nasa-ptml

July 3, 2022

1 Introduction

Authors:

- Baptiste Bourdet
- Philippe Bernet
- Marius Dubosc
- Hugo Levy

The dataset comes from Kaggle: https://www.kaggle.com/datasets/sameepvani/nasa-nearest-earth-objects.

In this report we will try to analyze this data and compute models of both supervised and unsupervised learning to respond to a problem that was highlighted in many movies of science fiction: are any objects currently in orbit a danger to either satellites or earth.

1.1 Loading the data

First we need to load the data and analyse the different component it is made out of.

```
              id                 name  est_diameter_min  est_diameter_max  \
0        2162635  162635 (2000 SS164)          1.198271          2.679415
1        2277475    277475 (2005 WK4)          0.265800          0.594347
2        2512244   512244 (2015 YE18)          0.722030          1.614507
3        3596030          (2012 BV13)          0.096506          0.215794
4        3667127          (2014 GE35)          0.255009          0.570217
…            …                    …                 …                 …
90831    3763337           (2016 VX1)          0.026580          0.059435
90832    3837603           (2019 AD3)          0.016771          0.037501
90833   54017201           (2020 JP3)          0.031956          0.071456
90834   54115824           (2021 CN5)          0.007321          0.016370
90835   54205447           (2021 TW7)          0.039862          0.089133

       relative_velocity  miss_distance orbiting_body  sentry_object  \
0           13569.249224   5.483974e+07         Earth          False
1           73588.726663   6.143813e+07         Earth          False
2          114258.692129   4.979872e+07         Earth          False
3           24764.303138   2.543497e+07         Earth          False
```

```
4          42737.733765  4.627557e+07        Earth          False
…                    …             …            …             …
90831      52078.886692  1.230039e+07        Earth          False
90832      46114.605073  5.432121e+07        Earth          False
90833       7566.807732  2.840077e+07        Earth          False
90834      69199.154484  6.869206e+07        Earth          False
90835      27024.455553  5.977213e+07        Earth          False

       absolute_magnitude  hazardous
0                   16.73      False
1                   20.00       True
2                   17.83      False
3                   22.20      False
4                   20.09       True
…                   …         …
90831               25.00      False
90832               26.00      False
90833               24.60      False
90834               27.80      False
90835               24.12      False

[90836 rows x 10 columns]
```

The data is composed of the following columns:

- id: index number
- name: name of the object
- est_dimater_min: smallest size of the object in km
- est_dimater_max: biggest size of the object in km
- relative_velocity: velocity relative to Earth in km/h
- orbiting_body: the body the object is orbiting (Earth, Sun, the Moon …)
- sentry_object: whether or not the object is tracked by the sentry system of the nasa
- absolute_magnitude: visibility index, the smaller it is, the brigther the object it, the magnitude of the sun is -27 for example
- **hazardous**: whether or not the object is considerer a potential threat by the nasa, it is this column we will want to monitor

