In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from IPython import get_ipython
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
data = pd.read_csv('nearest_earth_objects.csv')
```

In [3]:

```
data.head()
```

Out[3]:

	id	name	est_diameter_min	est_diameter_max	relative_velocity	miss_distance	orbit
0	2162635	162635 (2000 SS164)	1.198271	2.679415	13569.24922	54839744.08	
1	2277475	277475 (2005 WK4)	0.265800	0.594347	73588.72666	61438126.52	
2	2512244	512244 (2015 YE18)	0.722030	1.614507	114258.69210	49798724.94	
3	3596030	(2012 BV13)	0.096506	0.215794	24764.30314	25434972.72	
4	3667127	(2014 GE35)	0.255009	0.570217	42737.73376	46275567.00	

In [4]:

```
data.tail()
```

Out[4]:

	id	name	est_diameter_min	est_diameter_max	relative_velocity	miss_distance
90831	3763337	(2016 VX1)	0.026580	0.059435	52078.886690	12300389.18
90832	3837603	(2019 AD3)	0.016771	0.037501	46114.605070	54321206.42
90833	54017201	(2020 JP3)	0.031956	0.071456	7566.807732	28400768.16
90834	54115824	(2021 CN5)	0.007321	0.016370	69199.154480	68692060.53
90835	54205447	(2021 TW7)	0.039862	0.089133	27024.455550	59772130.59

In [5]:

```
data.duplicated().sum()
```

Out[5]:

```
0
```

In [6]:

```
data.isnull().sum()
```

Out[6]:

```
id                    0
name                  0
est_diameter_min      0
est_diameter_max      0
relative_velocity     0
miss_distance         0
orbiting_body         0
sentry_object         0
absolute_magnitude    0
hazardous             0
dtype: int64
```

In [7]:

```
data.shape
```

Out[7]:

```
(90836, 10)
```

In [8]:

```
data.columns
```

Out[8]:

```
Index(['id', 'name', 'est_diameter_min', 'est_diameter_max',
       'relative_velocity', 'miss_distance', 'orbiting_body', 'sentry_obje
ct',
       'absolute_magnitude', 'hazardous'],
      dtype='object')
```

In [9]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90836 entries, 0 to 90835
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   id                  90836 non-null  int64  
 1   name                90836 non-null  object 
 2   est_diameter_min    90836 non-null  float64
 3   est_diameter_max    90836 non-null  float64
 4   relative_velocity   90836 non-null  float64
 5   miss_distance       90836 non-null  float64
 6   orbiting_body       90836 non-null  object 
 7   sentry_object       90836 non-null  bool   
 8   absolute_magnitude  90836 non-null  float64
 9   hazardous           90836 non-null  bool   
dtypes: bool(2), float64(5), int64(1), object(2)
memory usage: 5.7+ MB
```

In [10]:

```
data.describe()
```

Out[10]:

	id	est_diameter_min	est_diameter_max	relative_velocity	miss_distance	abs
count	9.083600e+04	90836.000000	90836.000000	90836.000000	9.083600e+04	
mean	1.438288e+07	0.127432	0.284947	48066.918918	3.706655e+07	
std	2.087202e+07	0.298511	0.667491	25293.296961	2.235204e+07	
min	2.000433e+06	0.000609	0.001362	203.346432	6.745533e+03	
25%	3.448110e+06	0.019256	0.043057	28619.020648	1.721082e+07	
50%	3.748362e+06	0.048368	0.108153	44190.117890	3.784658e+07	
75%	3.884023e+06	0.143402	0.320656	62923.604635	5.654900e+07	
max	5.427591e+07	37.892650	84.730541	236990.128100	7.479865e+07	

In [11]:

```
data.nunique()
```

Out[11]:

```
id                    27423
name                  27423
est_diameter_min       1638
est_diameter_max       1638
relative_velocity     90785
miss_distance         90536
orbiting_body             1
sentry_object             1
absolute_magnitude     1638
hazardous                 2
dtype: int64
```

In [12]:

```
data['hazardous'].unique()
```

Out[12]:

```
array([False,  True])
```

In [13]:

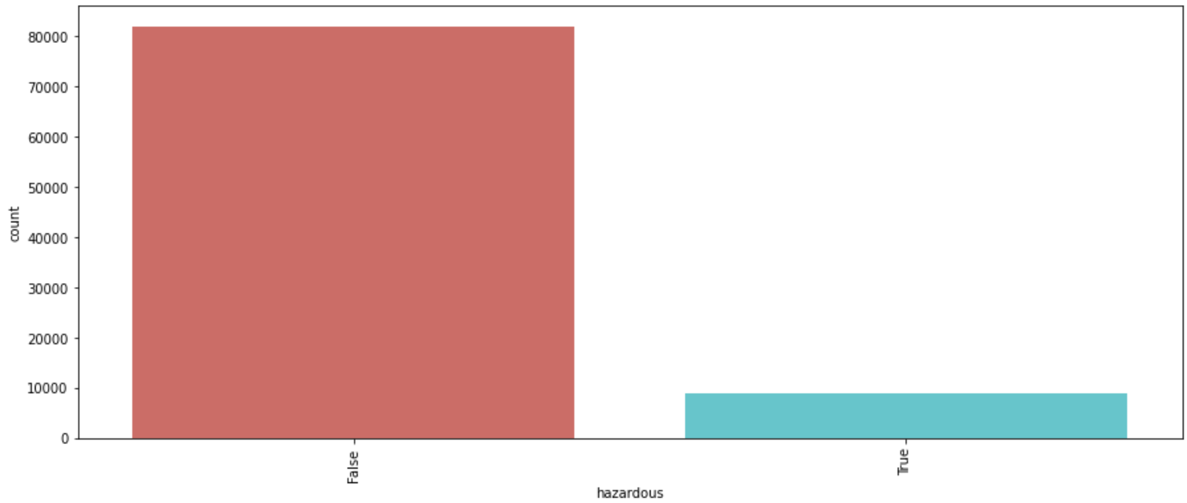
```
data['hazardous'].value_counts()
```

