NASA:Asteroids Classification

The data is about Asteroids - NeoWs. NeoWs (Near Earth Object Web Service) is a RESTful web service for near earth Asteroid information. With NeoWs a user can: search for Asteroids based on their closest approach date to Earth, lookup a specific Asteroid with its NASA JPL small body id, as well as browse the overall data-set.

Inspiration

1. Finding potential hazardous and non-hazardous asteroids
2. Features responsible for claiming an asteroid to be hazardous

In [1]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?
client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleuser
content.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=emai
l%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2
Fwww.googleapis.com%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2
Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fwww.googleapis.com%2Faut
h%2Fpeopleapi.readonly&response_type=code

Enter your authorization code:
··········
Mounted at /content/gdrive
```

In [0]:
```
root_path = 'gdrive/My Drive/AITS/'
```

In [0]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
import numpy as np
import string
import matplotlib.pyplot as plt
import seaborn as sns
from pandas import DataFrame


from sklearn.metrics import confusion_matrix
from sklearn import metrics
%matplotlib inline
import os
import json,os,datetime
import csv
from sklearn.preprocessing import StandardScaler
```

1.1 Reading Data

In [6]:
```
nasa_data = pd.read_csv("gdrive/My Drive/AITS/nasa.csv")
nasa_data.head()
```

Out[6]:

	Neo Reference ID	Name	Absolute Magnitude	Est Dia in KM(min)	Est Dia in KM(max)	Est Dia in M(min)	Est Dia in M(max)	Est Dia in Miles(min)	Est I Miles(
0	3703080	3703080	21.6	0.127220	0.284472	127.219879	284.472297	0.079051	0.17
1	3723955	3723955	21.3	0.146068	0.326618	146.067964	326.617897	0.090762	0.20

	Neo Reference ID	Name	Absolute Magnitude	Est Dia in KM(min)	Est Dia in KM(max)	Est Dia in M(min)	Est Dia in M(max)	Est Dia in Miles(min)	Est I Miles
2	2446862	2446862	20.3	0.231502	0.517654	231.502122	517.654482	0.143849	0.32
3	3092506	3092506	27.4	0.008801	0.019681	8.801465	19.680675	0.005469	0.01
4	3514799	3514799	21.6	0.127220	0.284472	127.219879	284.472297	0.079051	0.17

In [7]:
```
nasa_data.shape
```

Out[7]: (4687, 40)

In [0]:
```
nasa_data=nasa_data.drop(['Close Approach Date','Orbit Determination Da
te','Orbiting Body','Equinox'], axis=1)
nasa_data.head()
```

Out[0]:

	Neo Reference ID	Name	Absolute Magnitude	Est Dia in KM(min)	Est Dia in KM(max)	Est Dia in M(min)	Est Dia in M(max)	Est Dia in Miles(min)	Est I Miles
0	3703080	3703080	21.6	0.127220	0.284472	127.219879	284.472297	0.079051	0.17
1	3723955	3723955	21.3	0.146068	0.326618	146.067964	326.617897	0.090762	0.20
2	2446862	2446862	20.3	0.231502	0.517654	231.502122	517.654482	0.143849	0.32
3	3092506	3092506	27.4	0.008801	0.019681	8.801465	19.680675	0.005469	0.01
4	3514799	3514799	21.6	0.127220	0.284472	127.219879	284.472297	0.079051	0.17

In [0]:
```
nasa_data.Hazardous[nasa_data.Hazardous==False]=0
nasa_data.Hazardous[nasa_data.Hazardous==True]=1
nasa_data.head()
```

Out[0]:

