# Example_Project_B

March 1, 2022

**CMSE 201 Final Project**

#### 0.0.1 Emma Oswald

**12/6/2020**

# 1 *The Possible Impacts of Near-Earth Objects*

## 1.1 Background and Motivation

### 1.1.1 My Question:

1. How many NEO close approaches within the next year are potentially hazardous (will not burn up in the atmosphere)?
2. What kind of impact would they have if one of them were to come into contact with the earth?

### 1.1.2 Motivation:

The idea for this project grew more from a fascination with outer space and sci-fi movies than anything else. I have been staring at the night sky for as long as I can remember, always wondering what is up there to explore. Now, I am an astrophysics major whose passion for space has not died down. I knew right away that I wanted to do something concerning my major. At first I started looking at planetary datasets and then solar spectra. However, it was eventually the scenario from a sci-fi movie that convinced me to look at near-earth objects instead. That being said, I did not have a specific question that I wanted to answer when I started this project. Instead it was more 'action-based', meaning I first wanted to figure out how to read and interpret data about NEO's that has close approaches with the earth. Once I did that and started creating visuals of the data the question revealed itself.

### 1.1.3 Background:

Each day earth is hit with over 100 tons of dust and sand-sized particles. What's more is that about once a year a car-sized asteroid hits earth and creates a fireball that burns up in the atmosphere. That being said we hardly have events large enough to create any real damage to the surface of the earth, with them happening approximately once every 2,000 years (Dunbar, Brian. "Asteroid Fast Facts." NASA, NASA, 24 Mar. 2015). However, upon further research I realized that we have more close encounters with NEO's than one might think. In the next year alone we will have 111 close approaches with NEO's of varying type, size, and velocity. Here are some facts about the NEO's I will be looking at in my data set.

JPL description: Near-Earth Objects (NEOs) are comets and asteroids that have been nudged by the gravitational attraction of nearby planets into orbits that allow them to enter the Earth's neighborhood.

The C-type (chondrite) asteroids are most common, probably consist of clay and silicate rocks, and are dark in appearance. They are among the most ancient objects in the solar system.

The S-types ("stony") are made up of silicate materials and nickel-iron.

The M-types are metallic (nickel-iron). The asteroids' compositional differences are related to how far from the sun they formed. Some experienced high temperatures after they formed and partly melted, with iron sinking to the center and forcing basaltic (volcanic) lava to the surface. ("In Depth." NASA, NASA, 19 Dec. 2019)

Krasinsky et al. gives calculations for the mean densities of C, S, and M class asteroids as 1.38, 2.71, and 5.32 g/cm3 ("Standard Asteroid Physical Characteristics." Wikipedia, Wikimedia Foundation, 22 Jan. 2020.).

These are the density values I used to calculate each impact value.

## 1.2 Methodology

Dataset:

https://cneos.jpl.nasa.gov/ca/

JPL description: Near-Earth Objects (NEOs) are comets and asteroids that have been nudged by the gravitational attraction of nearby planets into orbits that allow them to enter the Earth's neighborhood.

The C-type (chondrite) asteroids are most common, probably consist of clay and silicate rocks, and are dark in appearance. They are among the most ancient objects in the solar system.

The S-types ("stony") are made up of silicate materials and nickel-iron.

The M-types are metallic (nickel-iron). The asteroids' compositional differences are related to how far from the sun they formed. Some experienced high temperatures after they formed and partly melted, with iron sinking to the center and forcing basaltic (volcanic) lava to the surface.

Krasinsky et al. gives calculations for the mean densities of C, S, and M class asteroids as 1.38, 2.71, and 5.32 g/cm3

We found that the kinetic energy of the impactor is in the range from 1.3e24 J to 5.8e25 J

In international standard units (SI), one ton of TNT is equal to 4.184 × 109 joule (J).

```
[1]: import matplotlib.pyplot as plt
     %matplotlib inline
     import numpy as np
     import pandas as pd
     import seaborn as sns
```

```
[2]: #reading in the dataset and dropping columns that aren't needed
     neo_1year = pd.read_csv('cneos_closeapproach_data.csv',index_col='Object')
     neo_1year = neo_1year.drop(columns='Unnamed: 8')
     #renaming certain columns so the names are easier to use in later code
     neo_1year = neo_1year.rename(columns={"Estimated Diameter": "Diameter"})
     neo_1year = neo_1year.rename(columns={'CA Distance Nominal (LD | au)':␣
      ↪'Distancenominalldau','CA Distance Minimum (LD | au)':␣
      ↪'distanceminimumldau'})
     neo_1year.head()
```

```
[2]:                  Close-Approach (CA) Date Distancenominalldau  \
     Object
     (2018 PK21)   2020-Dec-08 02:24 ± < 00:01     12.18 | 0.03129
     (2020 WK3)    2020-Dec-08 08:09 ± < 00:01      9.99 | 0.02567
     (2020 XD)     2020-Dec-09 08:33 ± < 00:01     12.47 | 0.03205
     (2020 VC7)    2020-Dec-11 20:03 ± < 00:01     14.62 | 0.03756
     (2019 XQ1)   2020-Dec-13 08:27 ±    01:19     18.30 | 0.04703

                 distanceminimumldau  V relative (km/s)  V infinity (km/s)  \
     Object
     (2018 PK21)     12.18 | 0.03129               3.13               3.10
     (2020 WK3)       9.94 | 0.02554               6.90               6.89
     (2020 XD)       12.46 | 0.03202               9.14               9.13
     (2020 VC7)      14.58 | 0.03745               6.97               6.96
     (2019 XQ1)      16.31 | 0.04190               8.61               8.60

                  H (mag)       Diameter
     Object
     (2018 PK21)      25.9  18 m -   39 m
     (2020 WK3)       25.1  26 m -   58 m
     (2020 XD)        25.6  20 m -   45 m
     (2020 VC7)       23.9  44 m -   99 m
     (2019 XQ1)       25.4  22 m -   49 m
```

```
[3]: #splitting the two values in the diameter column into two separate columns
     neo_1year[['Low','High']] = neo_1year.Diameter.str.split('-', expand=True)
     neo_1year[['Low Estimated Diameter (m)', 'Drop 1']] = neo_1year.Low.str.
      ↪split(expand=True)
     neo_1year[['High Estimated Diameter (m)', 'Drop 2']] = neo_1year.High.str.
      ↪split(expand=True)
     #dropping the new columns that are not needed
     neo_1year = neo_1year.drop(columns=['Low','High','Drop 1','Drop 2','Diameter'])

     #splitting the close approach distance columns into two separate columns each
     neo_1year[['CA Distance Nominal (LD)','CA Distance Nominal (au)']] = neo_1year.
      ↪Distancenominalldau.str.split('|',expand=True)
```

```
neo_1year[['CA Distance Minimum (LD)','CA Distance Minimum (au)']] = neo_1year.
 ↪distanceminimumldau.str.split('|',expand=True)
#dropping the Lunar Distance (LD) columns because they are not needed
neo_1year = neo_1year.drop(columns=['CA Distance Nominal (LD)', 'CA Distance␣
 ↪Minimum (LD)','Distancenominalldau','distanceminimumldau'])

#converting each of the new columns into floating point type so they can be␣
 ↪used in calculations.
neo_1year = neo_1year.astype({'Low Estimated Diameter (m)': 'float64', 'High␣
 ↪Estimated Diameter (m)': 'float64', 'CA Distance Nominal (au)': 'float64',␣
 ↪'CA Distance Minimum (au)': 'float64', 'V relative (km/s)': 'float64', 'V␣
 ↪infinity (km/s)': 'float64'})

neo_1year.head()
```

```
[3]:                  Close-Approach (CA) Date  V relative (km/s)  \
Object
(2018 PK21)   2020-Dec-08 02:24 ± < 00:01               3.13
(2020 WK3)    2020-Dec-08 08:09 ± < 00:01               6.90
(2020 XD)     2020-Dec-09 08:33 ± < 00:01               9.14
(2020 VC7)    2020-Dec-11 20:03 ± < 00:01               6.97
(2019 XQ1)   2020-Dec-13 08:27 ±    01:19               8.61

             V infinity (km/s)  H (mag)  Low Estimated Diameter (m)  \
Object
(2018 PK21)               3.10     25.9                        18.0
(2020 WK3)                6.89     25.1                        26.0
(2020 XD)                 9.13     25.6                        20.0
(2020 VC7)                6.96     23.9                        44.0
(2019 XQ1)                8.60     25.4                        22.0

             High Estimated Diameter (m)  CA Distance Nominal (au)  \
Object
(2018 PK21)                         39.0                   0.03129
(2020 WK3)                          58.0                   0.02567
(2020 XD)                           45.0                   0.03205
(2020 VC7)                          99.0                   0.03756
(2019 XQ1)                          49.0                   0.04703

             CA Distance Minimum (au)
Object
(2018 PK21)                   0.03129
(2020 WK3)                    0.02554
(2020 XD)                     0.03202
(2020 VC7)                    0.03745
(2019 XQ1)                    0.04190
```