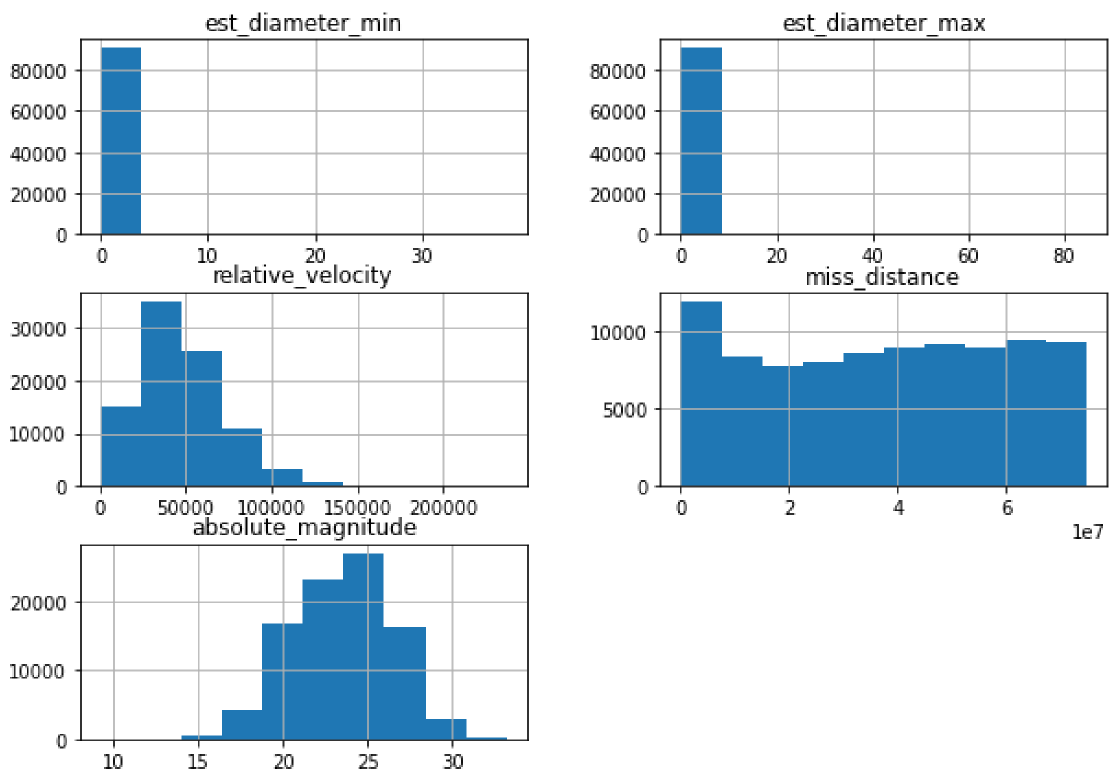
2 Analysis

2.1 Basic statistics

First we will look at statistics on the dataset.

```
Length of the dataset is 90836
```

```
Summary of all numerical values :
       est_diameter_min  est_diameter_max  relative_velocity  miss_distance  \
count      90836.000000      90836.000000       90836.000000   9.083600e+04
mean           0.127432          0.284947       48066.918918   3.706655e+07
std            0.298511          0.667491       25293.296961   2.235204e+07
min            0.000609          0.001362         203.346433   6.745533e+03
25%            0.019256          0.043057       28619.020645   1.721082e+07
50%            0.048368          0.108153       44190.117890   3.784658e+07
75%            0.143402          0.320656       62923.604633   5.654900e+07
max           37.892650         84.730541      236990.128088   7.479865e+07

       absolute_magnitude
count        90836.000000
mean            23.527103
std              2.894086
min              9.230000
25%             21.340000
50%             23.700000