	Neo Reference ID	Name	Absolute Magnitude	Est Dia in KM(min)	Est Dia in KM(max)	Est Dia in M(min)	Est Dia in M(max)	Est Dia in Miles(min)	Est I Miles
0	3703080	3703080	21.6	0.127220	0.284472	127.219879	284.472297	0.079051	0.17
1	3723955	3723955	21.3	0.146068	0.326618	146.067964	326.617897	0.090762	0.20
2	2446862	2446862	20.3	0.231502	0.517654	231.502122	517.654482	0.143849	0.32
3	3092506	3092506	27.4	0.008801	0.019681	8.801465	19.680675	0.005469	0.01
4	3514799	3514799	21.6	0.127220	0.284472	127.219879	284.472297	0.079051	0.17

In [0]:
```
label = nasa_data['Hazardous']
final_data= nasa_data.drop('Hazardous', axis=1)
final_data.head()
```

Out[0]:

	Neo Reference ID	Name	Absolute Magnitude	Est Dia in KM(min)	Est Dia in KM(max)	Est Dia in M(min)	Est Dia in M(max)	Est Dia in Miles(min)	Est I Miles
0	3703080	3703080	21.6	0.127220	0.284472	127.219879	284.472297	0.079051	0.17
1	3723955	3723955	21.3	0.146068	0.326618	146.067964	326.617897	0.090762	0.20
2	2446862	2446862	20.3	0.231502	0.517654	231.502122	517.654482	0.143849	0.32
3	3092506	3092506	27.4	0.008801	0.019681	8.801465	19.680675	0.005469	0.01
4	3514799	3514799	21.6	0.127220	0.284472	127.219879	284.472297	0.079051	0.17

1.2 Data Cleaning

In [8]:
```
nasa_data.isnull().head()
```

Out[8]:

	Neo Reference ID	Name	Absolute Magnitude	Est Dia in KM(min)	Est Dia in KM(max)	Est Dia in M(min)	Est Dia in M(max)	Est Dia in Miles(min)	Est Dia in Miles(max)	E Fee
0	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	

In [9]:
```
nasa_data.describe()
```

Out[9]:

	Neo Reference ID	Name	Absolute Magnitude	Est Dia in KM(min)	Est Dia in KM(max)	Est Dia in M(min)	Est M
count	4.687000e+03	4.687000e+03	4687.000000	4687.000000	4687.000000	4687.000000	4687.0
mean	3.272298e+06	3.272298e+06	22.267865	0.204604	0.457509	204.604203	457.5
std	5.486011e+05	5.486011e+05	2.890972	0.369573	0.826391	369.573402	826.3
min	2.000433e+06	2.000433e+06	11.160000	0.001011	0.002260	1.010543	2.2
25%	3.097594e+06	3.097594e+06	20.100000	0.033462	0.074824	33.462237	74.8
50%	3.514799e+06	3.514799e+06	21.900000	0.110804	0.247765	110.803882	247.7
75%	3.690060e+06	3.690060e+06	24.500000	0.253837	0.567597	253.837029	567.5
max	3.781897e+06	3.781897e+06	32.100000	15.579552	34.836938	15579.552413	34836.9