[4]:
```
#looping through values in the two diameter columns to find the average/median␣
↪value
#this is so my calculations later on will be better approximated
avg_diameter = []
i=0
while i <= (len(neo_1year['Low Estimated Diameter (m)'])-1):
    avg = ((neo_1year.iloc[i,4])+(neo_1year.iloc[i,5]))/2
    avg_diameter.append(avg)
    i+=1

print(avg_diameter)

#the diameter values given are estimates based off of the luminosity of the NEO
#this is why there is such a large margin of error
```

```
[28.5, 42.0, 32.5, 71.5, 35.5, 27.5, 56.5, 13.0, 151.0, 88.0, 11.5, 56.5, 19.0,
285.0, 47.0, 14.4, 340.0, 41.0, 495.0, 495.0, 130.0, 94.0, 24.5, 295.0, 6.5,
265.0, 24.0, 310.0, 195.0, 225.0, 7.9, 13.7, 10.85, 190.0, 29.5, 151.0, 66.0,
26.5, 26.0, 108.5, 51.5, 47.0, 565.0, 16.5, 195.0, 68.5, 385.85, 10.15, 17.5,
71.5, 16.5, 24.0, 27.5, 12.35, 12.4, 24.5, 8.7, 19.0, 195.0, 12.35, 13.0, 10.05,
14.4, 144.0, 34.0, 9.4, 27.5, 225.0, 24.0, 13.0, 14.4, 21.5, 151.0, 24.5, 158.5,
35.0, 18.0, 73.0, 19.0, 240.55, 102.0, 37.5, 116.5, 31.0, 54.0, 68.0, 20.5,
29.5, 24.5, 68.0, 14.4, 19.5, 7.9, 165.0, 151.0, 20.5, 65.0, 165.0, 21.5, 13.85,
62.0, 275.0, 24.5, 79.0, 28.5, 137.0, 675.0, 32.5, 86.5, 19.5, 151.0]
```

[5]:
```
#adding the list of average diameters to the dataset as a column
neo_1year['Average Estimated Diameter (m)'] = avg_diameter
neo_1year.head()
```

[5]:
```
                 Close-Approach (CA) Date  V relative (km/s)  \
Object
(2018 PK21)   2020-Dec-08 02:24 ± < 00:01               3.13
(2020 WK3)    2020-Dec-08 08:09 ± < 00:01               6.90
(2020 XD)     2020-Dec-09 08:33 ± < 00:01               9.14
(2020 VC7)    2020-Dec-11 20:03 ± < 00:01               6.97
(2019 XQ1)   2020-Dec-13 08:27 ±    01:19               8.61

             V infinity (km/s)  H (mag)  Low Estimated Diameter (m)  \
Object
(2018 PK21)               3.10     25.9                        18.0
(2020 WK3)                6.89     25.1                        26.0
(2020 XD)                 9.13     25.6                        20.0
(2020 VC7)                6.96     23.9                        44.0
(2019 XQ1)                8.60     25.4                        22.0

             High Estimated Diameter (m)  CA Distance Nominal (au)  \
Object
(2018 PK21)                         39.0                   0.03129
```

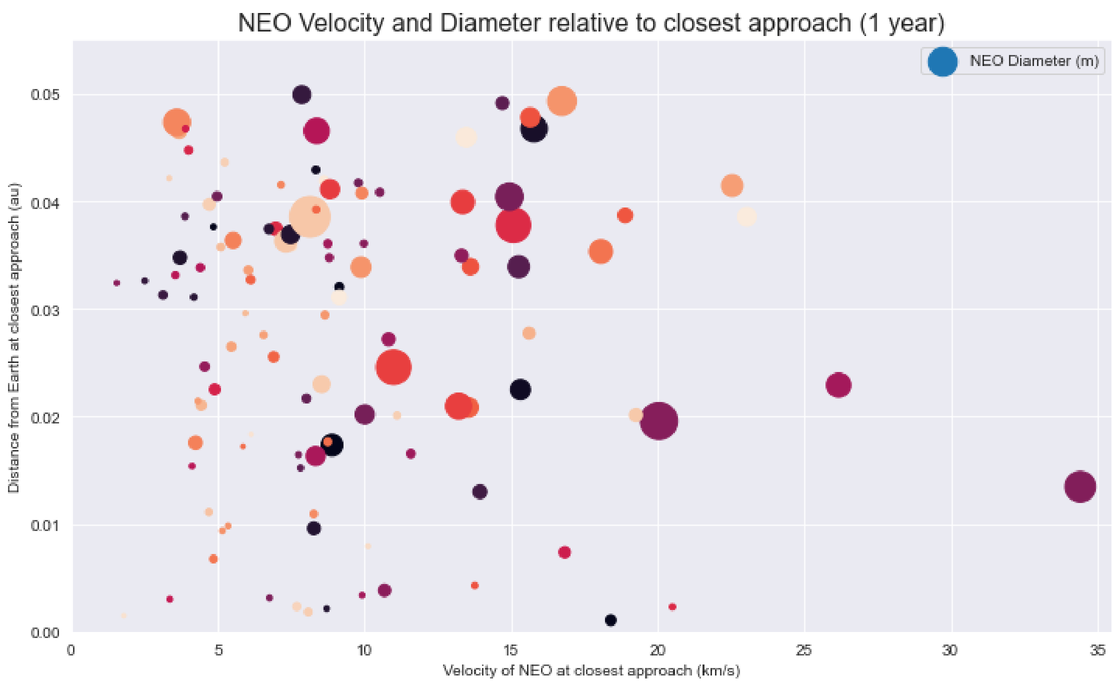
```
(2020 WK3)                        58.0                 0.02567
(2020 XD)                         45.0                 0.03205
(2020 VC7)                        99.0                 0.03756
(2019 XQ1)                        49.0                 0.04703

           CA Distance Minimum (au)  Average Estimated Diameter (m)
Object
(2018 PK21)                 0.03129                            28.5
(2020 WK3)                  0.02554                            42.0
(2020 XD)                   0.03202                            32.5
(2020 VC7)                  0.03745                            71.5
(2019 XQ1)                  0.04190                            35.5
```

[6]:
```
#renaming the date column so it will be easier to use later on
neo_1year = neo_1year.rename(columns={'Close-Approach (CA) Date':␣
 ↪'approachdate'})

#splitting the date column so that it only includes the year-month-day values
neo_1year[['Close Approach (CA) Date', 'approachtime', 'Uncertainty']] =␣
 ↪neo_1year.approachdate.str.split(' ', expand=True)
#splitting the time value so that it only includes the military time value
neo_1year[['Close Approach Time', 'drop']] = neo_1year.approachtime.str.
 ↪split('±',expand=True)

#dropping the unneeded columns
neo_1year = neo_1year.drop(columns = ['drop','approachdate', 'approachtime',␣
 ↪'Uncertainty'])

neo_1year.head()
```

[6]:
```
           V relative (km/s)  V infinity (km/s)  H (mag)  \
Object
(2018 PK21)             3.13               3.10     25.9
(2020 WK3)              6.90               6.89     25.1
(2020 XD)               9.14               9.13     25.6
(2020 VC7)              6.97               6.96     23.9
(2019 XQ1)              8.61               8.60     25.4

           Low Estimated Diameter (m)  High Estimated Diameter (m)  \
Object
(2018 PK21)                      18.0                         39.0
(2020 WK3)                       26.0                         58.0
(2020 XD)                        20.0                         45.0
(2020 VC7)                       44.0                         99.0
(2019 XQ1)                       22.0                         49.0

           CA Distance Nominal (au)  CA Distance Minimum (au)  \
Object
(2018 PK21)                   0.03129                    0.03129
(2020 WK3)                    0.02567                    0.02554
(2020 XD)                     0.03205                    0.03202
(2020 VC7)                    0.03756                    0.03745
(2019 XQ1)                    0.04703                    0.04190

             Average Estimated Diameter (m) Close Approach (CA) Date  \
Object
(2018 PK21)                            28.5                2020-Dec-08
(2020 WK3)                             42.0                2020-Dec-08
(2020 XD)                              32.5                2020-Dec-09
(2020 VC7)                             71.5                2020-Dec-11
(2019 XQ1)                             35.5                2020-Dec-13

            Close Approach Time
Object
(2018 PK21)             02:24
(2020 WK3)              08:09
(2020 XD)               08:33
(2020 VC7)              20:03
(2019 XQ1)              08:27
```

[7]:
```
#initializing a list of color values for easily distinguishable points on the␣
 ↪graph
colors = np.random.rand(111)
```

[8]:
```
#plotting distance from earth at closest approach vs.velocity at closest␣
 ↪approach
#the size of the points is relative to the diameter of each NEO
plt.figure(figsize = (12,7))
sns.set_style('darkgrid')
plt.scatter(neo_1year.iloc[:,0],neo_1year.iloc[:,6], s=neo_1year.iloc[:
 ↪,7],c=colors, label='NEO Diameter (m)')
plt.xlabel('Velocity of NEO at closest approach (km/s)')
plt.ylabel('Distance from Earth at closest approach (au)')
plt.axis([0,35.5,0,.055])
plt.title('NEO Velocity and Diameter relative to closest approach (1 year)',␣
 ↪fontsize='16')
plt.legend()

plt.savefig('1yeardiameter.jpeg')
```