75%             25.700000
max             33.200000
```




```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90836 entries, 0 to 90835
```

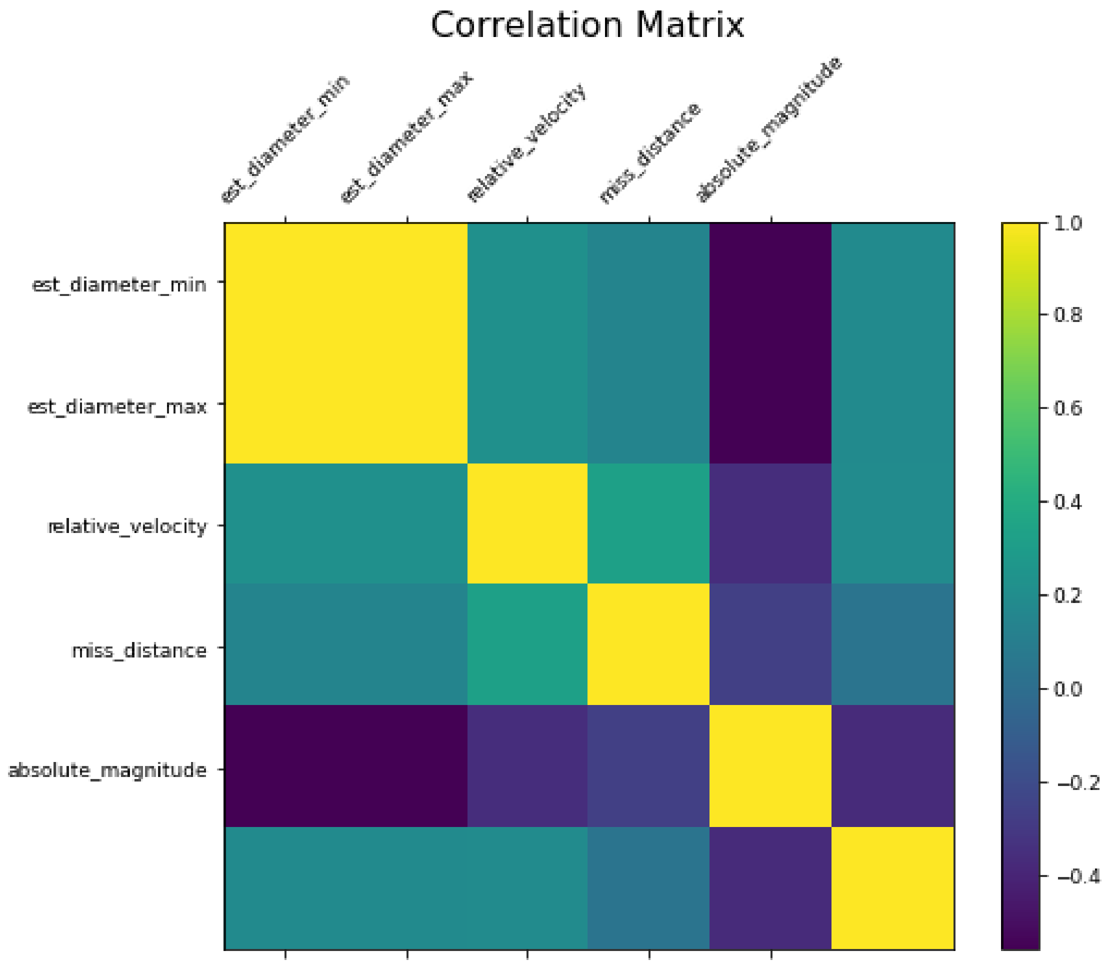
```
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  90836 non-null  object
 1   name                90836 non-null  object
 2   est_diameter_min    90836 non-null  float64
 3   est_diameter_max    90836 non-null  float64
 4   relative_velocity   90836 non-null  float64
 5   miss_distance       90836 non-null  float64
 6   orbiting_body       90836 non-null  category
 7   sentry_object       90836 non-null  bool
 8   absolute_magnitude  90836 non-null  float64
 9   hazardous           90836 non-null  bool
dtypes: bool(2), category(1), float64(5), object(2)
memory usage: 5.1+ MB
```

```
Visualization of categorical values :
       orbiting_body sentry_object hazardous
count          90836         90836     90836
unique             1             1         2
top            Earth         False     False
freq           90836         90836     81996
```

