```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv("/content/drive/MyDrive/nearest-earth-objects(1910-
2024).csv")

data.head()
```

```
{"type":"dataframe","variable_name":"data"}
```

```
data.describe(include='all')
```

```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 11,\n  \"fields\":
[\n    {\n      \"column\": \"neo_id\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 22716782.698697973,\n
\"min\": 338199.0,\n        \"max\": 54462807.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
17599388.682018574,\n          3742127.0,\n          338199.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"name\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
4,\n        \"samples\": [\n          33514,\n          \"211\",\n
\"338199\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"absolute_magnitude\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 119554.56863782684,\n        \"min\":
2.911216390292293,\n        \"max\": 338171.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
22.93252495926617,\n          22.8,\n          338171.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"estimated_diameter_min\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
119559.56966825192,\n        \"min\": 0.0005111578,\n        \"max\":
338171.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n
0.1578120466605549,\n          0.0732073989,\n          338171.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"estimated_diameter_max\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
119557.18111686174,\n        \"min\": 0.0011429835,\n        \"max\":
338171.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n
0.3528784640005492,\n          0.1636967205,\n          338171.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"orbiting_body\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"338199\",\n          1,\n          \"Earth\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"relative_velocity\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
130715.5645630245,\n        \"min\": 203.34643253,\n        \"max\":
338199.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n
51060.662907595586,\n          47560.4654744848,\n          338199.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"miss_distance\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
26699262.28644739,\n        \"min\": 6745.532515957,\n        \"max\":
74798651.4521972,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          41535350.93219019,\n