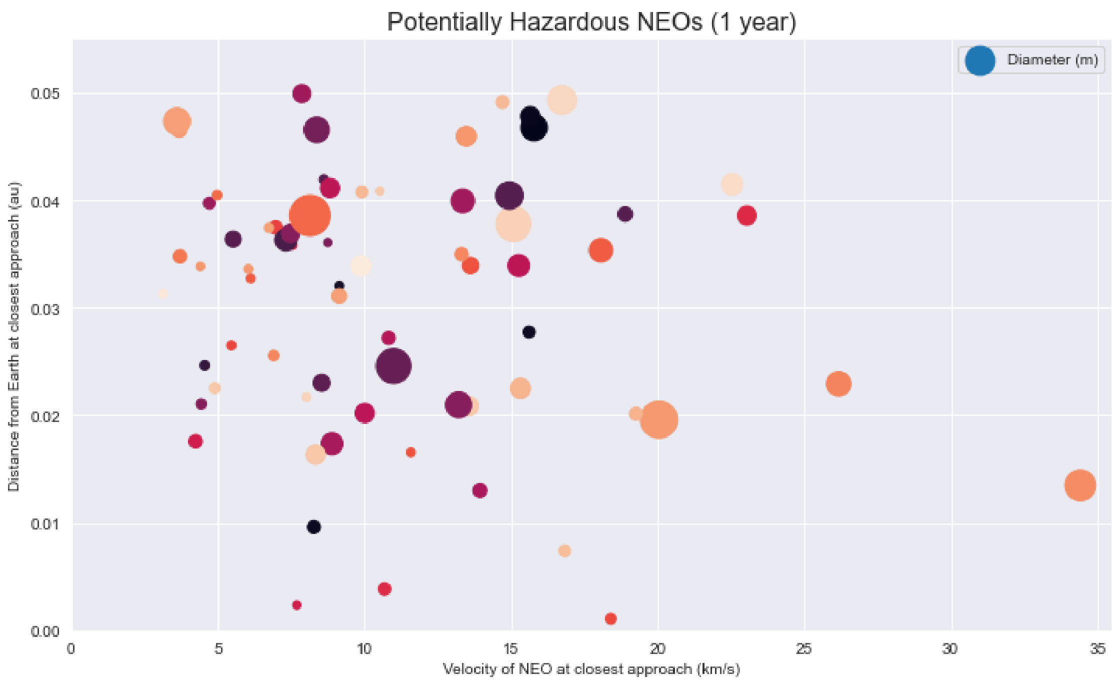
[9]:
```
# Sorting out the NEO's whose aveerage estimated diameter is small enough that␣
↪they will burn up in the atmosphere
# Sorting out the NEO's whose diameter is large enough to cause worldwide␣
↪effects.
# There is nothing in the dataset above that value.
wontburn = neo_1year[neo_1year['Average Estimated Diameter (m)']>=25]
#I used the average diameter value to stay consistent with the diameter value I␣
↪use througout the project.
small = neo_1year[neo_1year['Average Estimated Diameter (m)']<=25]
bigimpact = neo_1year[neo_1year['Average Estimated Diameter (m)']>=1000]
print(len(wontburn))
print(len(small))
print(len(bigimpact))
```

```
69
42
0
```

[10]:
```
# creating a smaller list of color values to distinguish the NEO's agaisnt␣
↪eachother
shortcolors = np.random.rand(69)
```

[11]:
```
# plotting the distance, velocity and diameter of NEO's larger than 25 meters␣
↪in diameter.
plt.figure(figsize = (12,7))
```

```
sns.set_style('darkgrid')
plt.scatter(wontburn.iloc[:,0],wontburn.iloc[:,6], s=wontburn.iloc[:
 ↪,7],c=shortcolors, label='Diameter (m)')
plt.xlabel('Velocity of NEO at closest approach (km/s)')
plt.ylabel('Distance from Earth at closest approach (au)')
plt.axis([0,35.5,0,.055])
plt.title('Potentially Hazardous NEOs (1 year)', fontsize='16')
plt.legend()

plt.savefig('1yeardiameterdangerous.jpeg')
```



[12]:
```
#reading in the csv of NEO close approaches within the next 60 days
neo_60 = pd.read_csv('cneos_closeapproach_data_60days.csv',index_col='Object')
neo_60 = neo_60.drop(columns='Unnamed: 8') #dropping unneeded data
neo_60 = neo_60.rename(columns={"Estimated Diameter": "Diameter"}) #renaming␣
 ↪columns so they are easier to call
neo_60 = neo_60.rename(columns={'CA Distance Nominal (LD | au)':␣
 ↪'Distancenominalldau','CA Distance Minimum (LD | au)':␣
 ↪'distanceminimumldau'})

#splitting diameter column and dropping the resulting unneeded columns
neo_60[['Low','High']] = neo_60.Diameter.str.split('-', expand=True)
neo_60[['Low Estimated Diameter (m)', 'Drop 1']] = neo_60.Low.str.
 ↪split(expand=True)
```

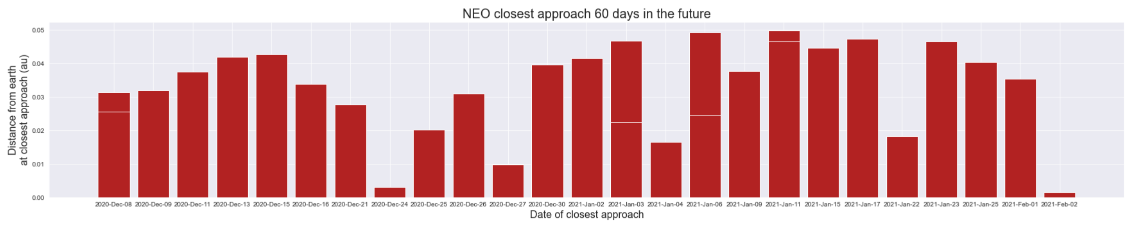
```
neo_60[['High Estimated Diameter (m)', 'Drop 2']] = neo_60.High.str.
 ↪split(expand=True)
neo_60 = neo_60.drop(columns=['Low','High','Drop 1','Drop 2','Diameter'])

#spliting the close approach distance columns and dropping the resulting␣
 ↪unneeded columns
neo_60[['CA Distance Nominal (LD)','CA Distance Nominal (au)']] = neo_60.
 ↪Distancenominalldau.str.split('|',expand=True)
neo_60[['CA Distance Minimum (LD)','CA Distance Minimum (au)']] = neo_60.
 ↪distanceminimumldau.str.split('|',expand=True)
neo_60 = neo_60.drop(columns=['CA Distance Nominal (LD)', 'CA Distance Minimum␣
 ↪(LD)','Distancenominalldau','distanceminimumldau'])

#converting the new columns to floating point type
neo_60 = neo_60.astype({'Low Estimated Diameter (m)': 'float64', 'High␣
 ↪Estimated Diameter (m)': 'float64', 'CA Distance Nominal (au)': 'float64',␣
 ↪'CA Distance Minimum (au)': 'float64', 'V relative (km/s)': 'float64', 'V␣
 ↪infinity (km/s)': 'float64'})

#calculating the average diameter of each NEO at close approach
avg_diameter = []
i=0
while i <= (len(neo_60['Low Estimated Diameter (m)'])-1):
    avg = ((neo_60.iloc[i,4])+(neo_60.iloc[i,5]))/2
    avg_diameter.append(avg)
    i+=1

neo_60['Average Estimated Diameter (m)'] = avg_diameter

#renaming columns so it is easier to call them
neo_60 = neo_60.rename(columns={'Close-Approach (CA) Date': 'approachdate'})

#splitting the date and time columns
neo_60[['Close Approach (CA) Date', 'approachtime', 'Uncertainty']] = neo_60.
 ↪approachdate.str.split(' ', expand=True)
neo_60[['Close Approach Time', 'drop']] = neo_60.approachtime.str.
 ↪split('±',expand=True)

#dropping the unneeded columns
neo_60 = neo_60.drop(columns = ['drop','approachdate', 'approachtime'])

neo_60.head()
```

[12]:

```
             V relative (km/s)  V infinity (km/s)  H (mag)  \
Object
(2018 PK21)               3.13               3.10     25.9
```

```
(2020 WK3)               6.90               6.89     25.1
(2020 XD)                9.14               9.13     25.6
(2020 VC7)               6.97               6.96     23.9
(2019 XQ1)               8.61               8.60     25.4

             Low Estimated Diameter (m)  High Estimated Diameter (m)  \
Object
(2018 PK21)                        18.0                         39.0
(2020 WK3)                         26.0                         58.0
(2020 XD)                          20.0                         45.0
(2020 VC7)                         44.0                         99.0
(2019 XQ1)                         22.0                         49.0

             CA Distance Nominal (au)  CA Distance Minimum (au)  \
Object
(2018 PK21)                   0.03129                   0.03129
(2020 WK3)                    0.02567                   0.02554
(2020 XD)                     0.03205                   0.03202
(2020 VC7)                    0.03756                   0.03745
(2019 XQ1)                    0.04703                   0.04190

             Average Estimated Diameter (m) Close Approach (CA) Date  \
Object
(2018 PK21)                            28.5              2020-Dec-08
(2020 WK3)                             42.0              2020-Dec-08
(2020 XD)                              32.5              2020-Dec-09
(2020 VC7)                             71.5              2020-Dec-11
(2019 XQ1)                             35.5              2020-Dec-13

            Uncertainty Close Approach Time
Object
(2018 PK21)      00:01               02:24
(2020 WK3)       00:01               08:09
(2020 XD)        00:01               08:33
(2020 VC7)       00:01               20:03
(2019 XQ1)        None               08:27
```

[13]:
```
#creating a bar graph to show the difference in distance of each close approach␣
↪over the next 60 days

plt.figure(figsize=(30,5))
plt.bar(neo_60.iloc[:,8],neo_60.iloc[:,6], color='firebrick')
plt.ylabel('Distance from earth\nat closest approach (au)',fontsize='16')
plt.xlabel('Date of closest approach', fontsize='16')
plt.title('NEO closest approach 60 days in the future', fontsize='20')
plt.savefig('NEO60daysbargraph.jpeg')
```



[14]:
```
#average densities of asteroids by class type (kg/m^3)
#I am using each class type because I was not able to find the mass/density of␣
 ↪each NEO either in the dataset I used
#or elsewhere. Using each class type will allow me to find the impact energy in␣
 ↪each possible scenario.
cdensity = 1.38*1000
sdensity = 2.71*1000
mdensity = 5.32*1000

#loop to calculate to approximate spherical volume of the NEO's (m^3)
spherical_volume = []
i = 0
while i <= (len(neo_1year['Average Estimated Diameter (m)'])-1):
    volume = (4/3)*3.14*(neo_1year.iloc[i,7]/2)**3
    spherical_volume.append(volume)
    i+=1
```

[15]:
```
#calculating the theoretical mass based off of average density, and spherical␣
 ↪volume
#mass is in kilograms

cmass = []
smass = []
mmass = []
i = 0
while i <= (len(spherical_volume)-1):
    c_mass = cdensity*spherical_volume[i]
    cmass.append(c_mass)
    s_mass = sdensity*spherical_volume[i]
    smass.append(s_mass)
    m_mass = mdensity*spherical_volume[i]
    mmass.append(m_mass)
    i+=1
```