As we can see we have only one `orbiting body` and only one value for the `sentry`. Those two columns can therefore be removed from the dataset safely. The final column, the one we want to predict with the models, seems to have 10% of dangerous objects, the outliers we will try to identify.

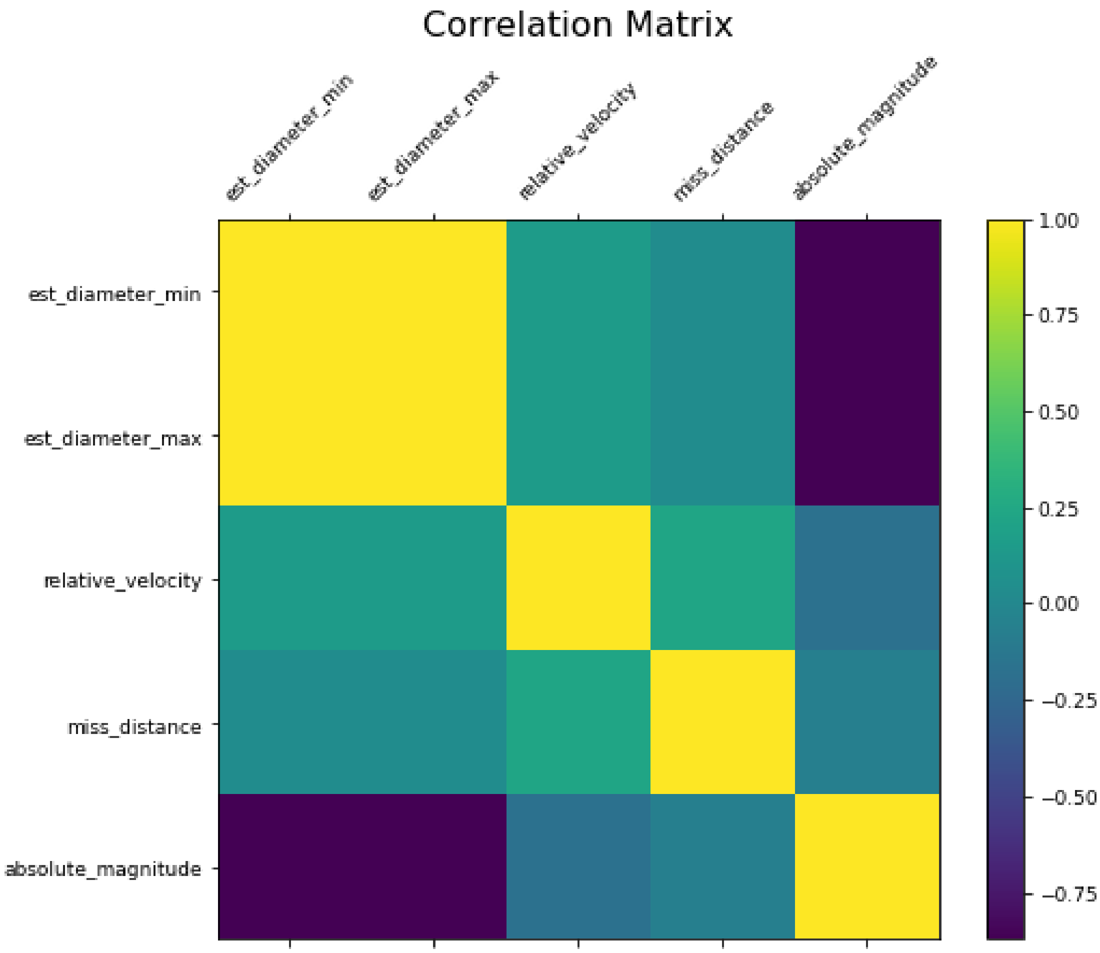
2.2 Correlation

The correlation matrix of our data is the following:



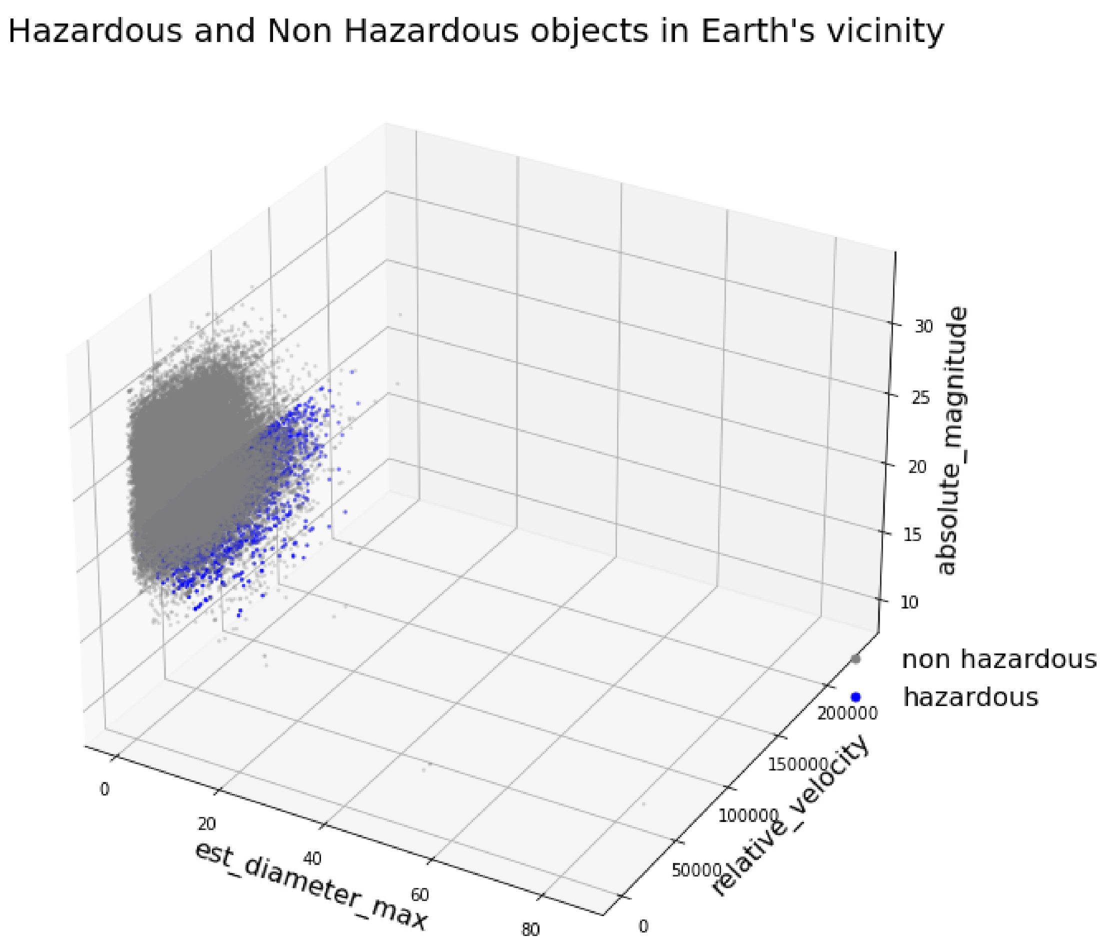
We can see in the matrix that the magnitude is negatively correlated with mos of the other variables. This implies that objects with a high magntiude, meaning not very visible objects, are usually smaller and slower. They also more importently do not consistute a threat seeing how the magnitude is negatively correlated with the hazardous.