43326743.82834823,\n          338199.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"is_hazardous\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          2,\n
\"295037\",\n          \"338199\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```

```python
data.info()
data.isnull().sum()
data = data.drop(['neo_id','name','orbiting_body'],axis = 1)
data.head()
```

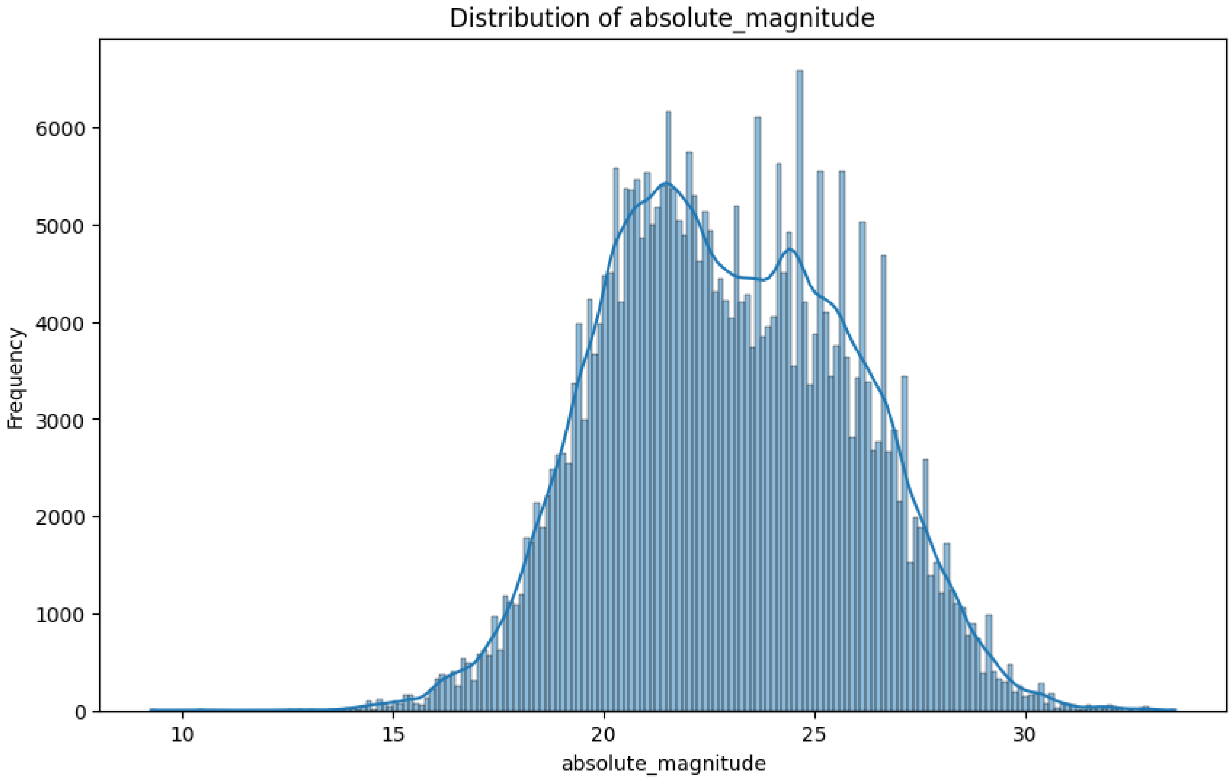
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 338199 entries, 0 to 338198
Data columns (total 9 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   neo_id                  338199 non-null  int64  
 1   name                    338199 non-null  object 
 2   absolute_magnitude      338171 non-null  float64
 3   estimated_diameter_min  338171 non-null  float64
 4   estimated_diameter_max  338171 non-null  float64
 5   orbiting_body           338199 non-null  object 
 6   relative_velocity       338199 non-null  float64
 7   miss_distance           338199 non-null  float64
 8   is_hazardous            338199 non-null  bool   
dtypes: bool(1), float64(5), int64(1), object(2)
memory usage: 21.0+ MB
```

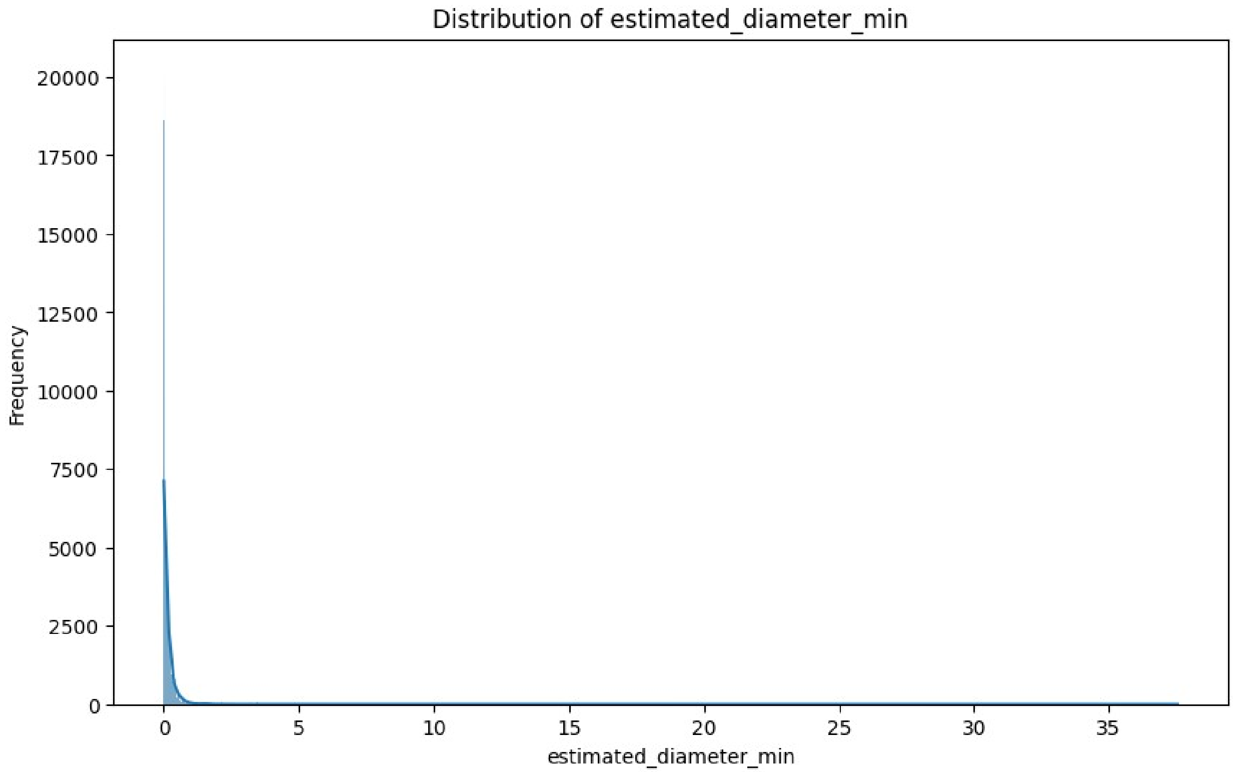
```
{"type":"dataframe","variable_name":"data"}
```

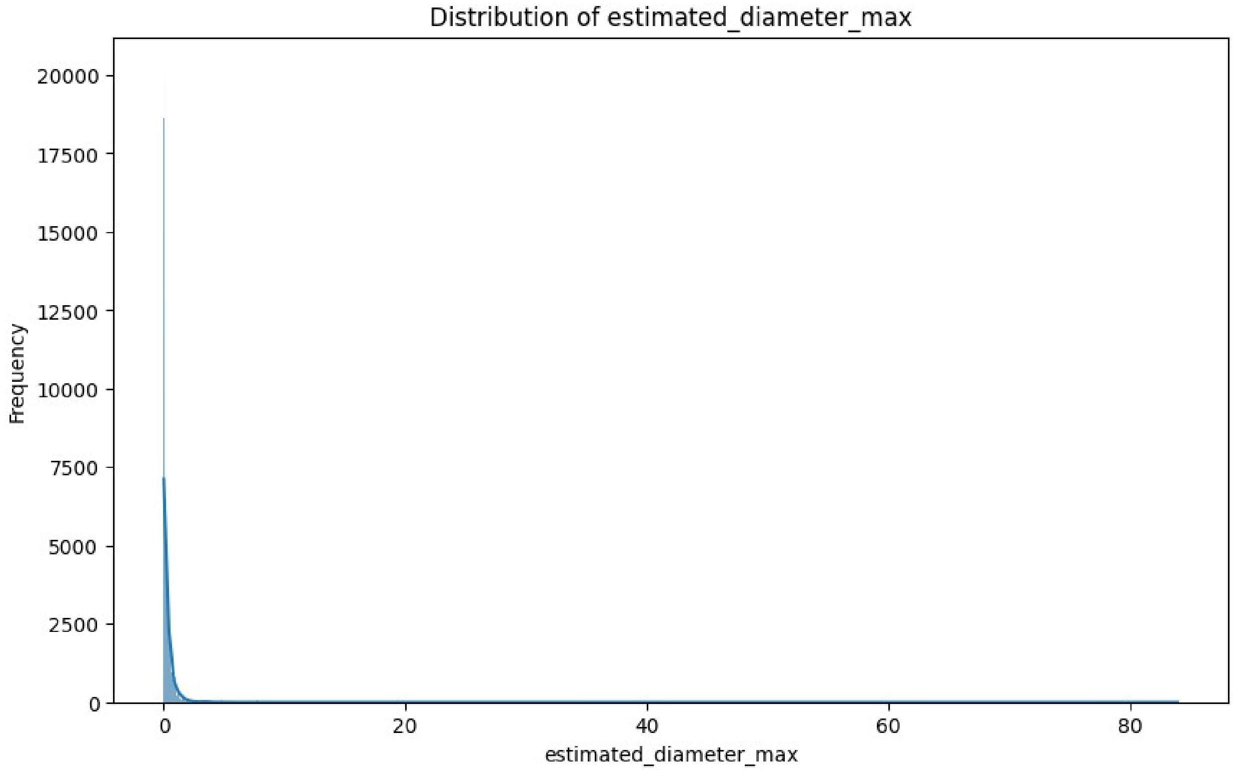
```python