[16]:
```
#calculating the theoretical kinetic energy of an impact assuming nothing␣
 ↪burned in the atmosphere.
```

```
c_kinetic_energy = []
s_kinetic_energy = []
m_kinetic_energy = []
i = 0
while i<= (len(cmass)-1):
    cenergy = (1/2)*cmass[i]*(neo_1year.iloc[i,0])**2
    c_kinetic_energy.append(cenergy)
    senergy = (1/2)*smass[i]*(neo_1year.iloc[i,0])**2
    s_kinetic_energy.append(senergy)
    menergy = (1/2)*mmass[i]*(neo_1year.iloc[i,0])**2
    m_kinetic_energy.append(menergy)
    i+=1

# adding columns for energy upon impact in the dataset
neo_1year['Impact Energy based on C Class (Joules)'] = c_kinetic_energy
neo_1year['Impact Energy based on S Class (Joules)'] = s_kinetic_energy
neo_1year['Impact Energy based on M Class (Joules)'] = m_kinetic_energy

neo_1year.head()
```

[16]:

```
             V relative (km/s)  V infinity (km/s)  H (mag)  \
Object
(2018 PK21)               3.13               3.10     25.9
(2020 WK3)                6.90               6.89     25.1
(2020 XD)                 9.14               9.13     25.6
(2020 VC7)                6.97               6.96     23.9
(2019 XQ1)                8.61               8.60     25.4

             Low Estimated Diameter (m)  High Estimated Diameter (m)  \
Object
(2018 PK21)                        18.0                         39.0
(2020 WK3)                         26.0                         58.0
(2020 XD)                          20.0                         45.0
(2020 VC7)                         44.0                         99.0
(2019 XQ1)                         22.0                         49.0

             CA Distance Nominal (au)  CA Distance Minimum (au)  \
Object
(2018 PK21)                   0.03129                   0.03129
(2020 WK3)                    0.02567                   0.02554
(2020 XD)                     0.03205                   0.03202
(2020 VC7)                    0.03756                   0.03745
(2019 XQ1)                    0.04703                   0.04190

             Average Estimated Diameter (m) Close Approach (CA) Date  \
Object
(2018 PK21)                            28.5              2020-Dec-08
```

```
(2020 WK3)                              42.0              2020-Dec-08
(2020 XD)                               32.5              2020-Dec-09
(2020 VC7)                              71.5              2020-Dec-11
(2019 XQ1)                              35.5              2020-Dec-13

            Close Approach Time  Impact Energy based on C Class (Joules)  \
Object
(2018 PK21)             02:24                              8.189375e+07
(2020 WK3)              08:09                              1.273719e+09
(2020 XD)               08:33                              1.035547e+09
(2020 VC7)              20:03                              6.412261e+09
(2019 XQ1)              08:27                              1.197619e+09

             Impact Energy based on S Class (Joules)  \
Object
(2018 PK21)                             1.608203e+08
(2020 WK3)                              2.501288e+09
(2020 XD)                               2.033575e+09
(2020 VC7)                              1.259219e+10
(2019 XQ1)                              2.351847e+09

             Impact Energy based on M Class (Joules)
Object
(2018 PK21)                             3.157063e+08
(2020 WK3)                              4.910278e+09
(2020 XD)                               3.992110e+09
(2020 VC7)                              2.471973e+10
(2019 XQ1)                              4.616910e+09
```

[17]:
```
#redefining these datasets so that they include the Kinetic Energy columns
wontburn = neo_1year[neo_1year['Average Estimated Diameter (m)']>=25]
#I used the average diameter value to stay consistent with the diameter value I␣
↪use througout the project.
small = neo_1year[neo_1year['Average Estimated Diameter (m)']<=25]
bigimpact = neo_1year[neo_1year['Average Estimated Diameter (m)']>=1000]
print(len(wontburn))
print(len(small))
print(len(bigimpact))
```

```
69
42
0
```

[18]:
```
#finding the largest and smallest NEO in the dataset
print(max(neo_1year['Average Estimated Diameter (m)']))
print(min(neo_1year['Average Estimated Diameter (m)']))
```

```
675.0
```

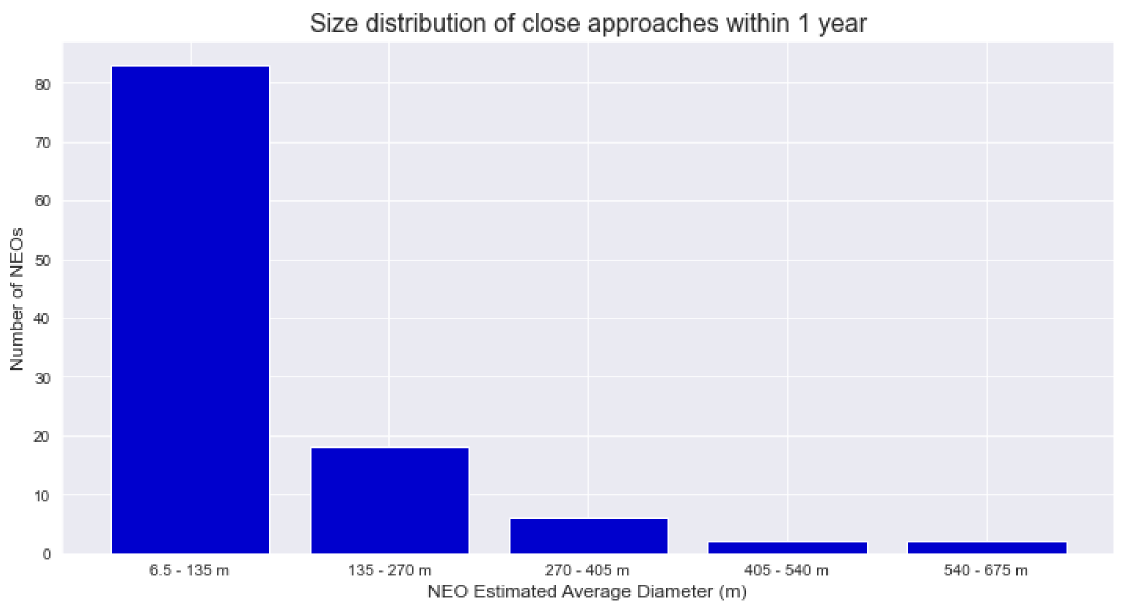
```
6.5
```

[19]:
```
#creating a loop to find how many NEO's are withing a particular size range
#the ranges were found by dividing up the max-min into five equal ranges.

count135 = 0 #initializing the ranges
count270 = 0
count405 = 0
count540 = 0
count675 = 0
i = 0

while i <= (len(neo_1year['Average Estimated Diameter (m)'])-1):  #setting the␣
 ↪loop
    if neo_1year.iloc[i,7] <= 135.0: #if it is less than 135 m in diameter add␣
 ↪one to that range
        count135 += 1
        i+=1
    elif neo_1year.iloc[i,7] > 135.0 and neo_1year.iloc[i,7] <= 270.0: #if it␣
 ↪is between 135 and 270 m add one to the range.
        count270 += 1
        i+=1
    elif neo_1year.iloc[i,7] > 270.0 and neo_1year.iloc[i,7] <= 405.0: #same␣
 ↪thing repeated for other ranges
        count405 += 1
        i+=1
    elif neo_1year.iloc[i,7] > 405.0 and neo_1year.iloc[i,7] <= 540.0:
        count540 += 1
        i+=1
    elif neo_1year.iloc[i,7] > 540.0 and neo_1year.iloc[i,7] <= 675.0:
        count675 += 1
        i+=1
    else:  # in case a value cannot be read correctly
        print('Error: Value is out of range', i)

sizelabels = ['6.5 - 135 m','135 - 270 m','270 - 405 m', '405 - 540 m','540 -␣
 ↪675 m'] # set range labels for bar plot
size = [count135,count270,count405,count540,count675] # save counts in a list␣
 ↪for plot
```

[20]:
```
# making sure the loop was able to calculate and sort all the values in the␣
 ↪column
print(count135+count270+count405+count540+count675)
```

```
111
```

[21]:
```
#plotting the distribution of sizes of the NEOs
plt.figure(figsize=(12,6))

plt.bar(sizelabels,size,color='mediumblue')
plt.xlabel('NEO Estimated Average Diameter (m)', fontsize='12')
plt.ylabel('Number of NEOs', fontsize='12')
plt.title('Size distribution of close approaches within 1 year', fontsize='16')

#saving the resulting figure
plt.savefig('sizedistribution.jpeg')
```



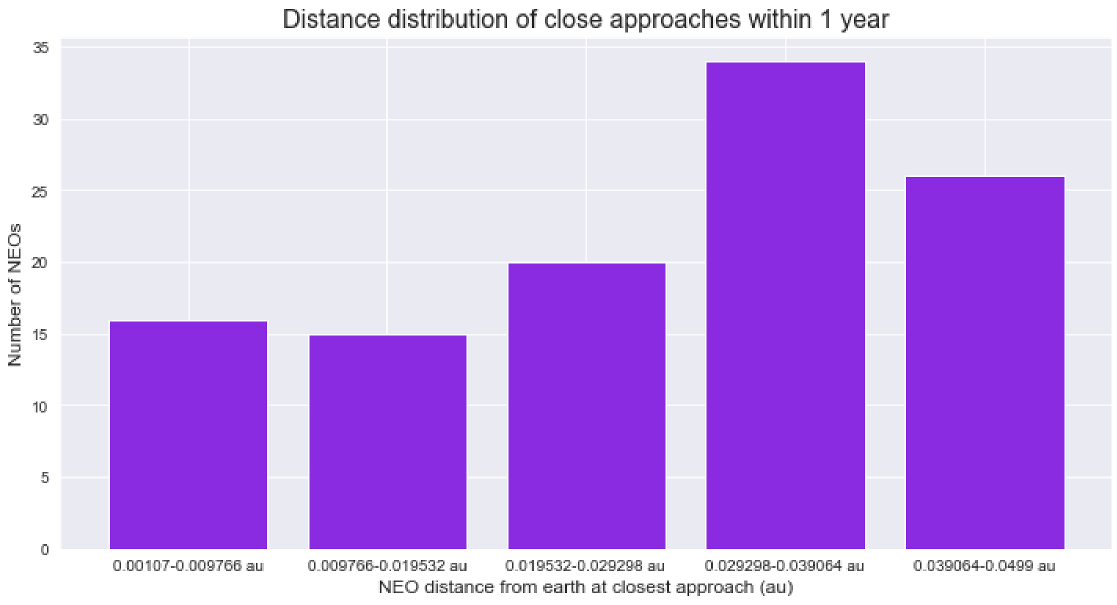
[22]:
```
#finding the closest and furthest distance from earth an NEO gets within a year

print(max(neo_1year['CA Distance Minimum (au)']))
print(min(neo_1year['CA Distance Minimum (au)']))
```

```
0.0499
0.00107
```