When computing the correlation matrix only for the hazardous objects, we can see an even greater correlation between the magnitude and the size.

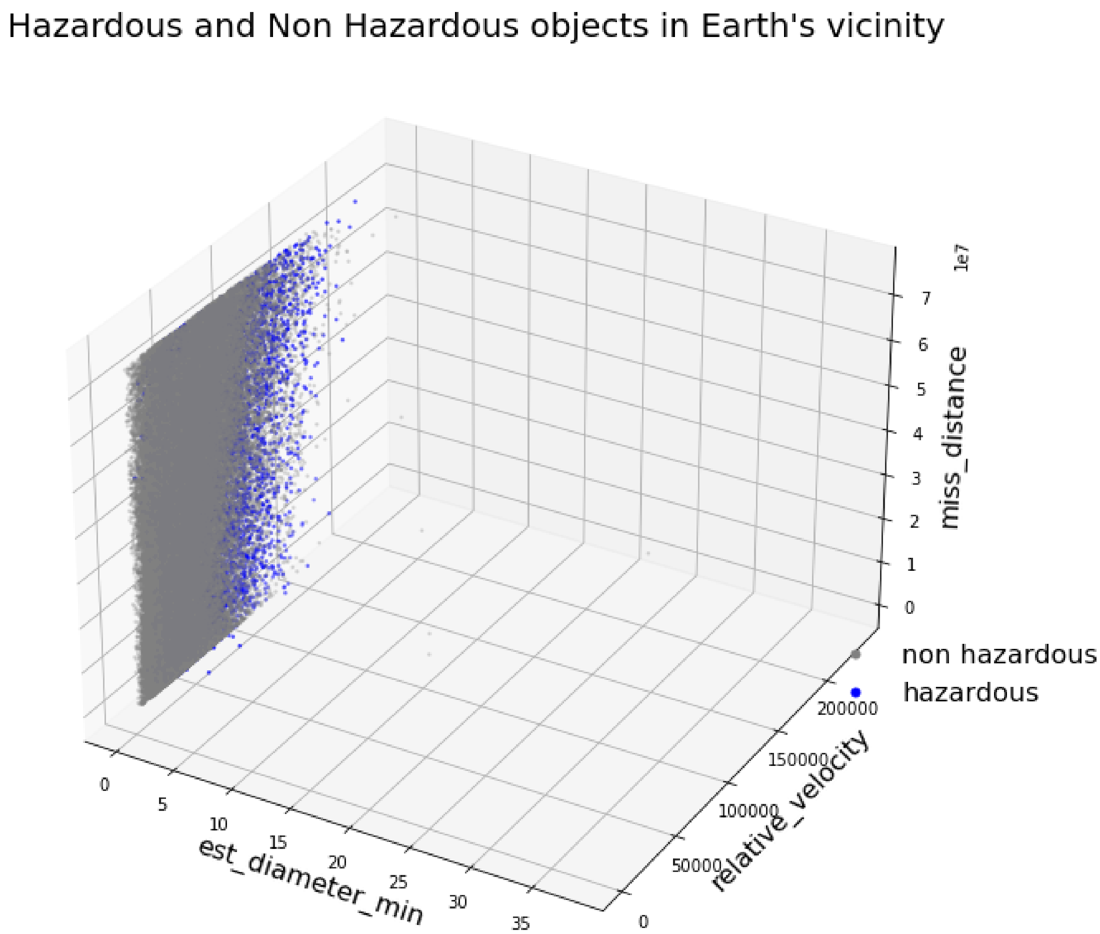


2.3 3D representation

We can also visualize multiple columns at once to see patterns.



Has can be seen on this first representation, the velocity seems to be slightly higher for objects considered a threat compared to the rest. The magnitude also seems to be capped at 20 for the hazardous objects, a high magnitude impliying a very dim object in the darkness of space.



On this second graph we can further see that the velocity seems to be an important factor but on the other end the miss distance is not very representative. Indeed the miss distance is not that important considering objects could be shifted our of their orbit very easily by the cosmic billiard played in the solar system by gravity. An oibject that missed earth by a lot could still be a threat.