Out[13]:

```
False    81996
True      8840
Name: hazardous, dtype: int64
```

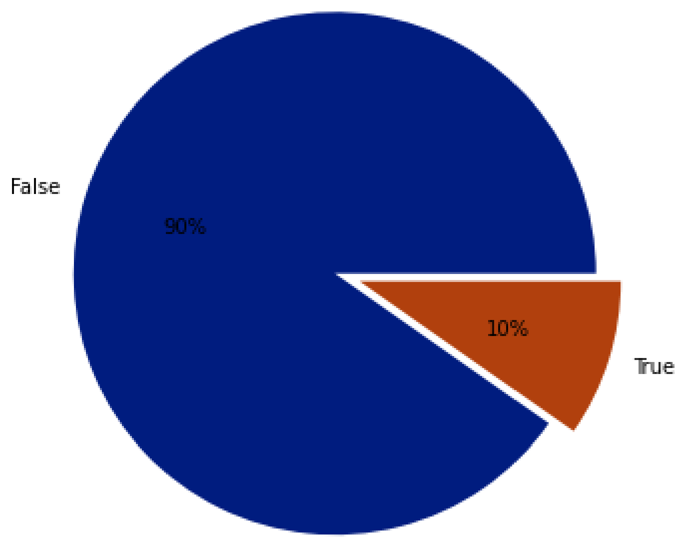
In [14]:

```
plt.figure(figsize=(15,6))
sns.countplot('hazardous', data = data, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



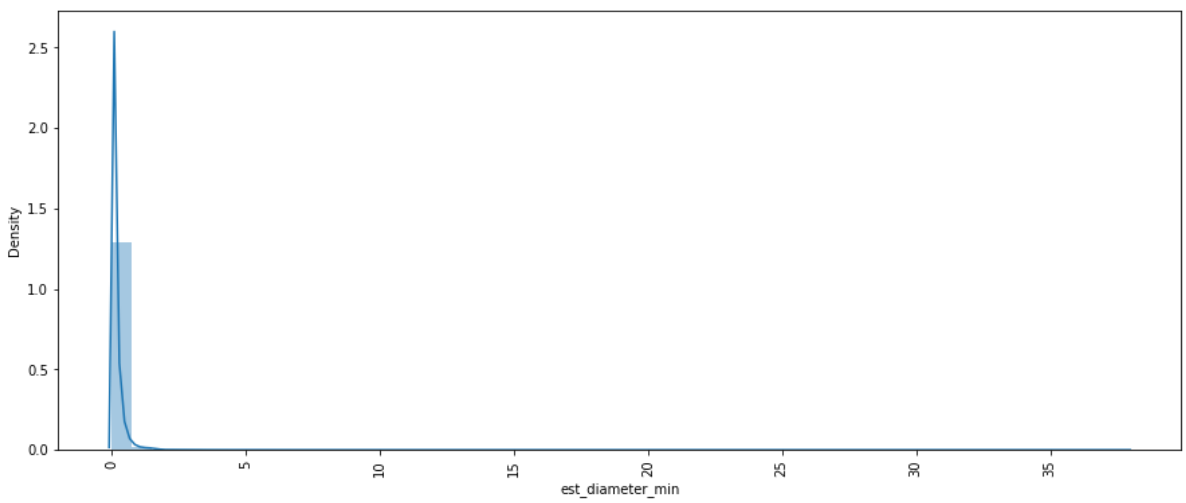
In [16]:

```
plt.figure(figsize=(15,6))
palette_color = sns.color_palette('dark')
explode = [0, 0.1]
plt.pie(data['hazardous'].value_counts(), labels= ['False', 'True'],
        colors=palette_color, explode=explode,
        autopct='%.0f%%')
plt.show()
```



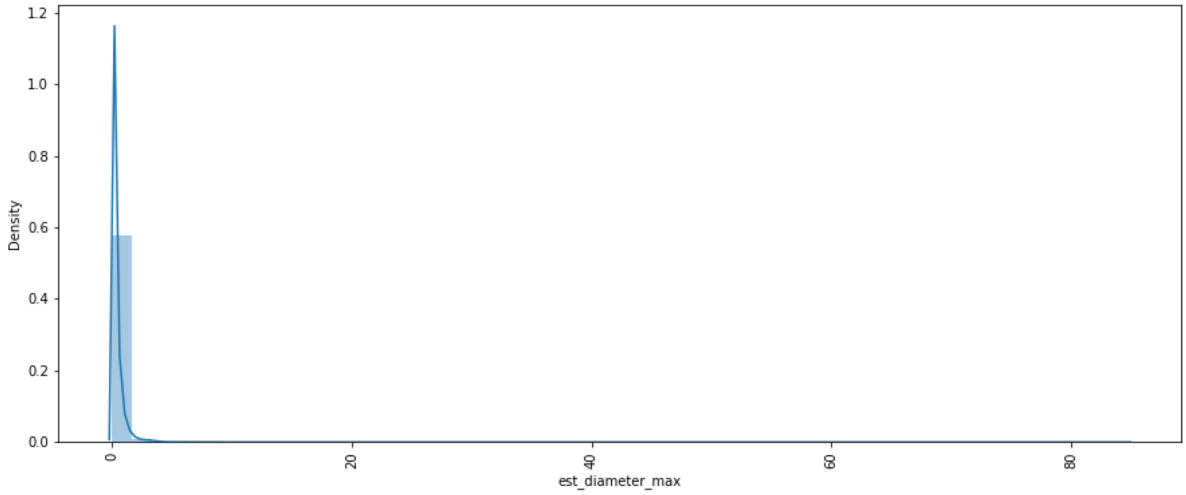
In [17]:

```
plt.figure(figsize=(15,6))
sns.distplot(data['est_diameter_min'])
plt.xticks(rotation = 90)
plt.show()
```



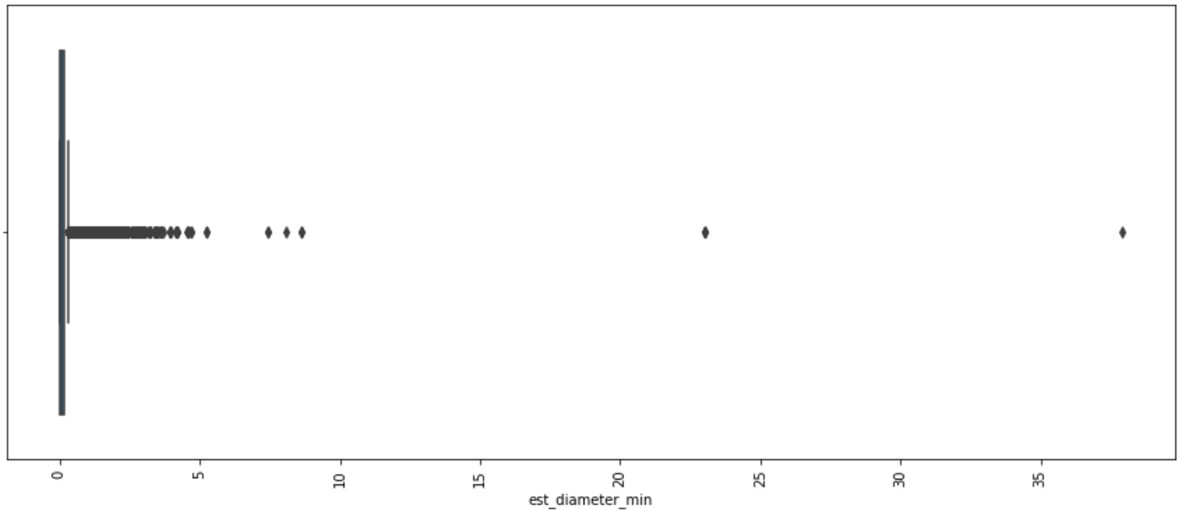
In [18]:

```
plt.figure(figsize=(15,6))
sns.distplot(data['est_diameter_max'])
plt.xticks(rotation = 90)
plt.show()
```



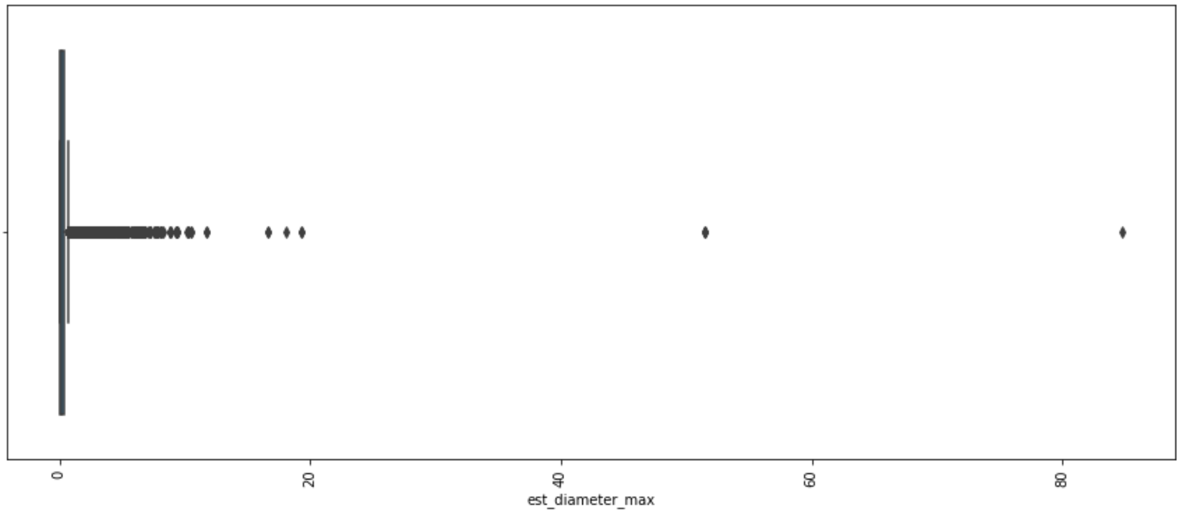
In [19]:

```
plt.figure(figsize=(15,6))
sns.boxplot(data['est_diameter_min'])
plt.xticks(rotation = 90)
plt.show()
```



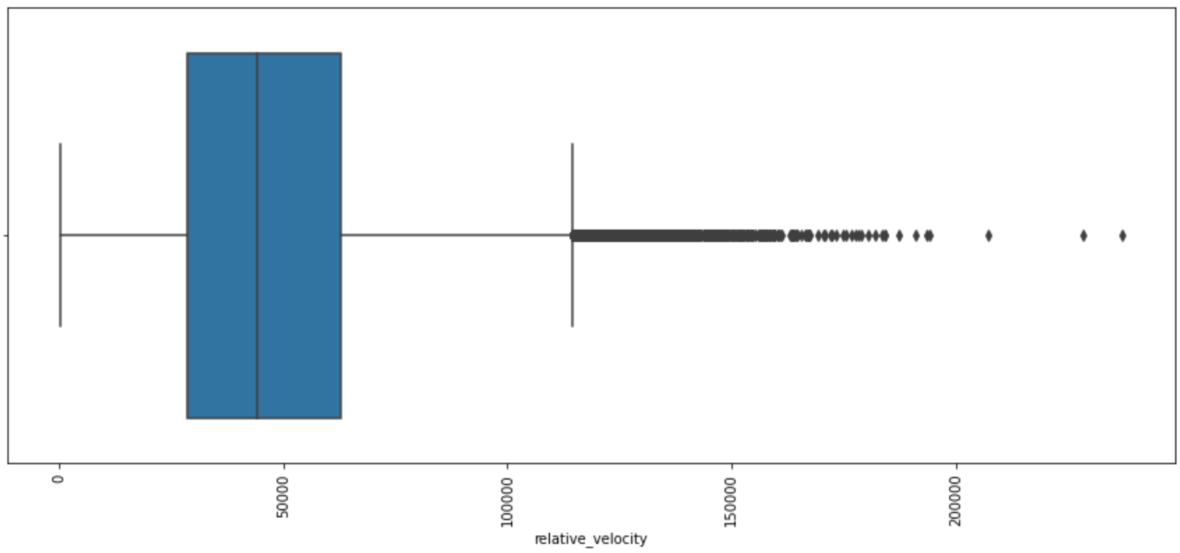
In [20]:

```
plt.figure(figsize=(15,6))
sns.boxplot(data['est_diameter_max'])
plt.xticks(rotation = 90)
plt.show()
```



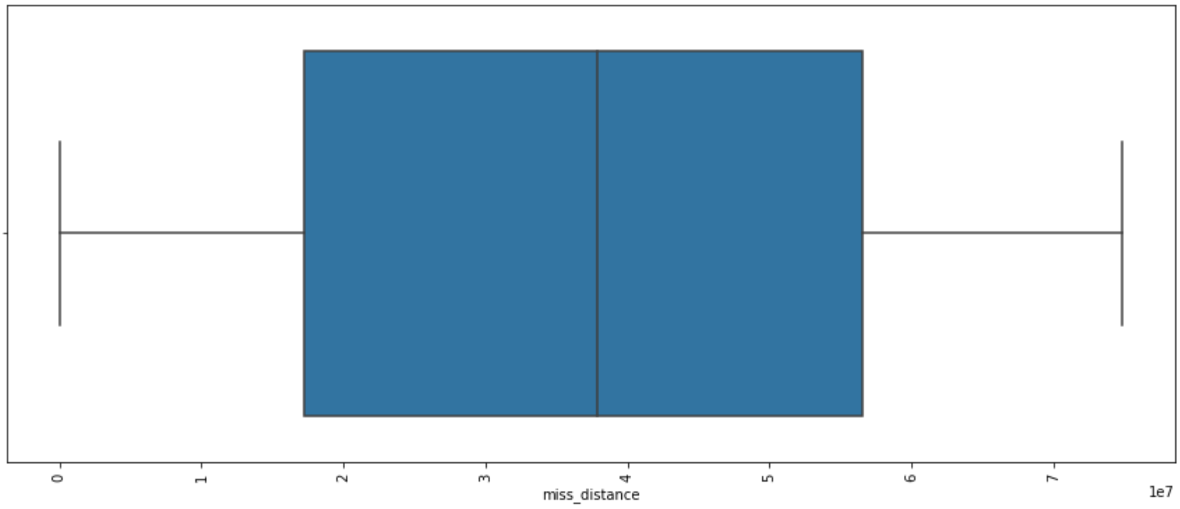
In [21]:

```
plt.figure(figsize=(15,6))
sns.boxplot(data['relative_velocity'])
plt.xticks(rotation = 90)
plt.show()
```



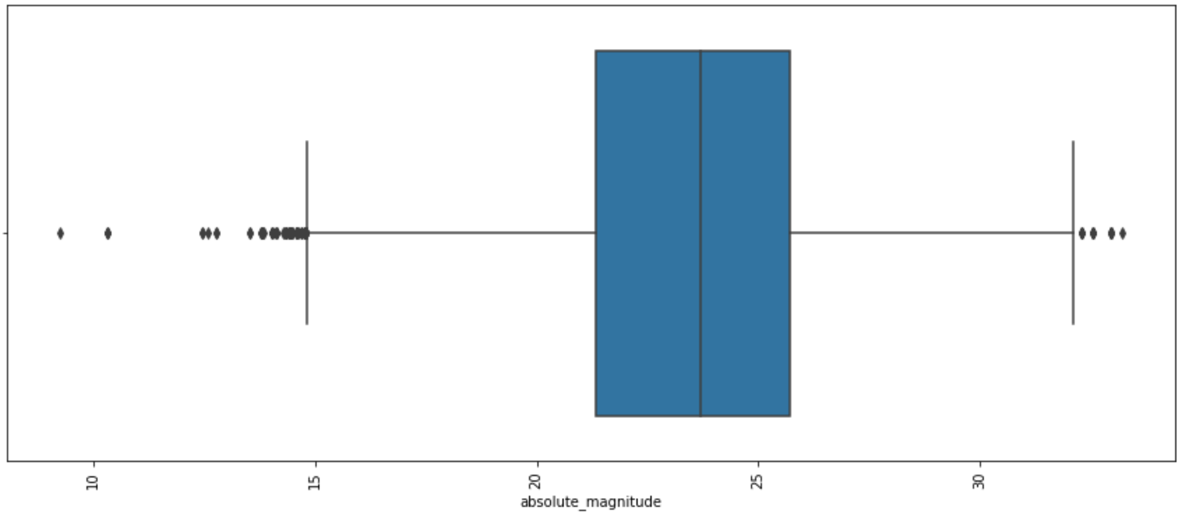
In [22]:

```
plt.figure(figsize=(15,6))
sns.boxplot(data['miss_distance'])
plt.xticks(rotation = 90)
plt.show()
```



In [23]:

```
plt.figure(figsize=(15,6))
sns.boxplot(data['absolute_magnitude'])
plt.xticks(rotation = 90)
plt.show()
```



In [26]:

```
from contextlib import contextmanager
from time import time
from tqdm import tqdm
import lightgbm as lgbm
import category_encoders as ce

from tensorflow.keras.utils import to_categorical
from sklearn.metrics import classification_report, log_loss, accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import KFold
```

In [27]:

```
from sklearn.preprocessing import LabelEncoder

def labelencoder(df):
    for c in df.columns:
        if df[c].dtype=='object':
            df[c] = df[c].fillna('N')
            lbl = LabelEncoder()
            lbl.fit(list(df[c].values))
            df[c] = lbl.transform(df[c].values)
    return df
```

In [28]:

```
data1=labelencoder(data)
```

In [30]:

```
import random
```

In [31]:

```
m=len(data1)
M=list(range(m))
random.seed(2021)
random.shuffle(M)
```

In [39]:

```
target=['hazardous']
dataY=data1[target]
dataX=data1.drop(target,axis=1)
```

In [40]:

```
dataX.shape
```

Out[40]:

```
(90836, 9)
```

In [41]:

```
dataY.shape
```

Out[41]:

```
(90836, 1)
```

In [42]:

```
dataX.columns
```

Out[42]:

```
Index(['id', 'name', 'est_diameter_min', 'est_diameter_max',
       'relative_velocity', 'miss_distance', 'orbiting_body', 'sentry_obje
ct',
       'absolute_magnitude'],
      dtype='object')
```

In [43]:

```
df_columns = list(dataX.columns)
```

In [44]:

```
trainX=dataX.iloc[M[0:(m//4)*3]]
trainY=dataY.iloc[M[0:(m//4)*3]]
testX=dataX.iloc[M[(m//4)*3:]]
testY=dataY.iloc[M[(m//4)*3:]]
```

In [45]:

```
train_df=trainX
test_df=testX
```

In [46]:

```
df_columns = list(dataX.columns)
train_df.columns=df_columns
test_df.columns=df_columns
```

In [47]:

```
def create_numeric_feature(input_df):
    use_columns = df_columns
    return input_df[use_columns].copy()
```

In [48]:

```
from contextlib import contextmanager
from time import time

class Timer:
    def __init__(self, logger=None, format_str='{:.3f}[s]', prefix=None, suffix=Non

        if prefix: format_str = str(prefix) + sep + format_str
        if suffix: format_str = format_str + sep + str(suffix)
        self.format_str = format_str
        self.logger = logger
        self.start = None
        self.end = None

    @property
    def duration(self):
        if self.end is None:
            return 0
        return self.end - self.start

    def __enter__(self):
        self.start = time()

    def __exit__(self, exc_type, exc_val, exc_tb):
        self.end = time()
        out_str = self.format_str.format(self.duration)
        if self.logger:
            self.logger.info(out_str)
        else:
            print(out_str)
```

In [49]:

```
from tqdm import tqdm

def to_feature(input_df):

    processors = [
        create_numeric_feature,
    ]

    out_df = pd.DataFrame()

    for func in tqdm(processors, total=len(processors)):
        with Timer(prefix='create' + func.__name__ + ' '):
            _df = func(input_df)

        assert len(_df) == len(input_df), func.__name__
        out_df = pd.concat([out_df, _df], axis=1)

    return out_df
```

In [50]:

```
train_feat_df = to_feature(train_df)
test_feat_df = to_feature(test_df)
```

```
100%|████████████████████████████████████████████| 1/1 [00:00<00:00, 17.85
it/s]
100%|███████████████████████████████████████████| 1/1 [00:00<00:00, 125.04
it/s]

createcreate_numeric_feature  0.048[s]
createcreate_numeric_feature  0.000[s]
```

In [51]:

```
import lightgbm as lgbm
from sklearn.metrics import mean_squared_error

def fit_lgbm(X, y, cv,
             params: dict=None,
             verbose: int=50):

    if params is None:
        params = {}

    models = []
    oof_pred = np.zeros_like(y, dtype=np.float)

    for i, (idx_train, idx_valid) in enumerate(cv):
        x_train, y_train = X[idx_train], y[idx_train]
        x_valid, y_valid = X[idx_valid], y[idx_valid]

        clf = lgbm.LGBMRegressor(**params)

        with Timer(prefix='fit fold={} '.format(i)):
            clf.fit(x_train, y_train,
                    eval_set=[(x_valid, y_valid)],
                    early_stopping_rounds=100,
                    verbose=verbose)

        pred_i = clf.predict(x_valid)
        oof_pred[idx_valid] = pred_i
        models.append(clf)
        print(f'Fold {i} RMSLE: {mean_squared_error(y_valid, pred_i) ** .5:.4f}')
        print()

    score = mean_squared_error(y, oof_pred) ** .5
    print('-' * 50)
    print('FINISHED | Whole RMSLE: {:.4f}'.format(score))
    return oof_pred, models
```

In [52]:

```
params = {
    'objective': 'rmse',
    'learning_rate': .1,
    'reg_lambda': 1.,
    'reg_alpha': .1,
    'max_depth': 5,
    'n_estimators': 10000,
    'colsample_bytree': .5,
    'min_child_samples': 10,
    'subsample_freq': 3,
    'subsample': .9,
    'importance_type': 'gain',
    'random_state': 71,
    'num_leaves': 62
}
```

In [53]:

```
y = trainY
ydf=pd.DataFrame(y)
ydf
```

Out[53]:

	hazardous
55817	False
65066	False
24247	False
71521	True
50888	False
...	...
86721	False
89457	False
25957	True
9203	False
45194	False

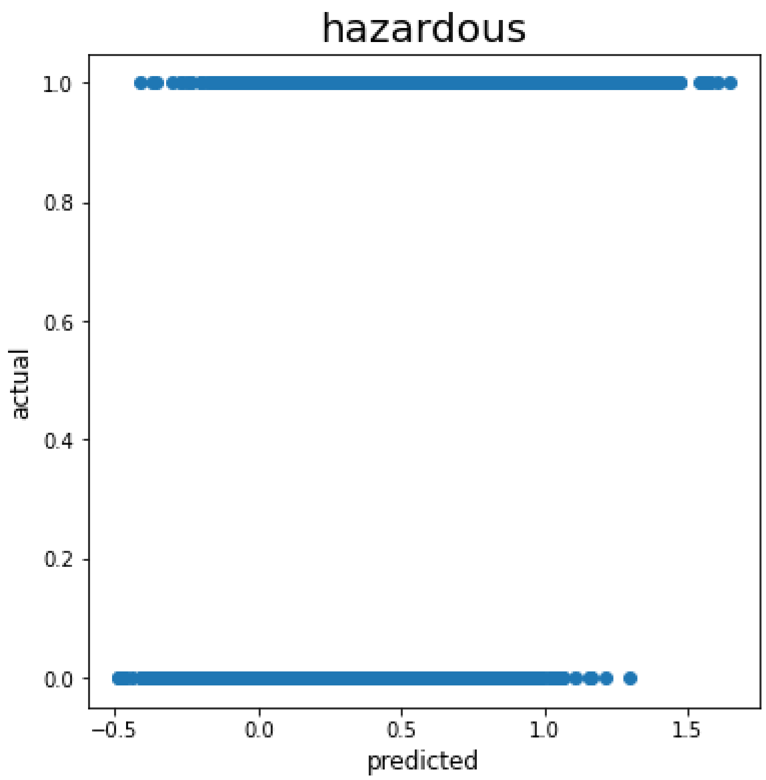
68127 rows × 1 columns

In [54]:

```
from sklearn.model_selection import KFold

for i in range(1):
    fold = KFold(n_splits=5, shuffle=True, random_state=71)
    ydfi=ydf.iloc[:,i]
    y=np.array(ydfi)
    cv = list(fold.split(train_feat_df, y))
    oof, models = fit_lgbm(train_feat_df.values, y, cv, params=params, verbose=500)

    fig,ax = plt.subplots(figsize=(6,6))
    ax.set_title(target[i],fontsize=20)
    ax.set_xlabel('predicted',fontsize=12)
    ax.set_ylabel('actual',fontsize=12)
    ax.scatter(oof,y)
```

```
[500]	valid_0's rmse: 0.208091
[1000]	valid_0's rmse: 0.199212
[1500]	valid_0's rmse: 0.19451
[2000]	valid_0's rmse: 0.191958
[2500]	valid_0's rmse: 0.189737
[3000]	valid_0's rmse: 0.188604
[3500]	valid_0's rmse: 0.187707
[4000]	valid_0's rmse: 0.186799
fit fold=0  16.076[s]
Fold 0 RMSLE: 0.1867

[500]	valid_0's rmse: 0.212186
[1000]	valid_0's rmse: 0.203067
[1500]	valid_0's rmse: 0.198317
[2000]	valid_0's rmse: 0.195867
[2500]	valid_0's rmse: 0.194005
[3000]	valid_0's rmse: 0.192648
[3500]	valid_0's rmse: 0.191834
[4000]	valid_0's rmse: 0.191206
fit fold=1  15.216[s]
Fold 1 RMSLE: 0.1910

[500]	valid_0's rmse: 0.21388
[1000]	valid_0's rmse: 0.205295
[1500]	valid_0's rmse: 0.200106
[2000]	valid_0's rmse: 0.196901
[2500]	valid_0's rmse: 0.195201
[3000]	valid_0's rmse: 0.194026
[3500]	valid_0's rmse: 0.193056
[4000]	valid_0's rmse: 0.19239
fit fold=2  16.449[s]
Fold 2 RMSLE: 0.1922

[500]	valid_0's rmse: 0.213621
[1000]	valid_0's rmse: 0.20372
[1500]	valid_0's rmse: 0.197921
[2000]	valid_0's rmse: 0.194829
[2500]	valid_0's rmse: 0.192182
[3000]	valid_0's rmse: 0.190936
[3500]	valid_0's rmse: 0.189823
[4000]	valid_0's rmse: 0.188969
```

```
fit fold=3  13.456[s]
Fold 3 RMSLE: 0.1889

[500]   valid_0's rmse: 0.210971
[1000]  valid_0's rmse: 0.202455
[1500]  valid_0's rmse: 0.197414
[2000]  valid_0's rmse: 0.194025
[2500]  valid_0's rmse: 0.192288
[3000]  valid_0's rmse: 0.190982
[3500]  valid_0's rmse: 0.190223
[4000]  valid_0's rmse: 0.189702
fit fold=4  13.405[s]
Fold 4 RMSLE: 0.1896