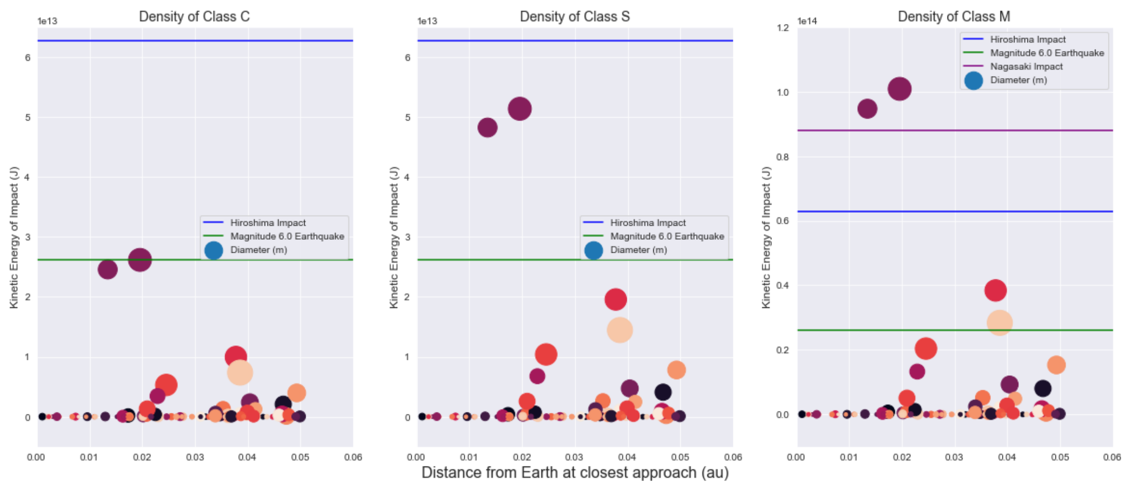
[23]:
```
#using the same method as above to find the distribution in distance from earth␣
 ↪at approach
#the ranges were found by dividing up the max-min into five equal ranges.

count_009766 = 0
count_019532 = 0
count_029298 = 0
count_039064 = 0
```

```
count_0499 = 0
i = 0

while i <= (len(neo_1year['CA Distance Minimum (au)'])-1):
    if neo_1year.iloc[i,6] <= 0.009766:
        count_009766 += 1
        i+=1
    elif neo_1year.iloc[i,6] > 0.009766 and neo_1year.iloc[i,6] <= 0.019532:
        count_019532 += 1
        i+=1
    elif neo_1year.iloc[i,6] > 0.019532 and neo_1year.iloc[i,6] <= 0.029298:
        count_029298 += 1
        i+=1
    elif neo_1year.iloc[i,6] > 0.029298 and neo_1year.iloc[i,6] <= 0.039064:
        count_039064 += 1
        i+=1
    elif neo_1year.iloc[i,6] > 0.039064 and neo_1year.iloc[i,6] <= 0.0499:
        count_0499 += 1
        i+=1
    else:
        print('Error: Index out of bounds')

distancelabels = ['0.00107-0.009766 au','0.009766-0.019532 au','0.019532-0.
 ↪029298 au', '0.029298-0.039064 au','0.039064-0.0499 au']
distancesize = [count_009766,count_019532,count_029298,count_039064,count_0499]
```

[24]:
```
#plotting the figure in the same way as above

plt.figure(figsize=(12,6))

plt.bar(distancelabels,distancesize,color='blueviolet')
plt.xlabel('NEO distance from earth at closest approach (au)', fontsize='12')
plt.ylabel('Number of NEOs', fontsize='12')
plt.title('Distance distribution of close approaches within 1 year',␣
 ↪fontsize='16')

plt.savefig('distancedistribution.jpeg')
```

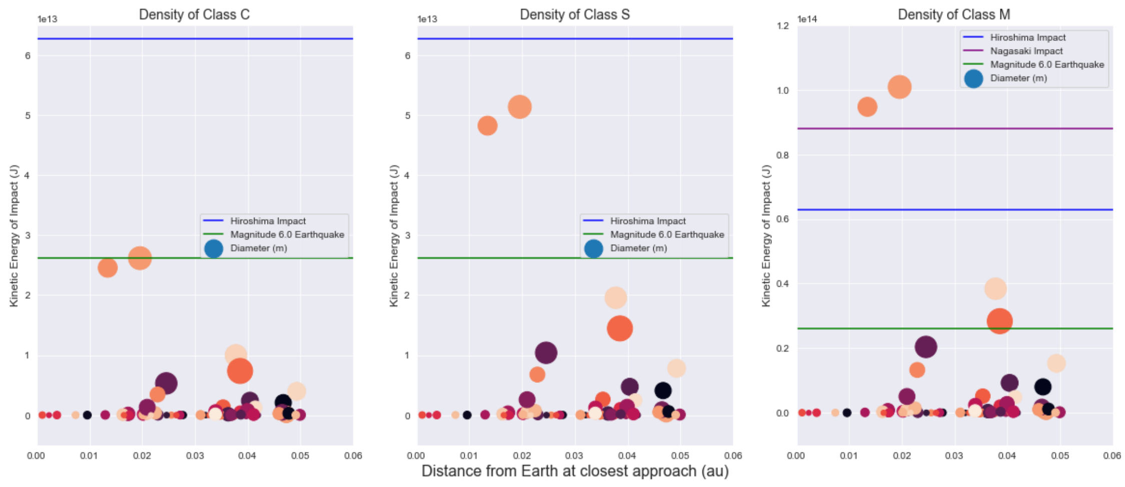


[25]:
```
#plotting the kinetic energy upon impact of each NEO at each class and␣
↪comparing them with other forces.

x = [0,.06] #initialize our x values for our comparison lines in the graph
littleboy_y = [(6.276*10**13),(6.276*10**13)] # the force of the Hiroshima␣
↪'Little Boy' Bomb (excludes the radiation impact)
earthquake6_y = [(2.6234*10**13),(2.6234*10**13)] # the force of a magnitude 6.
↪0 earthquake
fatman_y = [(8.786*10**13),(8.786*10**13)] # the force of the Nagasaki 'Fat␣
↪Man' Bomb (in Joules)
# each are converted to use as y coordinates

plt.figure(figsize=(20,8))

plt.subplot(1,3,1) #creating a subplot and designating its location
plt.scatter(neo_1year.iloc[:,6],neo_1year.iloc[:,10], s=neo_1year.iloc[:
↪,7],c=colors, label='Diameter (m)') #plotting the NEOs
plt.plot(x,littleboy_y,'b-', label='Hiroshima Impact') #plotting the hiroshima␣
↪impact
plt.plot(x,earthquake6_y,'g-', label='Magnitude 6.0 Earthquake') #plotting the␣
↪mag. 6.0 earthquake
plt.ylabel('Kinetic Energy of Impact (J)', fontsize='12')
plt.title('Density of Class C', fontsize='14') #titling it with its specific␣
↪density class
plt.legend()
plt.axis([0,0.06,(-.5*10**13),(6.5*10**13)]) #setting each axis to my desired␣
↪limits

plt.subplot(1,3,2) #creating the second subplot for density class S
plt.scatter(neo_1year.iloc[:,6],neo_1year.iloc[:,11], s=neo_1year.iloc[:
↪,7],c=colors, label='Diameter (m)')
plt.plot(x,littleboy_y,'b-', label='Hiroshima Impact')
plt.plot(x,earthquake6_y,'g-', label='Magnitude 6.0 Earthquake')
plt.ylabel('Kinetic Energy of Impact (J)', fontsize='12')
plt.title('Density of Class S', fontsize='14')
plt.legend()
plt.axis([0,0.06,(-.5*10**13),(6.5*10**13)])
plt.xlabel('Distance from Earth at closest approach (au)',fontsize='16')

plt.subplot(1,3,3) #creating the third subplot for density class M
plt.scatter(neo_1year.iloc[:,6],neo_1year.iloc[:,12], s=neo_1year.iloc[:
↪,7],c=colors, label='Diameter (m)')
plt.plot(x,littleboy_y,'b-', label='Hiroshima Impact')
plt.plot(x,earthquake6_y,'g-', label='Magnitude 6.0 Earthquake')
plt.plot(x,fatman_y,'purple', label='Nagasaki Impact')
plt.ylabel('Kinetic Energy of Impact (J)', fontsize='12')
plt.title('Density of Class M', fontsize='14')
plt.legend()
plt.axis([0,0.06,(-.1*10**14),(1.2*10**14)])

plt.savefig('energycomparisonall.jpeg') #saving the plot
```