In [11]:
```
nasa_data.corr().head()
```

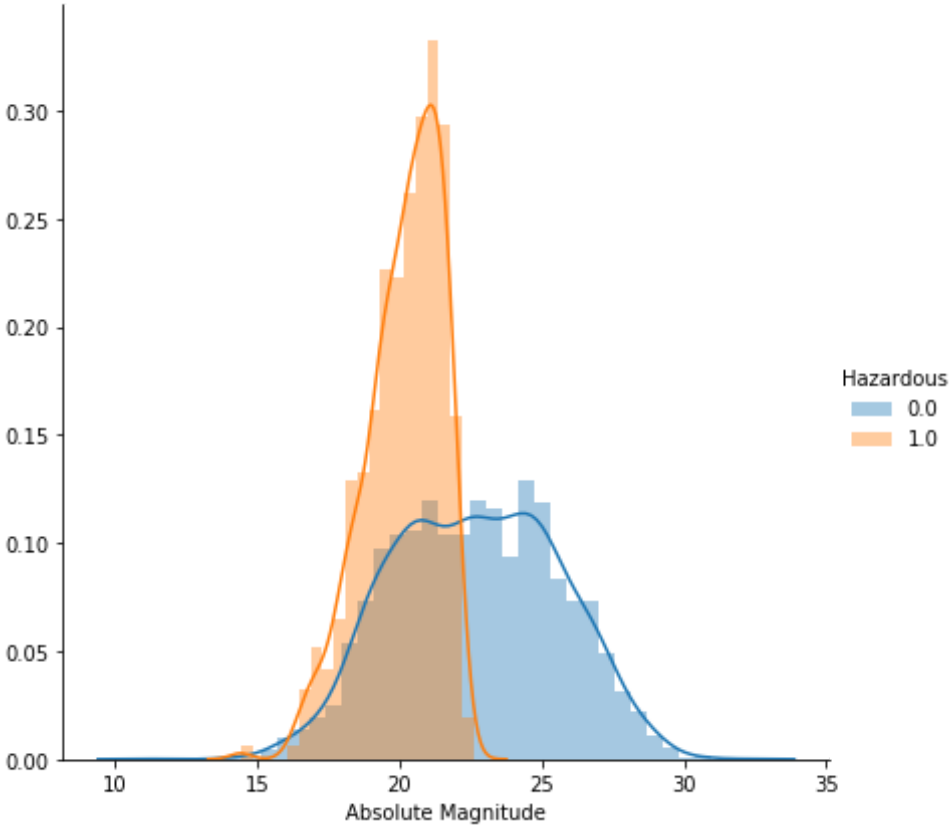
Out[11]:

	Neo Reference ID	Name	Absolute Magnitude	Est Dia in KM(min)	Est Dia in KM(max)	Est Dia in M(min)	Est Dia in M(max)	Est Dia i Miles(mir
Neo Reference ID	1.000000	1.000000	0.602381	-0.499821	-0.499821	-0.499821	-0.499821	-0.49982
Name	1.000000	1.000000	0.602381	-0.499821	-0.499821	-0.499821	-0.499821	-0.49982
Absolute Magnitude	0.602381	0.602381	1.000000	-0.613482	-0.613482	-0.613482	-0.613482	-0.61348
Est Dia in KM(min)	-0.499821	-0.499821	-0.613482	1.000000	1.000000	1.000000	1.000000	1.00000
Est Dia in KM(max)	-0.499821	-0.499821	-0.613482	1.000000	1.000000	1.000000	1.000000	1.00000

1.3 Exploratory Data Analysis

Histogram,Pdf

In [0]:
```
sns.FacetGrid(nasa_data,hue="Hazardous",size=6) \
   .map(sns.distplot, "Absolute Magnitude") \
   .add_legend()
plt.show()
```

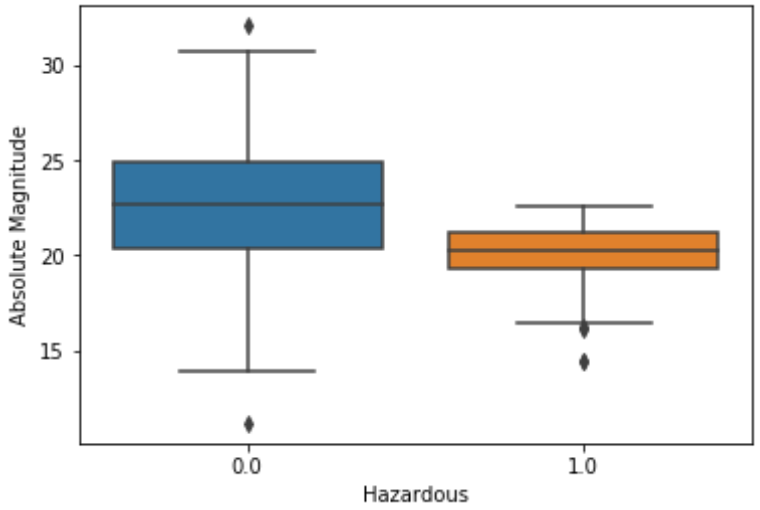


Observation

1. both points are ovelapping in 15 to 20 absolute magnitude
2. from range 22 magnitude onwords Non-hazardous Asteroids are present.