data['is_hazardous'] = data['is_hazardous'].map({True:1 , False:0})
data.head()
```

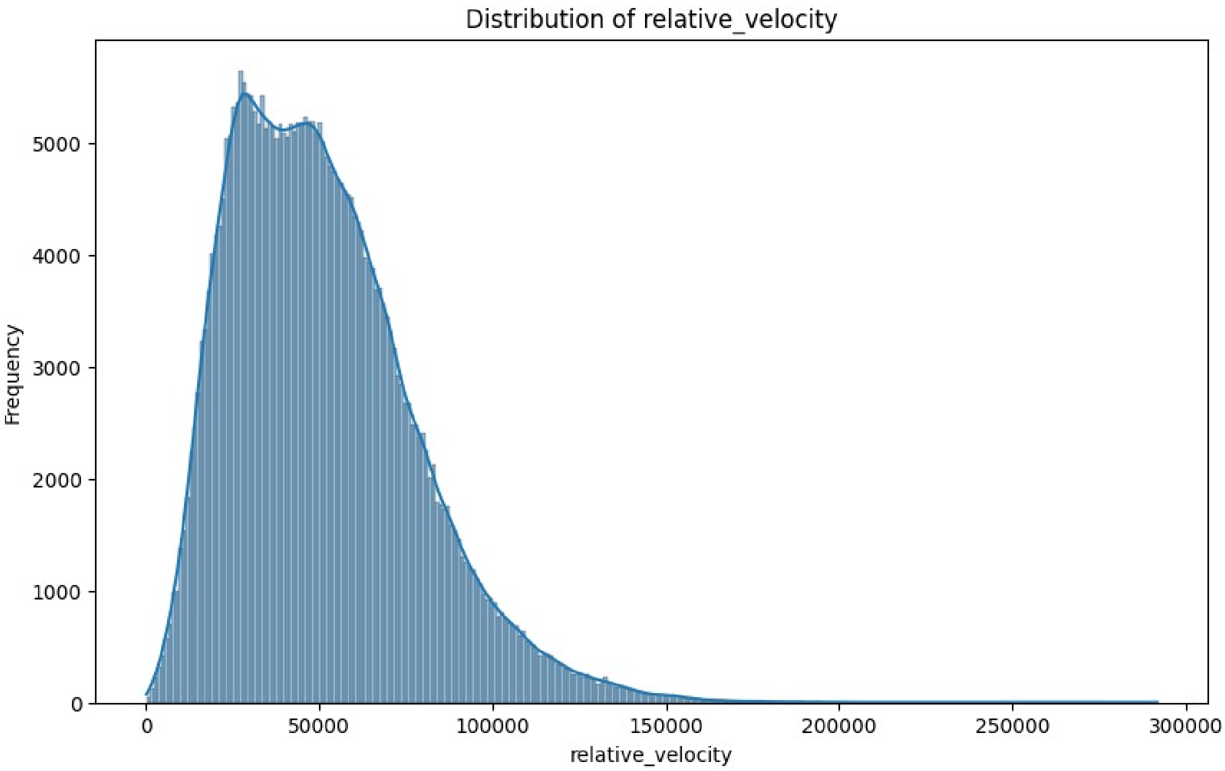
```
{"type":"dataframe","variable_name":"data"}
```

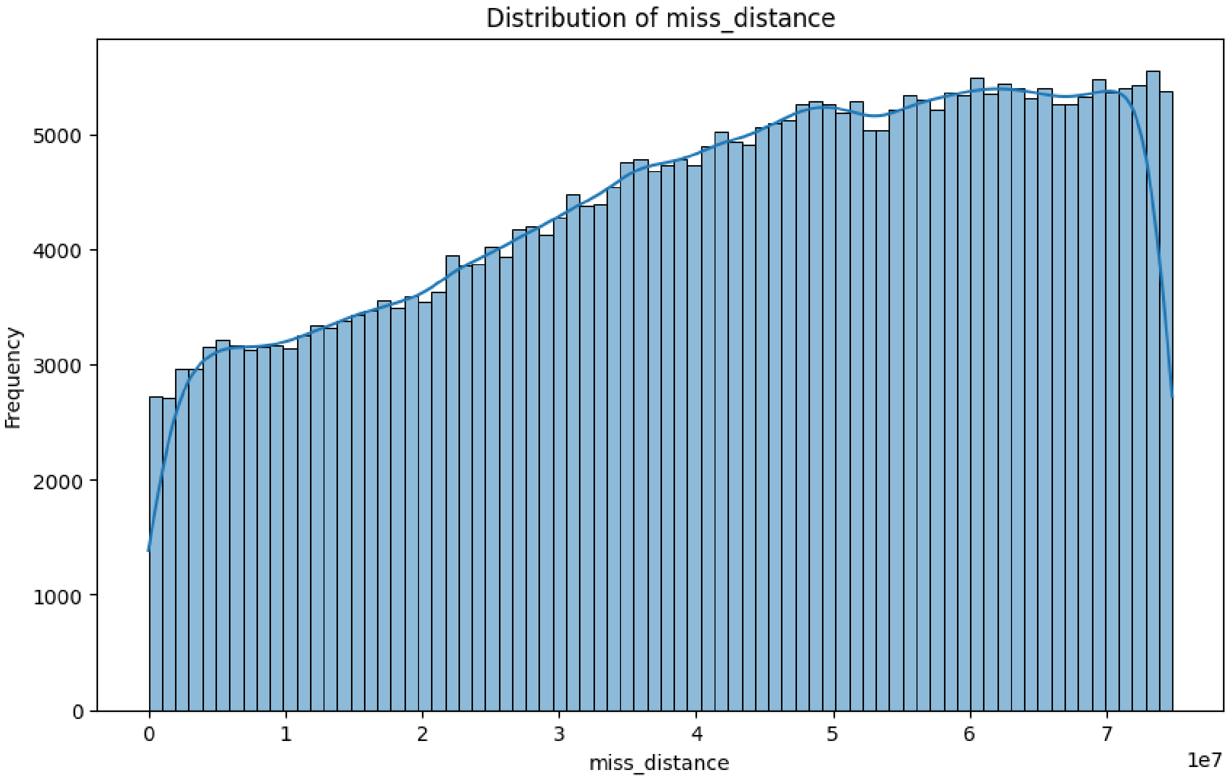
```python
columns = data.columns

# Plot each column
for col in columns:
    plt.figure(figsize=(10, 6))
    sns.histplot(data[col], kde=True)
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.show()

data = data.dropna()
```

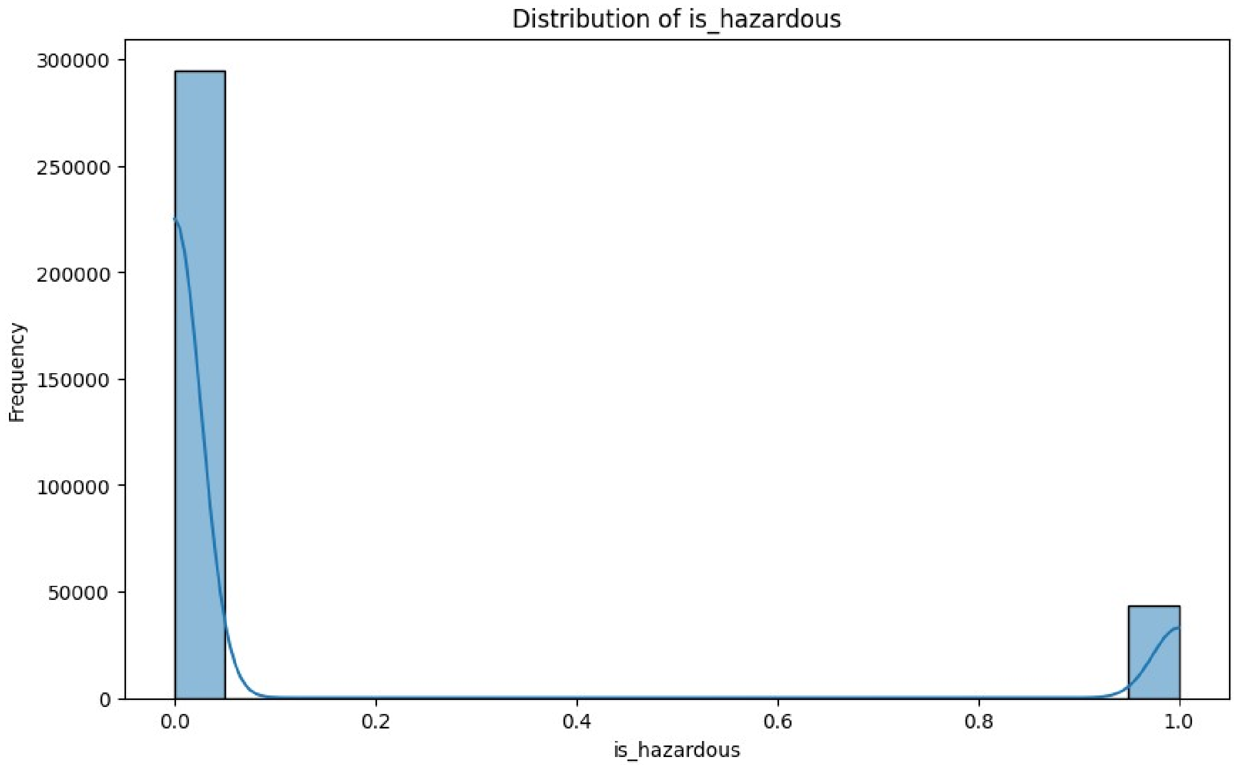



Distribution of estimated_diameter_min
Frequency
20000
17500
15000
12500
10000
7500
5000
2500
0
0
5
10
15
20
25
30
35
estimated_diameter_min


Distribution of estimated_diameter_max
Frequency
20000
17500
15000
12500
10000
7500
5000
2500
0
0
20
40
60
80
estimated_diameter_max


Distribution of relative_velocity
Frequency
0
1000
2000
3000
4000
5000
0
50000
100000
150000
200000
250000
300000
relative_velocity


Distribution of miss_distance
Frequency
0
1000
2000
3000
4000
5000
0
1
2
3
4
5
6
7
1e7
miss_distance



```python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x = data.drop('is_hazardous',axis = 1)
y = data['is_hazardous']

x_scaled = scaler.fit_transform(x)
print(x_scaled)

print(x)
print(y)
```

```
[[-1.30273056  0.75553115  0.75553115  0.78354069  0.79946485]
