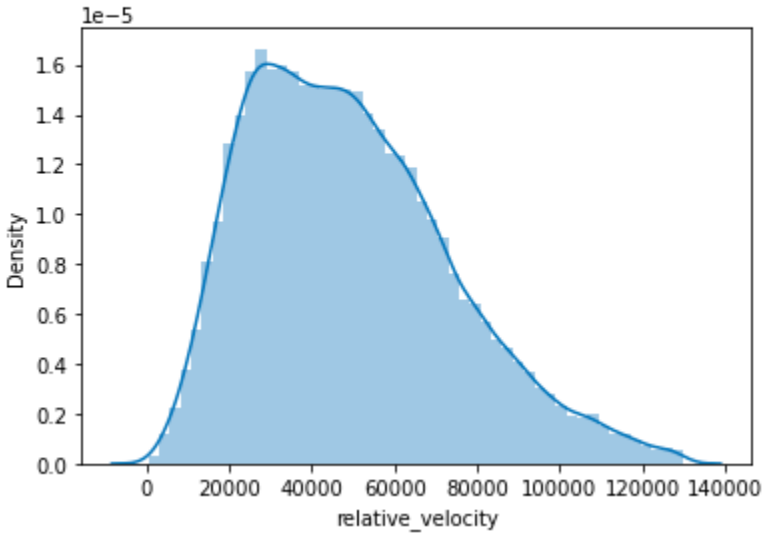
Here we observe that est_diameter_avg is mostly between 0 and 5 but there are a few asteroids with higher est_diameter_avg.

```
sns.distplot(df_NEO.est_diameter_avg,color='black')
plt.show()
```



Relative velocity mostly lies between 0 and 120000.

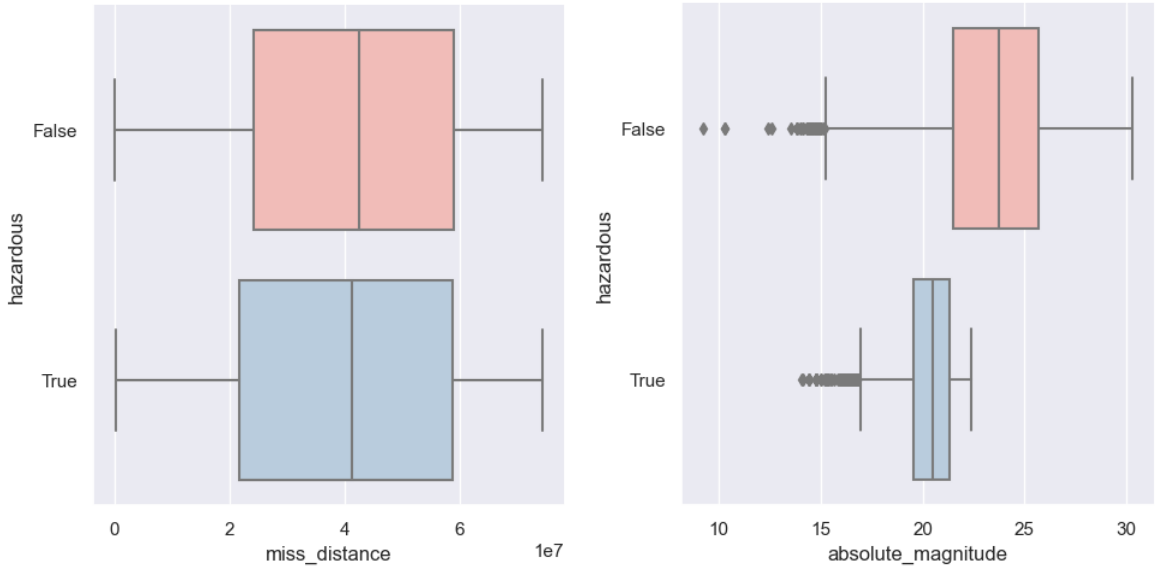
```
sns.distplot(df_NEO.relative_velocity)
plt.show()
```



Bivariate analysis:

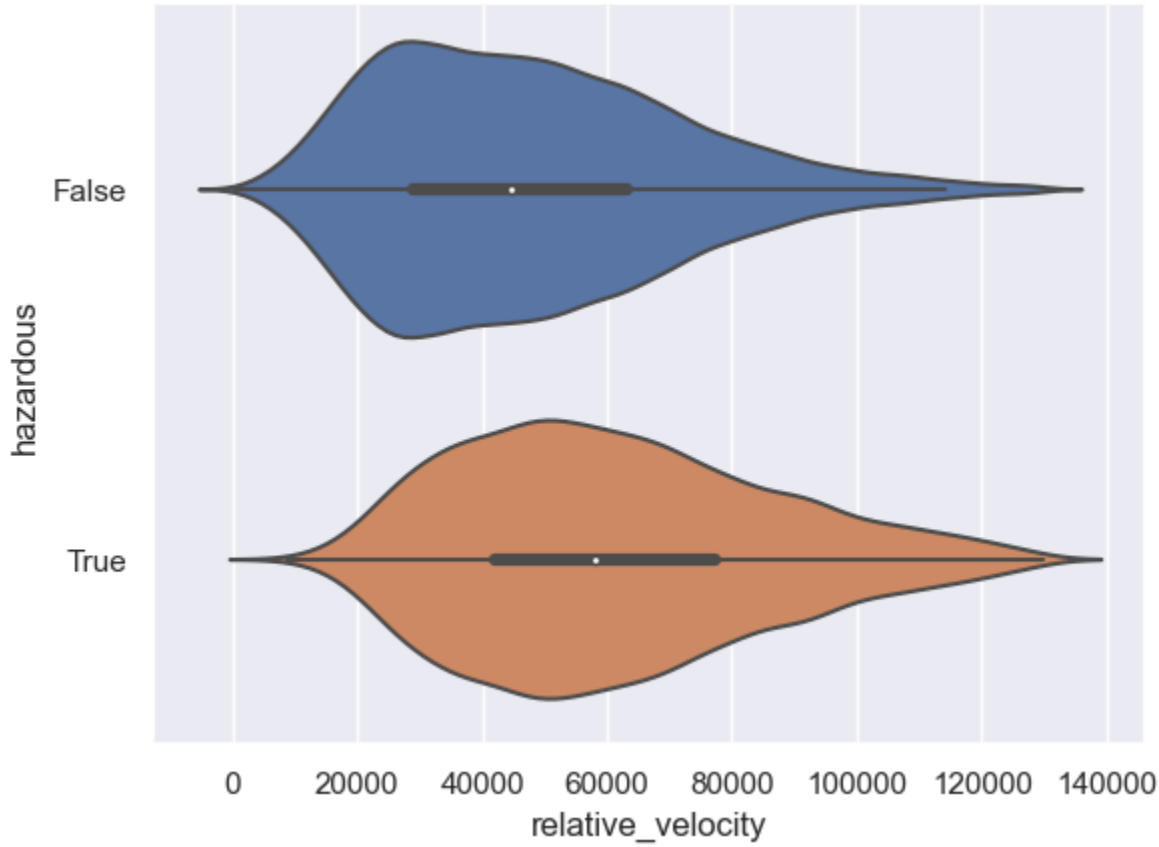
Box plot:
Here we have performed bivariate analysis to both miss-distance and absolute_magnitude. The results are very intuitive, graph shows that an object is less dangerous if its absolute magnitude and miss distance values are larger. As a result, the outcome holds true for both miss distance and absolute magnitude. Taking just one into account is enough.

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="darkgrid")