--------------------------------------------------
FINISHED | Whole RMSLE: 0.1897
```



In [55]:

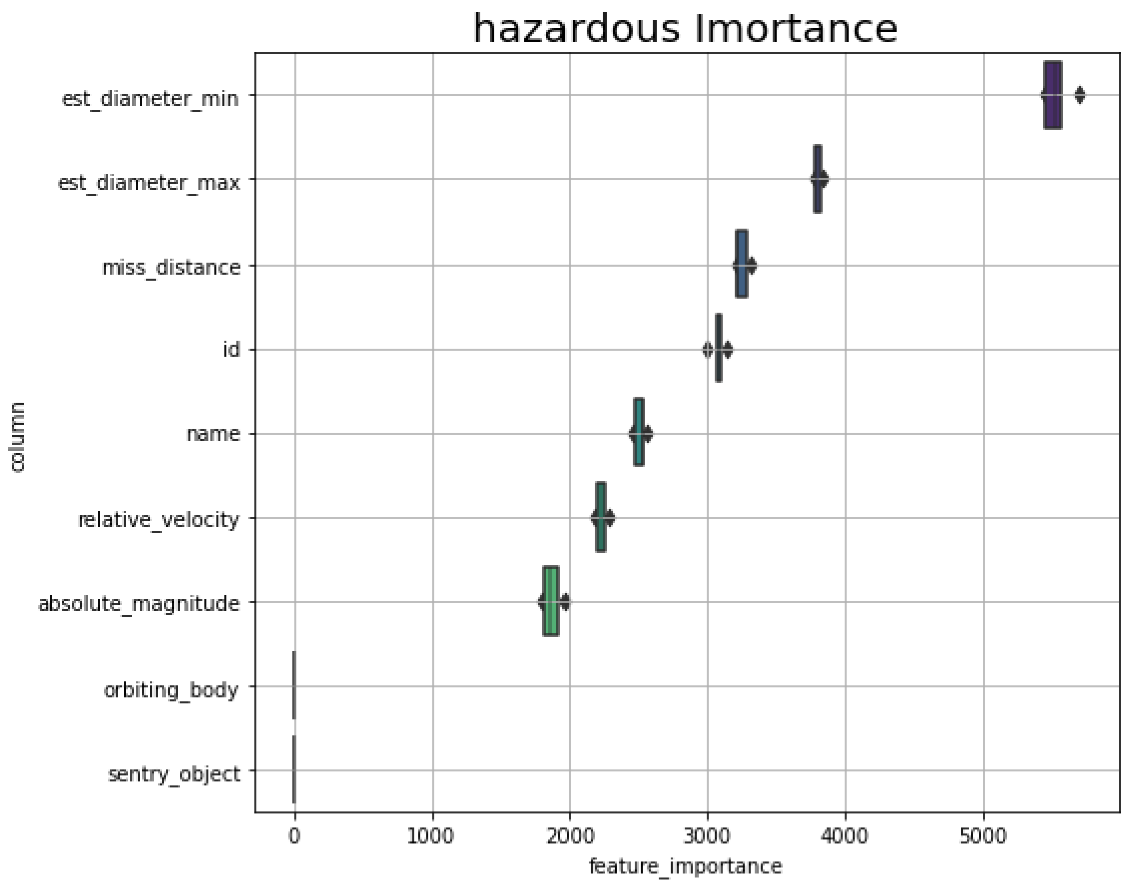
```
def visualize_importance(models, feat_train_df):

    feature_importance_df = pd.DataFrame()
    for i, model in enumerate(models):
        _df = pd.DataFrame()
        _df['feature_importance'] = model.feature_importances_
        _df['column'] = feat_train_df.columns
        _df['fold'] = i + 1
        feature_importance_df = pd.concat([feature_importance_df, _df],
                                          axis=0, ignore_index=True)

    order = feature_importance_df.groupby('column')\
        .sum()[['feature_importance']]\
        .sort_values('feature_importance', ascending=False).index[:50]

    fig, ax = plt.subplots(figsize=(8, max(6, len(order) * .25)))
    sns.boxenplot(data=feature_importance_df,
                  x='feature_importance',
                  y='column',
                  order=order,
                  ax=ax,
                  palette='viridis',
                  orient='h')

    ax.tick_params(axis='x', rotation=0)
    #ax.set_title('Importance')
    ax.grid()
    fig.tight_layout()

    return fig,ax
```

In [56]:

```
for i in range(1):
    fold = KFold(n_splits=5, shuffle=True, random_state=71)
    ydfi=ydf.iloc[:,i]
    y=np.array(ydfi)
    cv = list(fold.split(train_feat_df, y))
    oof, models = fit_lgbm(train_feat_df.values, y, cv, params=params, verbose=500)
    fig, ax = visualize_importance(models, train_feat_df)
    ax.set_title(target[i]+' Imortance',fontsize=20)
```

```
[500]   valid_0's rmse: 0.208091
[1000]  valid_0's rmse: 0.199212
[1500]  valid_0's rmse: 0.19451
[2000]  valid_0's rmse: 0.191958
[2500]  valid_0's rmse: 0.189737
[3000]  valid_0's rmse: 0.188604
[3500]  valid_0's rmse: 0.187707
[4000]  valid_0's rmse: 0.186799
fit fold=0  16.436[s]
Fold 0 RMSLE: 0.1867