Box Plot

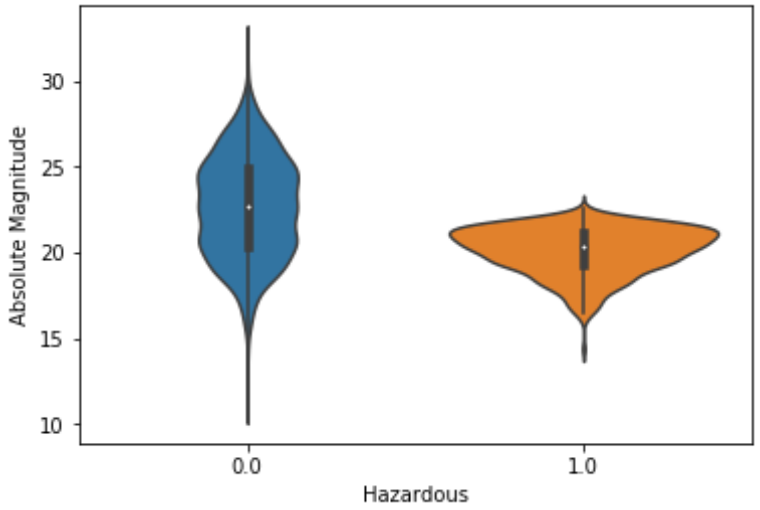
In [0]:
```
sns.boxplot(x='Hazardous',y='Absolute Magnitude', data=nasa_data)
plt.show()
```



Observation:

from absoute magnitude 20 to 25 non hazardous asteroids are present near earth. from absoute magnitude 19 to 22 hazardous asteroids are present near earth.

In [0]:
```
sns.violinplot(x='Hazardous',y='Absolute Magnitude', data=nasa_data)
plt.show()
```



In [0]:
```
sns.set_style("darkgrid");
sns.FacetGrid(nasa_data, hue="Hazardous", size=6) \
   .map(plt.scatter, "Absolute Magnitude","Mean Motion") \
   .add_legend();
plt.show();
```

1.4 PCA

In [0]:
```
# Data-preprocessing: Standardizing the data
std_data = StandardScaler().fit_transform(final_data)
print(std_data.shape)
```

```
(4687, 35)
```

In [0]:
```
#find the co-variance matrix which is : A^T * A
CoVMat = np.matmul(std_data.T , std_data)
print(CoVMat.shape)
```

```
(35, 35)
```

In [0]:
```
# finding the top two eigen-values and corresponding eigen-vectors
# for projecting onto a 2-Dim space.
from scipy.linalg import eigh
# the parameter 'eigvals' is defined (low value to heigh value)
# eigh function will return the eigen values in asending order
# this code generates only the top 2 (32 and 34) eigenvalues.
values, vectors = eigh(CoVMat , eigvals=(33,34))

print("The shape of Eigen Vectors", vectors.shape)
```

```
The shape of Eigen Vectors (35, 2)
```

In [0]:
```
# projecting the original data sample on the plane
#formed by two component eigen vectors by vector-vector multiplication.

new_data = np.matmul(vectors.T , std_data.T)
print("The shape of new data", new_data.shape)
```

```
The shape of new data (2, 4687)
```

In [0]:
```
# appending label to the 2d projected data
stacking = np.vstack((new_data , label)).T
dataframe = pd.DataFrame(data=stacking , columns=("1st component", "2nd
 Component" ,"Hazardous"))

dataframe.head(5)
```

Out[0]:

	1st component	2nd Component	Hazardous
0	0.460071	-0.693150	1.0
1	2.071261	0.304192	0.0
2	-1.446778	0.014228	1.0
3	0.718712	-2.039890	0.0
4	1.517848	-0.312831	1.0