[26]:
```
#creating the same plot as above, only exluding the NEOs that would burn up in␣
↪the atmosphere.
```

```
plt.figure(figsize=(20,8))

plt.subplot(1,3,1)
plt.scatter(wontburn.iloc[:,6],wontburn.iloc[:,10], s=wontburn.iloc[:
 ↪,7],c=shortcolors, label='Diameter (m)')
plt.plot(x,littleboy_y,'b-', label='Hiroshima Impact')
plt.plot(x,earthquake6_y,'g-', label='Magnitude 6.0 Earthquake')
plt.ylabel('Kinetic Energy of Impact (J)', fontsize='12')
plt.title('Density of Class C', fontsize='14')
plt.legend()
plt.axis([0,0.06,(-.5*10**13),(6.5*10**13)])

plt.subplot(1,3,2)
plt.scatter(wontburn.iloc[:,6],wontburn.iloc[:,11], s=wontburn.iloc[:
 ↪,7],c=shortcolors, label='Diameter (m)')
plt.plot(x,littleboy_y,'b-', label='Hiroshima Impact')
plt.plot(x,earthquake6_y,'g-', label='Magnitude 6.0 Earthquake')
plt.ylabel('Kinetic Energy of Impact (J)', fontsize='12')
plt.title('Density of Class S', fontsize='14')
plt.legend()
plt.axis([0,0.06,(-.5*10**13),(6.5*10**13)])
plt.xlabel('Distance from Earth at closest approach (au)',fontsize='16')

plt.subplot(1,3,3)
plt.scatter(wontburn.iloc[:,6],wontburn.iloc[:,12], s=wontburn.iloc[:
 ↪,7],c=shortcolors, label='Diameter (m)')
plt.plot(x,littleboy_y,'b-', label='Hiroshima Impact')
plt.plot(x,fatman_y,'purple', label='Nagasaki Impact')
plt.plot(x,earthquake6_y,'g-', label='Magnitude 6.0 Earthquake')
plt.ylabel('Kinetic Energy of Impact (J)', fontsize='12')
plt.title('Density of Class M', fontsize='14')
plt.legend()
plt.axis([0,0.06,(-.1*10**14),(1.2*10**14)])

plt.savefig('energycomparisonpdangerous.jpeg')
```



```
[27]: #creating a new dataframe that only includes the two NEOs with the largest␣
       ↪impacts
      top_2 = neo_1year[neo_1year['Impact Energy based on C Class␣
       ↪(Joules)']>(2*10**13)]
      top_2.head()
```

```
[27]:                     V relative (km/s)  V infinity (km/s)  H (mag)  \
      Object
      (1999 RM45)                     20.04              20.03     19.4
      231937 (2001 FO32)              34.41              34.41     17.7

                          Low Estimated Diameter (m)  High Estimated Diameter (m)  \
      Object
      (1999 RM45)                              350.0                        780.0
      231937 (2001 FO32)                       770.0                          1.7

                          CA Distance Nominal (au)  CA Distance Minimum (au)  \
      Object
      (1999 RM45)                          0.01959                   0.01958
      231937 (2001 FO32)                   0.01348                   0.01347

                          Average Estimated Diameter (m) Close Approach (CA) Date  \
      Object
      (1999 RM45)                                 565.00              2021-Mar-02
      231937 (2001 FO32)                          385.85              2021-Mar-21

                         Close Approach Time  \
      Object
      (1999 RM45)                      19:52
      231937 (2001 FO32)               16:03
```

```
                     Impact Energy based on C Class (Joules)  \
Object
(1999 RM45)                                      2.615582e+13
231937 (2001 FO32)                               2.456136e+13

                     Impact Energy based on S Class (Joules)  \
Object
(1999 RM45)                                      5.136396e+13
231937 (2001 FO32)                               4.823281e+13

                     Impact Energy based on M Class (Joules)
Object
(1999 RM45)                                      1.008326e+14
231937 (2001 FO32)                               9.468581e+13
```

[28]:
```
#saving the values of each impact/force
hiroshima = 6.276*10**13
earthquake = 2.6234*10**13
fatman = 8.786*10**13

#calculating how much more impactful NEO 1999 RM45 would be than a mag. 6.0␣
 ↪earthquake at density class M
mearthquake = (top_2.iloc[0,12]/earthquake)

print('NEO 1999 RM45 at density class M would create an impact',␣
 ↪round(mearthquake,2), 'times that of a magnitude 6.0 earthquake.')
```

```
NEO 1999 RM45 at density class M would create an impact 3.84 times that of a
magnitude 6.0 earthquake.
```

[29]:
```
#creating a function that will covert distance in astronomical units to miles
def automi(au):
    miles = (9.296*10**7)*au
    return miles
```

[30]:
```
#calculating the percentage of NEOs that come close to earth
p = ((count_009766+count_019532)/111)*100
percent = round(p,1) #rounding the percentage value so it is not too long
dinmiles = automi(.019532) #converting the distance from au to mi
print('Only', percent, '% of NEOs approaching earth in the next year will come␣
 ↪within .01953 astronomical units or',dinmiles,'miles of the earth.')
```

```
Only 27.9 % of NEOs approaching earth in the next year will come within .01953
astronomical units or 1815694.72 miles of the earth.
```

[31]:
```
#calculating what percentage of the NEOs would be able to make it through the␣
 ↪atmosphere
```

```
pburn = (len(small)/len(neo_1year))*100
percentburn = round(pburn,1)
pwontburn = (len(wontburn)/len(neo_1year))*100
percentwontburn = round(pwontburn, 1)

print(percentburn, '% of the NEOs will burn up completely in the atmosphere.␣
↪This means that', percentwontburn, '% of NEOs with close approaches within␣
↪the next year have the potential to reach the ground.')
```

```
37.8 % of the NEOs will burn up completely in the atmosphere. This means that
62.2 % of NEOs with close approaches within the next year have the potential to
reach the ground.
```

[32]:
```
p135 = ((count270+count405+count540+count675)/111)*100
percent135 = round(p135,1)

print(percent135, '% of the total NEO close approaches in the next year will be␣
↪larger than 135 meters in diameter.')
```

```
25.2 % of the total NEO close approaches in the next year will be larger than
135 meters in diameter.
```

Toutatis Data:

https://en.wikipedia.org/wiki/4179_Toutatis

https://www.spacereference.org/asteroid/4179-toutatis-1989-ac

[33]:
```
#inserting the toutatis data
#i chose toutatis to use as a comparison to the other NEOs because we have a␣
↪lot of information about it that we dont
#have on other asteroids/NEOs. It's listed as a potentially hazardous object␣
↪because it could produce significant damage
#to the earth upon impact, but poses no threat because its orbit does not␣
↪approach the earth close enough to raise alarms.

tou_mass = 5.05*10**13 #mass of asteroid/NEO Toutatis in kg
tou_diameter =  5400 #the mean diameter of Toutatis in m
tou_density = 2100 #the mean density of Toutatis in kg/m^3. Toutatis is␣
↪believed to be an S-type Asteroid.
tou_velocity = 8.406 #the velocity of Toutatis relative to the gravitational␣
↪pull of the earth at its close approach in 2069 in km/s
tou_distance = .02 #distance from earth at close approach in 2069 in au.

tou_impactenergy = (1/2)*tou_mass*(tou_velocity**2)

print('The energy from an impact from Toutatis in 2069 is equal to',␣
↪tou_impactenergy, 'Joules')
```

```
The energy from an impact from Toutatis in 2069 is equal to 1784186109000000.0
Joules
```