categoryList = ['miss_distance','absolute_magnitude']
for category in categoryList:
    sns.catplot(
        data=dfNew, x=category, y="hazardous", kind="box",orient='h', palette="Pastel1"
    )
plt.show()
```



Violin Plot:

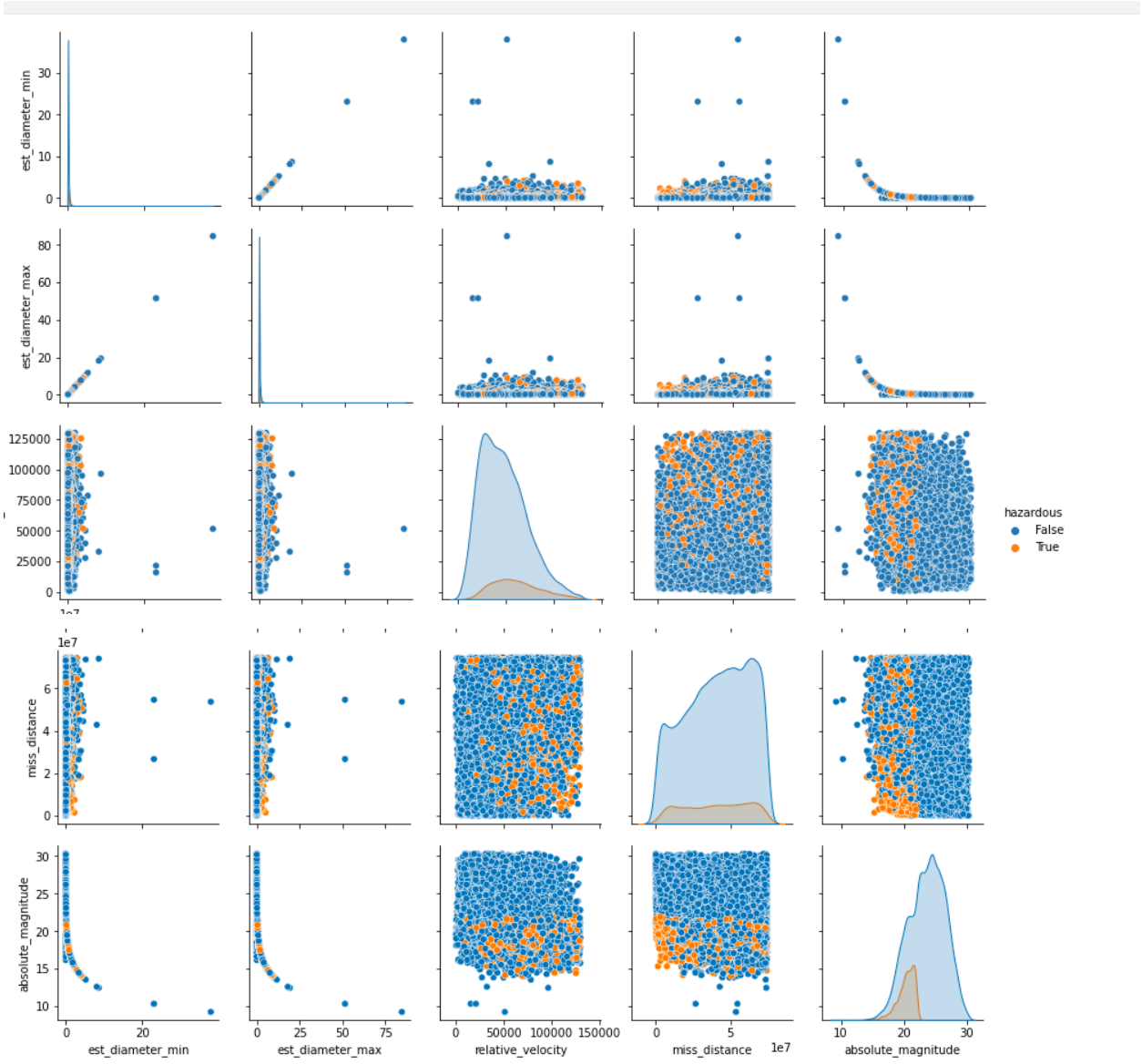
```
sns.violinplot(x ='relative_velocity', y='hazardous',data = dfNew, orient='h')
plt.show()
```



The white spot in the graph denoted the median. From the above graph, we can say that the hazardous objects are more likely to move faster than non-hazardous ones.

Pairwise Plots:

In this we are plotting pairplots by passing ‘hazardous’ as a parameter. Therefore we are using ‘hazardous’ column to derive a relationship with other columns. We observe that scatterplots and histograms are plotted. Scatter plot is the default pair plot. Histograms are drawn when there is same variable on both X and Y axis.



Correlation analysis and correlation heatmap

We performed correlation analysis on numerical features of the dataset. We observed that there is a high correlation between est_diameter_min, est_diameter_max and relative velocity. There is a negative correlation between relative_velocity and absolute magnitude. This indicates that if absolute magnitude increases relative velocity decreases.

```
In [105]: numeric_features.corr()
```

Out[105]:

	id	est_diameter_min	est_diameter_max	miss_distance	relative_velocity	absolute_magnitude
id	1.000000	-0.124501	-0.124501	0.065902	-0.014238	0.258618
est_diameter_min	-0.124501	1.000000	1.000000	0.090417	0.174138	-0.525366
est_diameter_max	-0.124501	1.000000	1.000000	0.090417	0.174138	-0.525366
miss_distance	0.065902	0.090417	0.090417	1.000000	0.309900	-0.164686
relative_velocity	-0.014238	0.174138	0.174138	0.309900	1.000000	-0.315899
absolute_magnitude	0.258618	-0.525366	-0.525366	-0.164686	-0.315899	1.000000

```
Out[106]: <AxesSubplot:>
```