 [-1.52257027  1.18683296  1.18683296  2.23068176  0.68670052]
 [-0.50924664 -0.06847576 -0.06847576 -0.9922207   1.2357423 ]
 ...
 [-0.34814535 -0.15283782 -0.15283782 -1.48591476  0.5740475 ]
 [ 0.32786173 -0.36139181 -0.36139181  0.19463588 -1.7498115 ]
 [ 0.00634617 -0.28520788 -0.28520788 -0.34089772 -1.65633252]]
        absolute_magnitude  estimated_diameter_min
estimated_diameter_max  \
0                   19.140                0.394962
0.883161
1                   18.500                0.530341
1.185878
2                   21.450                0.136319
0.304818
```

```
3                   20.630                   0.198863
0.444672
4                   22.700                   0.076658
0.171412
...                    ...                        ...
...
338194              28.580                   0.005112
0.011430
338195              28.690                   0.004859
0.010865
338196              21.919                   0.109839
0.245607
338197              23.887                   0.044377
0.099229
338198              22.951                   0.068290
0.152700

        relative_velocity  miss_distance
0            71745.401048   5.814362e+07
1           109949.757148   5.580105e+07
2            24865.506798   6.720689e+07
3            78890.076805   3.039644e+07
4            56036.519484   6.311863e+07
...                   ...            ...
338194       56646.985988   6.406548e+07
338195       21130.768947   2.948883e+07
338196       11832.041031   5.346078e+07
338197       56198.382733   5.184742e+06
338198       42060.357830   7.126682e+06

[338171 rows x 5 columns]
0         0
1         1
2         0
3         0
4         0
         ..
338194    0
338195    0
338196    0
338197    0
338198    0
Name: is_hazardous, Length: 338171, dtype: int64
```

```python
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(x_scaled,y,test_size
= 0.2 , random_state = 42)
print(X_train)
```

```
[[-1.45387036  1.03816614  1.03816614  2.94555792  0.99482004]
 [-0.06269724 -0.26410792 -0.26410792 -0.46793005 -1.1647807 ]
 [ 1.34908585 -0.46681368 -0.46681368 -0.82639138  0.31651595]
 ...
 [-0.49894165 -0.07443451 -0.07443451 -1.06662046  1.26623063]
 [ 1.56892556 -0.47599238 -0.47599238 -1.10754607  0.68708169]
 [ 1.39717579 -0.46905877 -0.46905877  0.32777902  1.30049185]]
```

```
from sklearn.neighbors import KNeighborsClassifier
Knn = KNeighborsClassifier(n_neighbors=10)
Knn.fit(X_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=10)
```

```
y_pred = Knn.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8851630073186959
```