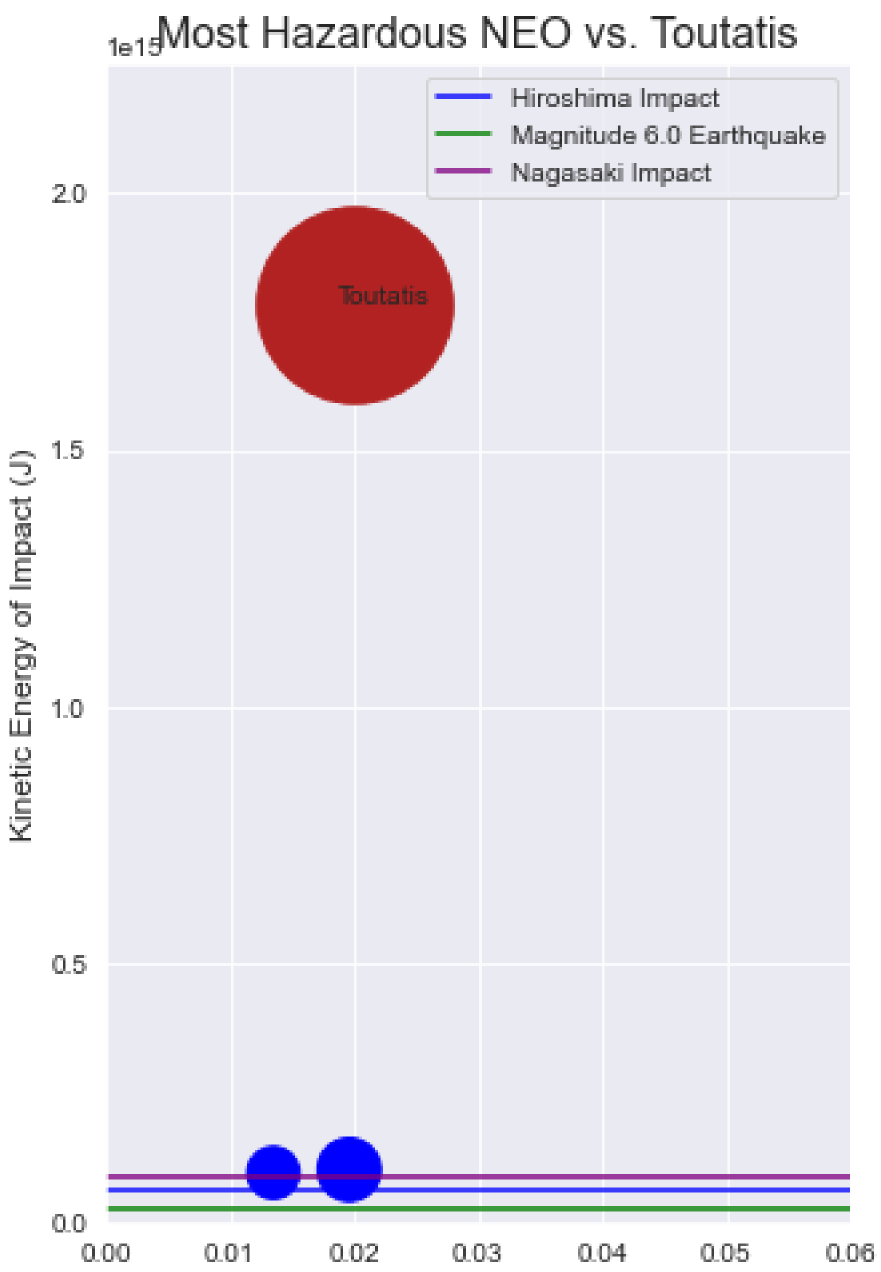
[34]:
```
top_2
```

[34]:
```
                       V relative (km/s)  V infinity (km/s)  H (mag)  \
Object
(1999 RM45)                        20.04              20.03     19.4
231937 (2001 FO32)                 34.41              34.41     17.7

                       Low Estimated Diameter (m)  High Estimated Diameter (m)  \
Object
(1999 RM45)                                 350.0                        780.0
231937 (2001 FO32)                          770.0                          1.7

                       CA Distance Nominal (au)  CA Distance Minimum (au)  \
Object
(1999 RM45)                             0.01959                   0.01958
231937 (2001 FO32)                      0.01348                   0.01347

                       Average Estimated Diameter (m) Close Approach (CA) Date  \
Object
(1999 RM45)                                    565.00              2021-Mar-02
231937 (2001 FO32)                             385.85              2021-Mar-21

                      Close Approach Time  \
Object
(1999 RM45)                        19:52
231937 (2001 FO32)                 16:03

                       Impact Energy based on C Class (Joules)  \
Object
(1999 RM45)                                       2.615582e+13
231937 (2001 FO32)                                2.456136e+13

                       Impact Energy based on S Class (Joules)  \
Object
(1999 RM45)                                       5.136396e+13
231937 (2001 FO32)                                4.823281e+13

                       Impact Energy based on M Class (Joules)
Object
(1999 RM45)                                       1.008326e+14
231937 (2001 FO32)                                9.468581e+13
```

[35]:
```
#plotting the asteroid Toutais at its approach in 2069 against the top 2 most␣
 ↪hazardous NEOs at close approach
```

```
#in the next year.

plt.figure(figsize=(5,8))

plt.scatter(tou_distance, tou_impactenergy, s=tou_diameter, color='firebrick')
plt.scatter(top_2.iloc[:,6], top_2.iloc[:,12], s=top_2.iloc[:,7],c='blue')
plt.plot(x,littleboy_y,'b-', label='Hiroshima Impact')
plt.plot(x,earthquake6_y,'g-', label='Magnitude 6.0 Earthquake')
plt.plot(x,fatman_y,'purple', label='Nagasaki Impact')
plt.ylabel('Kinetic Energy of Impact (J)', fontsize='12')
plt.title('Most Hazardous NEO vs. Toutatis', fontsize='16')
plt.annotate(s='Toutatis', xy=(.0185, tou_impactenergy)) #labeling the toutatis␣
 ↪asteroid on the graph
plt.legend()
plt.axis([0,0.06,(-.5*10**13),(2.25*10**15)])

plt.savefig('toutatiscomparison.jpeg')
```

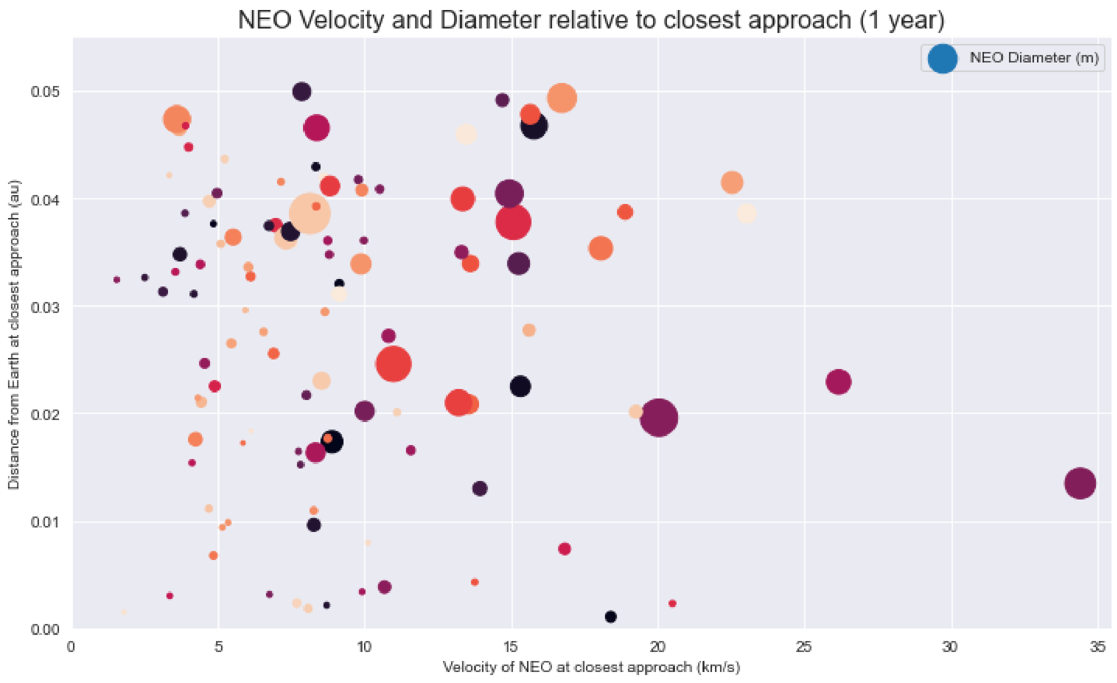
```
<ipython-input-35-2ea1986b687c>:13: MatplotlibDeprecationWarning: The 's'
parameter of annotate() has been renamed 'text' since Matplotlib 3.3; support
for the old name will be dropped two minor releases later.
  plt.annotate(s='Toutatis', xy=(.0185, tou_impactenergy)) #labeling the
toutatis asteroid on the graph
```



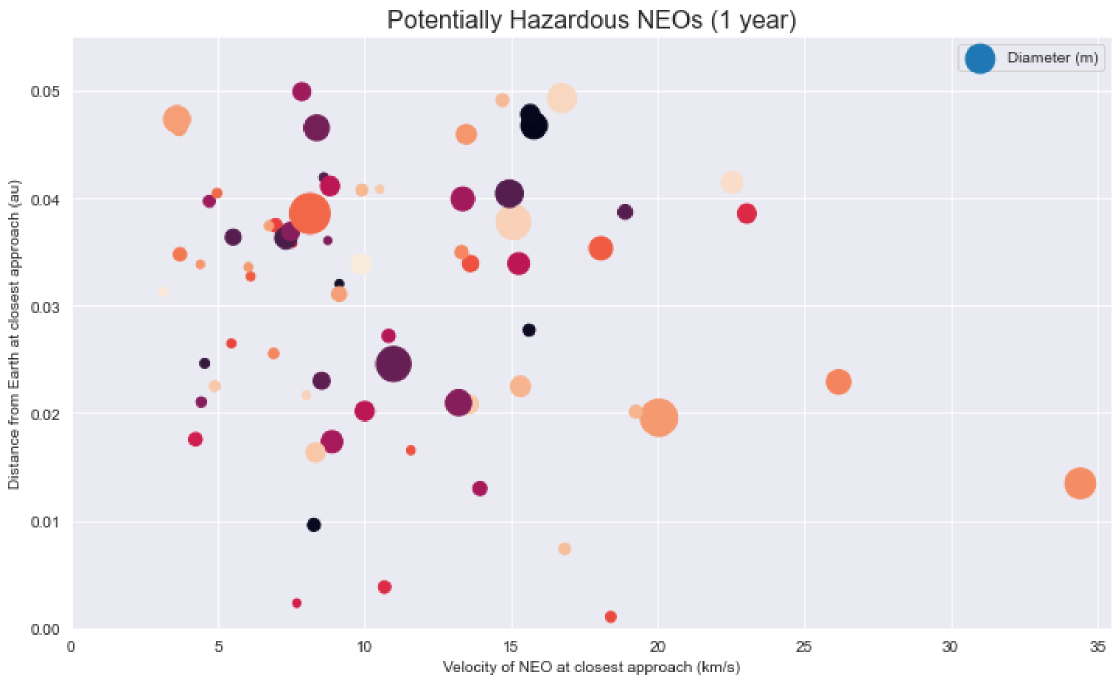
At this point in my project I would have liked to look at NEO close approaches further in the future to see if there were any larger threats in the near future (ten years from now). I also would have liked to explore more visualizations of the data I have gathered and calculated in this project.