3 Supervised Learning

First, we clean the dataset to get quantitative data to determine wether or not the object is considerer a potential threat by the nasa, it is this column we will want to monitor. We remove constant values (sentry_object=False and orbiting_body=Earth)

```
/opt/conda/lib/python3.9/site-packages/pandas/core/indexes/base.py:6982:
FutureWarning: In a future version, the Index constructor will not infer numeric
dtypes when passed object-dtype sequences (matching Series behavior)
  return Index(sequences[0], name=names)
```

```
          est_diameter_min  est_diameter_max  relative_velocity  \
id
2162635           1.198271          2.679415       13569.249224
2277475           0.265800          0.594347       73588.726663
2512244           0.722030          1.614507      114258.692129
3596030           0.096506          0.215794       24764.303138
3667127           0.255009          0.570217       42737.733765
…                    …                 …                  …
3763337           0.026580          0.059435       52078.886692
3837603           0.016771          0.037501       46114.605073
54017201          0.031956          0.071456        7566.807732
54115824          0.007321          0.016370       69199.154484
54205447          0.039862          0.089133       27024.455553

          miss_distance  absolute_magnitude  hazardous
id
2162635    5.483974e+07               16.73      False
2277475    6.143813e+07               20.00       True
2512244    4.979872e+07               17.83      False
3596030    2.543497e+07               22.20      False
3667127    4.627557e+07               20.09       True
…               …                 …         …
3763337    1.230039e+07               25.00      False
3837603    5.432121e+07               26.00      False
54017201   2.840077e+07               24.60      False
54115824   6.869206e+07               27.80      False
54205447   5.977213e+07               24.12      False

[90836 rows x 6 columns]
```

3.1 Initialize train and test sets

3.2 Let's try multiple models

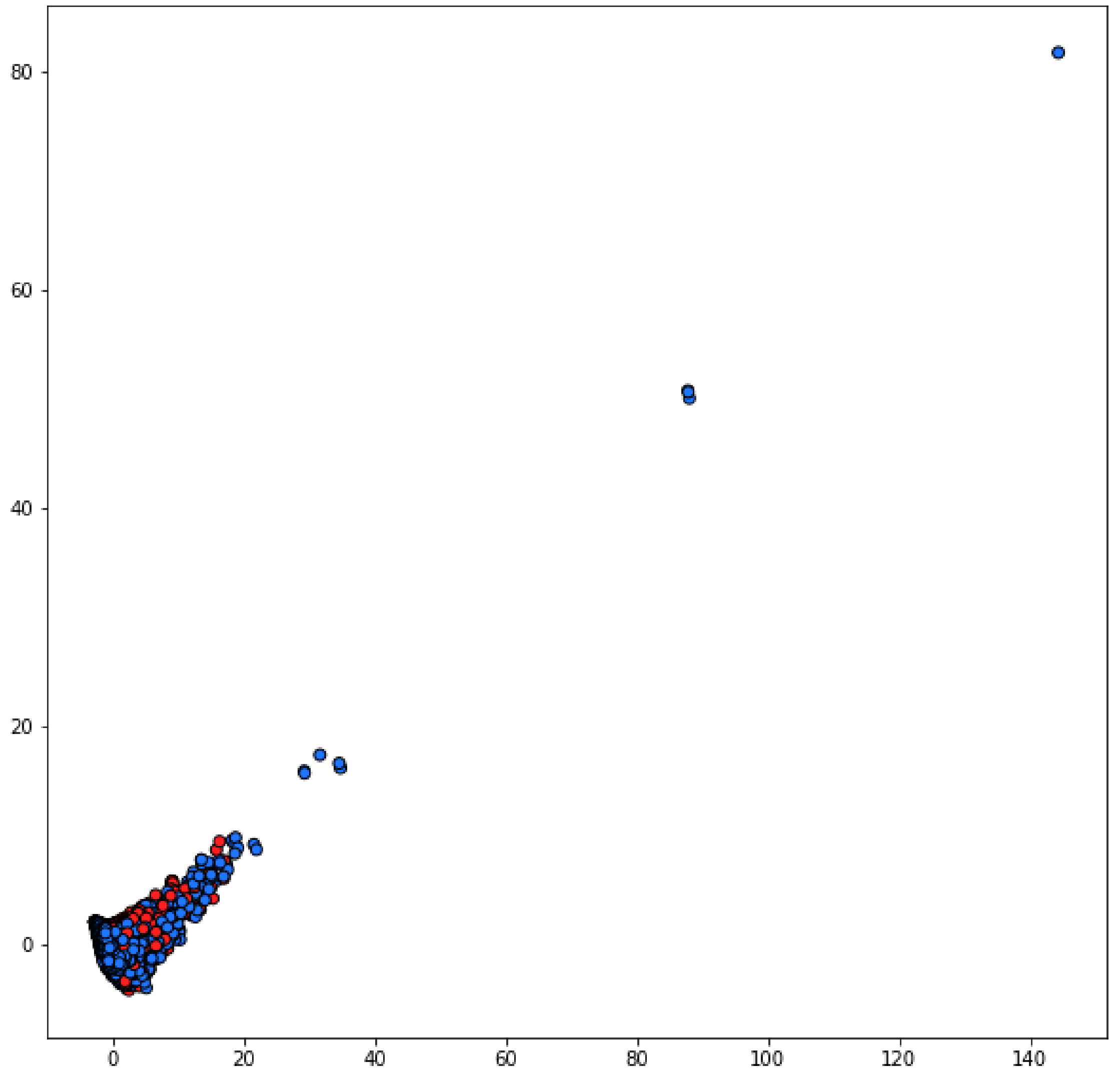
```
=============== LogisticRegressionCV(class_weight='balanced') ===============
------------ Classification Report ------------
train :
              precision    recall  f1-score   support

       False       0.94      0.38      0.54     54969
        True       0.12      0.76      0.20      5891

    accuracy                           0.41     60860
   macro avg       0.53      0.57      0.37     60860
weighted avg       0.86      0.41      0.50     60860

test :
              precision    recall  f1-score   support
```