In [0]:
```
# ploting the 2d data points with seaborn
import seaborn as sns
sns.FacetGrid(dataframe , hue='Hazardous', height=8).map(plt.scatter,
'1st component' , '2nd Component',s=10).add_legend()
plt.show()
```

1.5 Logistic Regression

In [0]:
```
#split dataset in features and target variable
feature_cols = ['Neo Reference ID','Name','Absolute Magnitude','Est Dia
 in KM(min)','Est Dia in KM(max)','Est Dia in M(min)','Est Dia in M(ma
x)','Est Dia in Miles(min)','Est Dia in Miles(max)','Est Dia in Feet(mi
n)','Est Dia in Feet(max)','Epoch Date Close Approach','Relative Veloci
ty km per sec','Relative Velocity km per hr','Miles per hour','Miss Dis
t.(Astronomical)','Miss Dist.(lunar)','Miss Dist.(kilometers)','Miss Di
st.(miles)','Orbit ID','Orbit Uncertainity','Minimum Orbit Intersectio
n','Jupiter Tisserand Invariant','Epoch Osculation','Eccentricity','Sem
i Major Axis','Inclination','Asc Node Longitude','Orbital Period','Peri
helion Distance','Perihelion Arg','Aphelion Dist','Perihelion Time','Me
an Anomaly','Mean Motion']
X = nasa_data[feature_cols] # Features
y = nasa_data.Hazardous # Target variable
```

In [0]:
```
from sklearn.model_selection import train_test_split
```

In [0]:
```
# split X and y into training and testing sets
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,rando
m_state=0)
```

In [0]:
```
# import the class
from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
logreg = LogisticRegression()

# fit the model with data
logreg.fit(X_train,y_train)
```

```
#
y_pred=logreg.predict(X_test)
```

In [17]:
```
# import the metrics class
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

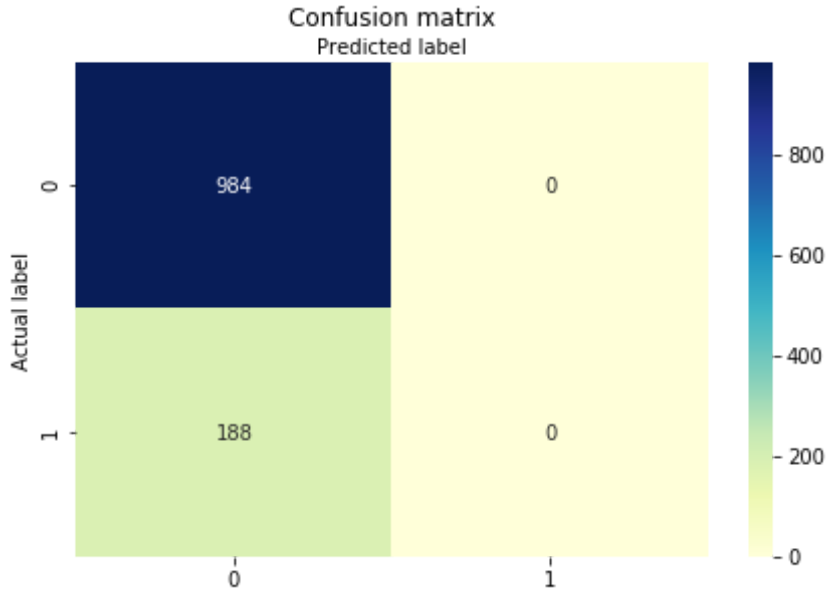
Out[17]:
```
array([[984,   0],
       [188,   0]])
```

Observation:

Actual Predictions are 984 and 0

In [18]:
```
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt=
'g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[18]:
```
Text(0.5, 257.44, 'Predicted label')
```



In [19]:
```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8395904436860068
```

Observation:

Accuracy is 83%

Conclusion

Using PCA, we reduced our feature data set and and by using regression model classifier we got 83% accuracy.