[500]   valid_0's rmse: 0.212186
[1000]  valid_0's rmse: 0.203067
[1500]  valid_0's rmse: 0.198317
[2000]  valid_0's rmse: 0.195867
[2500]  valid_0's rmse: 0.194005
[3000]  valid_0's rmse: 0.192648
[3500]  valid_0's rmse: 0.191834
[4000]  valid_0's rmse: 0.191206
fit fold=1  14.439[s]
Fold 1 RMSLE: 0.1910

[500]   valid_0's rmse: 0.21388
[1000]  valid_0's rmse: 0.205295
[1500]  valid_0's rmse: 0.200106
[2000]  valid_0's rmse: 0.196901
[2500]  valid_0's rmse: 0.195201
[3000]  valid_0's rmse: 0.194026
[3500]  valid_0's rmse: 0.193056
[4000]  valid_0's rmse: 0.19239
fit fold=2  15.624[s]
Fold 2 RMSLE: 0.1922

[500]   valid_0's rmse: 0.213621
[1000]  valid_0's rmse: 0.20372
[1500]  valid_0's rmse: 0.197921
[2000]  valid_0's rmse: 0.194829
[2500]  valid_0's rmse: 0.192182
[3000]  valid_0's rmse: 0.190936
[3500]  valid_0's rmse: 0.189823
[4000]  valid_0's rmse: 0.188969
fit fold=3  13.511[s]
Fold 3 RMSLE: 0.1889

[500]   valid_0's rmse: 0.210971
[1000]  valid_0's rmse: 0.202455
[1500]  valid_0's rmse: 0.197414
[2000]  valid_0's rmse: 0.194025
```

```
[2500]  valid_0's rmse: 0.192288
[3000]  valid_0's rmse: 0.190982
[3500]  valid_0's rmse: 0.190223
[4000]  valid_0's rmse: 0.189702
fit fold=4  14.723[s]
Fold 4 RMSLE: 0.1896

--------------------------------------------------
FINISHED | Whole RMSLE: 0.1897
```



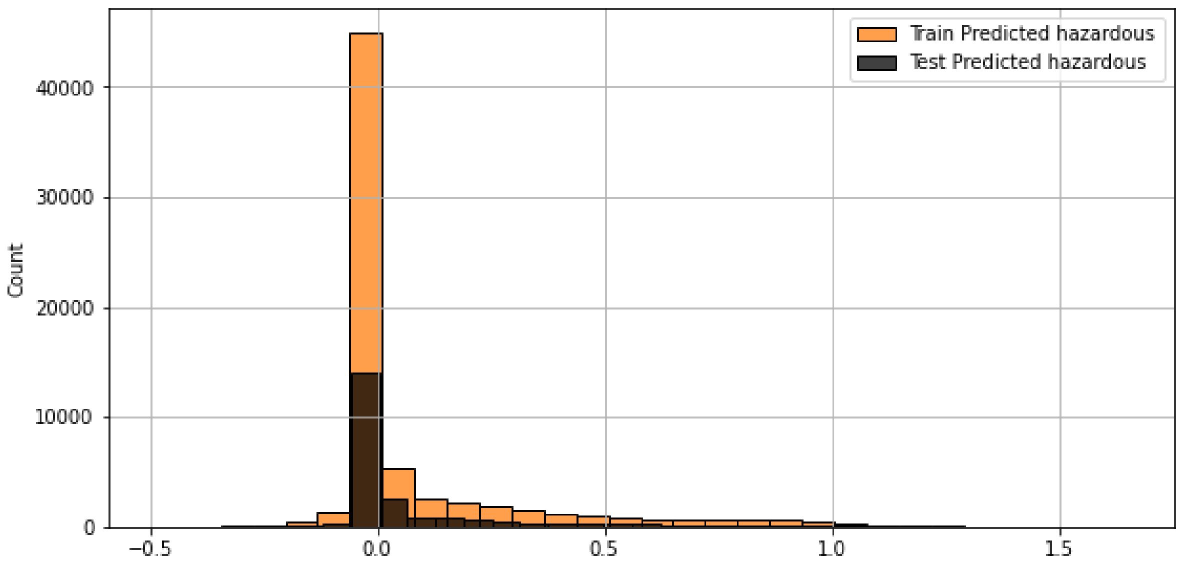
In [57]:

```python
preds=[]
for i in range(5):
    preds += [models[i].predict(test_feat_df.values)/5]
predsT=np.array(preds).T
preds2=[]
preds3=[]
for item in predsT:
    value=sum(item)
    preds2+=[value]
    preds3+=[int(np.where(value<0.5,0,1))]
print(preds2[0:5])
print(preds3[0:5])
```

```
[-0.002899381794562423, 0.0005748616101924781, 0.3833633806196074, 0.00013
233605794080955, 0.8473538524387714]
[0, 0, 0, 0, 1]
```

In [58]:

```python
for i in range(1):
    fig, ax = plt.subplots(figsize=(10,5))
    sns.histplot(oof, label='Train Predicted '+target[i], ax=ax, color='C1',bins=30
    sns.histplot(preds2, label='Test Predicted '+target[i], ax=ax, color='black',bi
    ax.legend()
    ax.grid()
```



In [59]:

```python
from sklearn.metrics import classification_report
print(classification_report(testY,preds3))
```

```
              precision    recall  f1-score   support

       False       0.97      0.99      0.98     20496
        True       0.86      0.69      0.77      2213

    accuracy                           0.96     22709
   macro avg       0.91      0.84      0.87     22709
weighted avg       0.96      0.96      0.96     22709
```

In [ ]:

```python

```