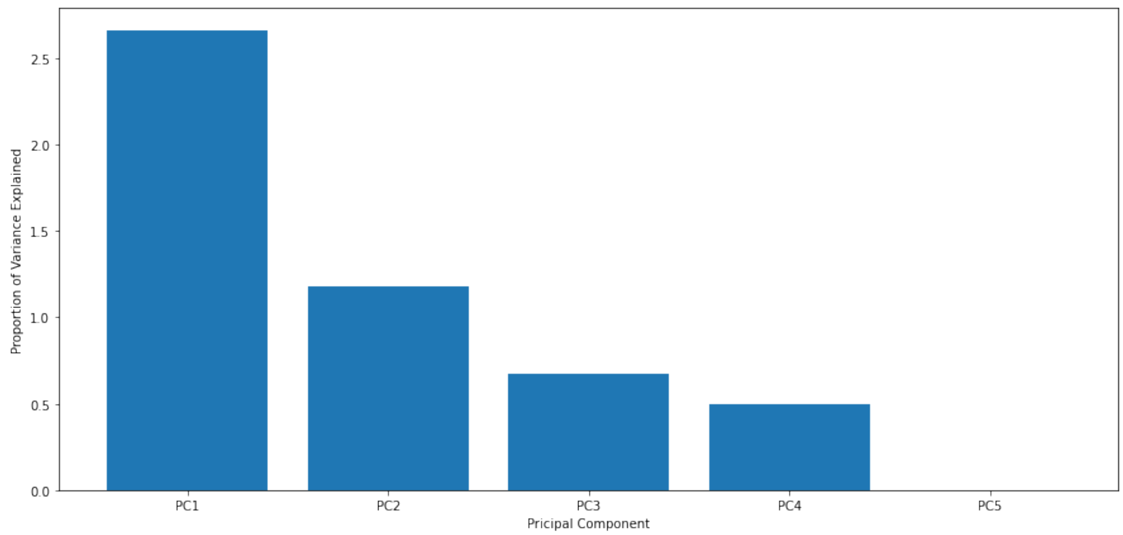
```
       False       0.93      0.38      0.54     27027
        True       0.12      0.76      0.20      2949

    accuracy                           0.42     29976
   macro avg       0.53      0.57      0.37     29976
weighted avg       0.85      0.42      0.51     29976

------------ Accuracy Score ------------
train :
0.41390075583305946
test :
0.4150987456631972

------------ Confusion matrix on test ------------
[[0.37758538 0.62241462]
 [0.24109868 0.75890132]]

=============== KNeighborsClassifier() ===============
------------ Classification Report ------------
train :
              precision    recall  f1-score   support

       False       0.91      0.99      0.95     54969
        True       0.66      0.11      0.18      5891

    accuracy                           0.91     60860
   macro avg       0.79      0.55      0.57     60860
weighted avg       0.89      0.91      0.88     60860

test :
              precision    recall  f1-score   support

       False       0.90      0.99      0.94     27027
        True       0.21      0.03      0.06      2949

    accuracy                           0.89     29976
   macro avg       0.56      0.51      0.50     29976
weighted avg       0.83      0.89      0.86     29976

------------ Accuracy Score ------------
train :
0.9083470259612225
test :
0.8922471310381639

------------ Confusion matrix on test ------------
[[0.98590299 0.01409701]
```

```
 [0.9660902  0.0339098 ]]

=============== DecisionTreeClassifier(random_state=0) ===============
------------ Classification Report ------------
train :
              precision    recall  f1-score   support

       False       1.00      1.00      1.00     54969
        True       1.00      1.00      1.00      5891

    accuracy                           1.00     60860
   macro avg       1.00      1.00      1.00     60860
weighted avg       1.00      1.00      1.00     60860

test :
              precision    recall  f1-score   support

       False       0.94      0.94      0.94     27027
        True       0.44      0.45      0.44      2949

    accuracy                           0.89     29976
   macro avg       0.69      0.69      0.69     29976
weighted avg       0.89      0.89      0.89     29976

------------ Accuracy Score ------------
train :
1.0
test :
0.8899786495863358

------------ Confusion matrix on test ------------
[[0.93817294 0.06182706]
 [0.55171244 0.44828756]]

=============== RandomForestClassifier(max_depth=3, n_estimators=50,
random_state=0) ===============
------------ Classification Report ------------
train :
              precision    recall  f1-score   support

       False       0.91      1.00      0.95     54969
        True       0.87      0.13      0.22      5891

    accuracy                           0.91     60860
   macro avg       0.89      0.56      0.59     60860
weighted avg       0.91      0.91      0.88     60860

test :
```

```
              precision    recall  f1-score   support

       False       0.91      1.00      0.95     27027
        True       0.85      0.11      0.20      2949

    accuracy                           0.91     29976
   macro avg       0.88      0.56      0.58     29976
weighted avg       0.91      0.91      0.88     29976

------------ Accuracy Score ------------
train :
0.9136214262241209
test :
0.9107619428876434

------------ Confusion matrix on test ------------
[[0.997817   0.002183  ]
 [0.88708037 0.11291963]]
```

3.2.1 Dimension Reduction to facilitate classification



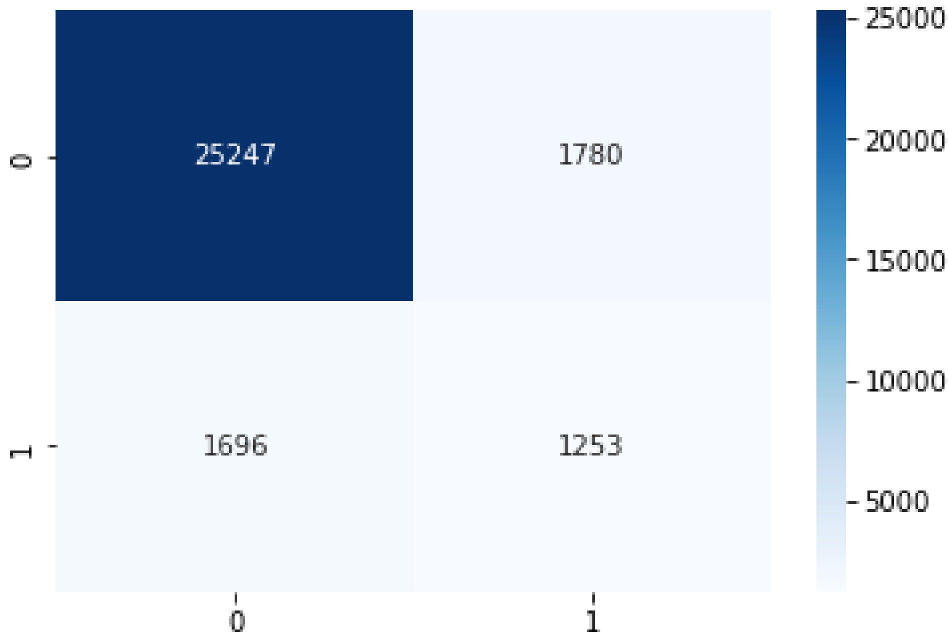



```
[5.31251159e-01 2.35392401e-01 1.33986946e-01 9.93694937e-02
 1.13445612e-21]
```

```
Score on training set : 1.0
Score on test set : 0.8840405657859621
```

```
Score on training set : 0.9060138021689123
Score on test set : 0.9042233787029623
```

```
[[25247  1780]
 [ 1696  1253]]
False negative : 1696
False positive : 1780
```




```
[[27018     9]
 [ 2862    87]]
False negative : 2862
False positive : 9
```