## 1.3 Results

[36]:
```
#plotting distance from earth at closest approach vs.velocity at closest␣
 ↪approach
#the size of the points is relative to the diameter of each NEO
plt.figure(figsize = (12,7))
sns.set_style('darkgrid')
plt.scatter(neo_1year.iloc[:,0],neo_1year.iloc[:,6], s=neo_1year.iloc[:
 ↪,7],c=colors, label='NEO Diameter (m)')
plt.xlabel('Velocity of NEO at closest approach (km/s)')
plt.ylabel('Distance from Earth at closest approach (au)')
plt.axis([0,35.5,0,.055])
plt.title('NEO Velocity and Diameter relative to closest approach (1 year)',␣
 ↪fontsize='16')
plt.legend()

plt.savefig('1yeardiameter.jpeg')
```

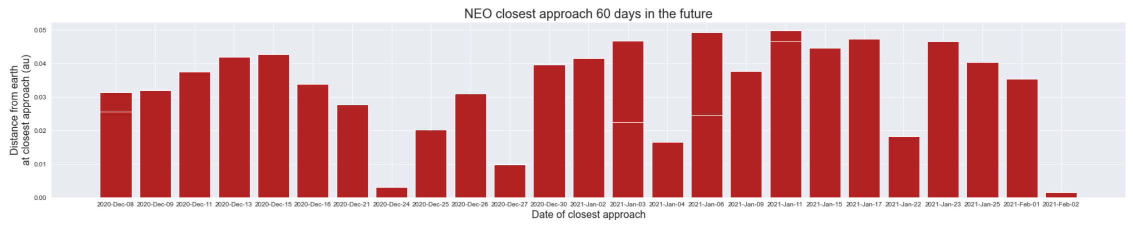


[37]:
```
# plotting the distance, velocity and diameter of NEO's larger than 25 meters␣
 ↪in diameter.
plt.figure(figsize = (12,7))
sns.set_style('darkgrid')
plt.scatter(wontburn.iloc[:,0],wontburn.iloc[:,6], s=wontburn.iloc[:
 ↪,7],c=shortcolors, label='Diameter (m)')
plt.xlabel('Velocity of NEO at closest approach (km/s)')
plt.ylabel('Distance from Earth at closest approach (au)')
plt.axis([0,35.5,0,.055])
plt.title('Potentially Hazardous NEOs (1 year)', fontsize='16')
plt.legend()

plt.savefig('1yeardiameterdangerous.jpeg')
```



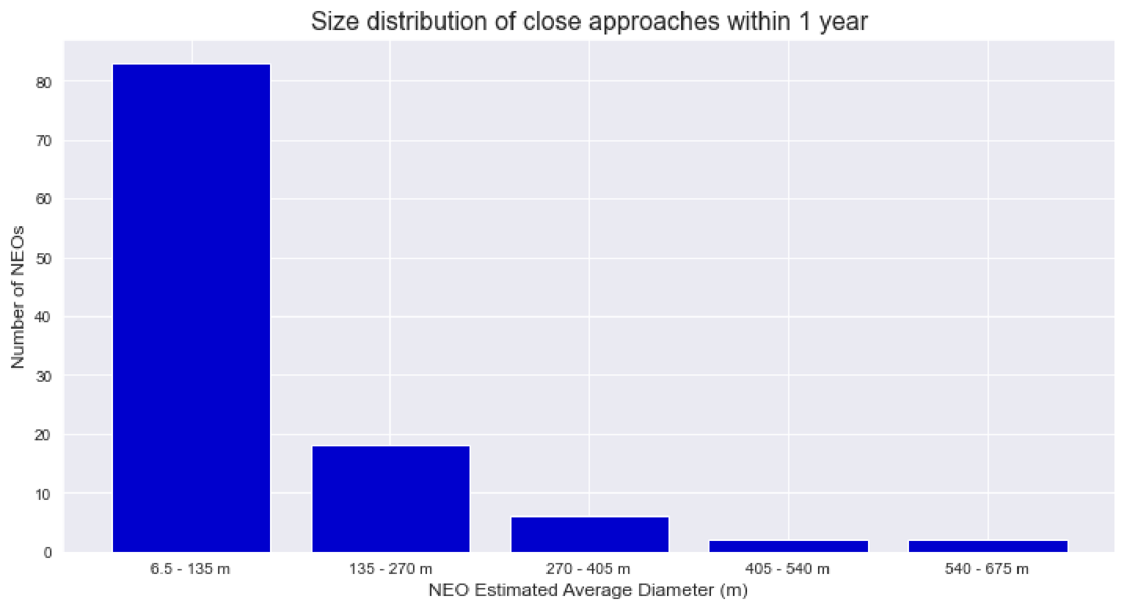
This graph is the result of my combing through the data. It shows the distance each NEO will be from earth at its closest approach, compared to the velocity of the NEO at that time. It also illustrates the diameter of each NEO relative to each other.

[38]:
```
#creating a bar graph to show the difference in distance of each close approach␣
 ↪over the next 60 days

plt.figure(figsize=(30,5))
plt.bar(neo_60.iloc[:,8],neo_60.iloc[:,6], color='firebrick')
plt.ylabel('Distance from earth\nat closest approach (au)',fontsize='16')
plt.xlabel('Date of closest approach', fontsize='16')
plt.title('NEO closest approach 60 days in the future', fontsize='20')

plt.savefig('NEO60daysbargraph.jpeg')
```



This graph shows the closest approaches we will experience over the next 60 days.

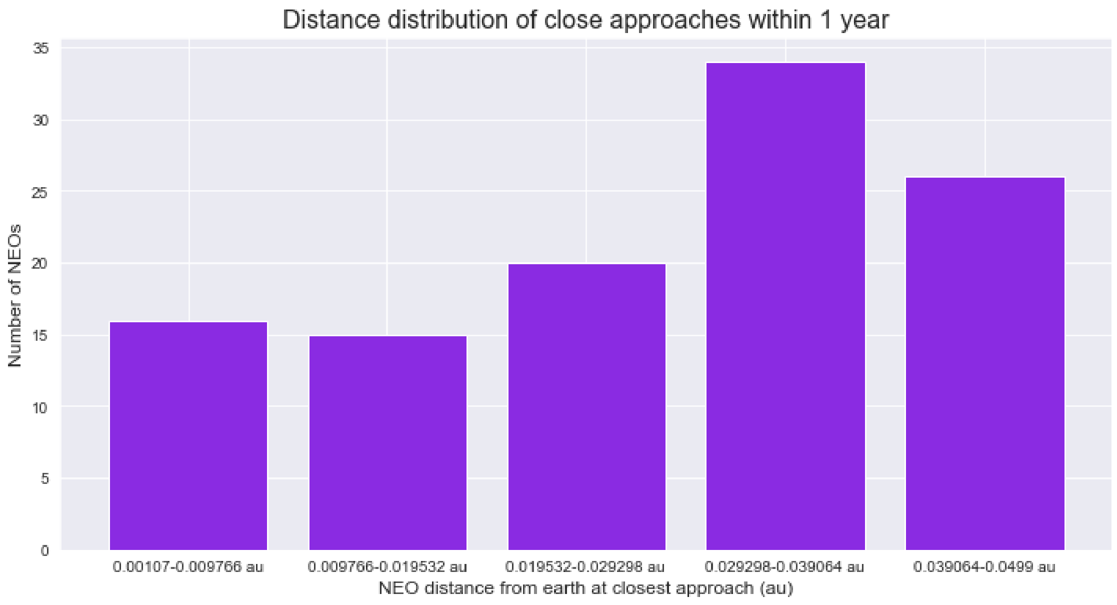
[39]:
```
#plotting the distribution of sizes of the NEOs
plt.figure(figsize=(12,6))

plt.bar(sizelabels,size,color='mediumblue')
plt.xlabel('NEO Estimated Average Diameter (m)', fontsize='12')
plt.ylabel('Number of NEOs', fontsize='12')
plt.title('Size distribution of close approaches within 1 year', fontsize='16')

#saving the resulting figure
plt.savefig('sizedistribution.jpeg')
```



This graph shows the amount of NEO close approaches in the next year that are within a certain size range

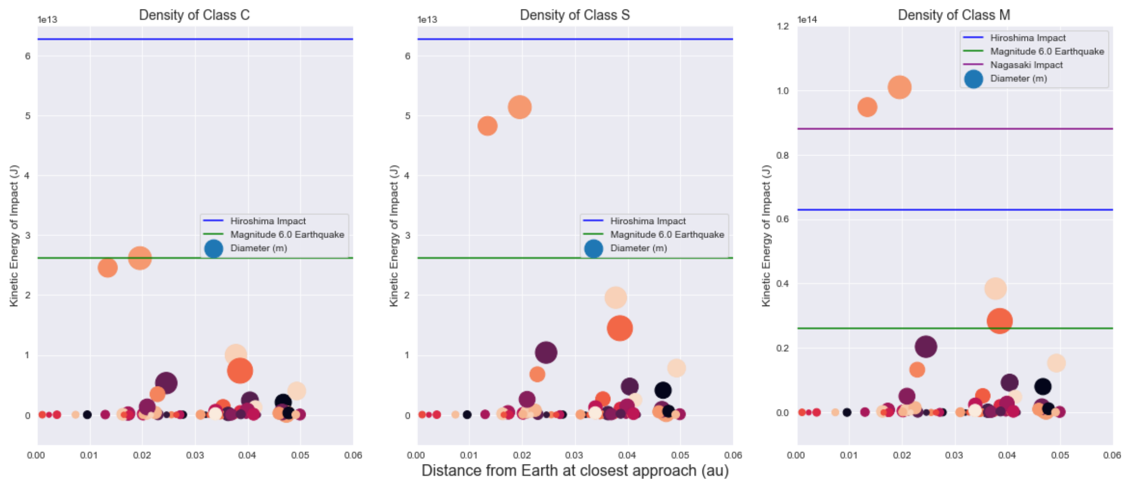
[40]:
```
#plotting the figure in the same way as above
```

```
plt.figure(figsize=(12,6))

plt.bar(distancelabels,distancesize,color='blueviolet')
plt.xlabel('NEO distance from earth at closest approach (au)', fontsize='12')
plt.ylabel('Number of NEOs', fontsize='12')
plt.title('Distance distribution of close approaches within 1 year',␣
↪fontsize='16')

plt.savefig('distancedistribution.jpeg')
```



This graph illustates the distribution of NEOs and their distance from earth at closest approach

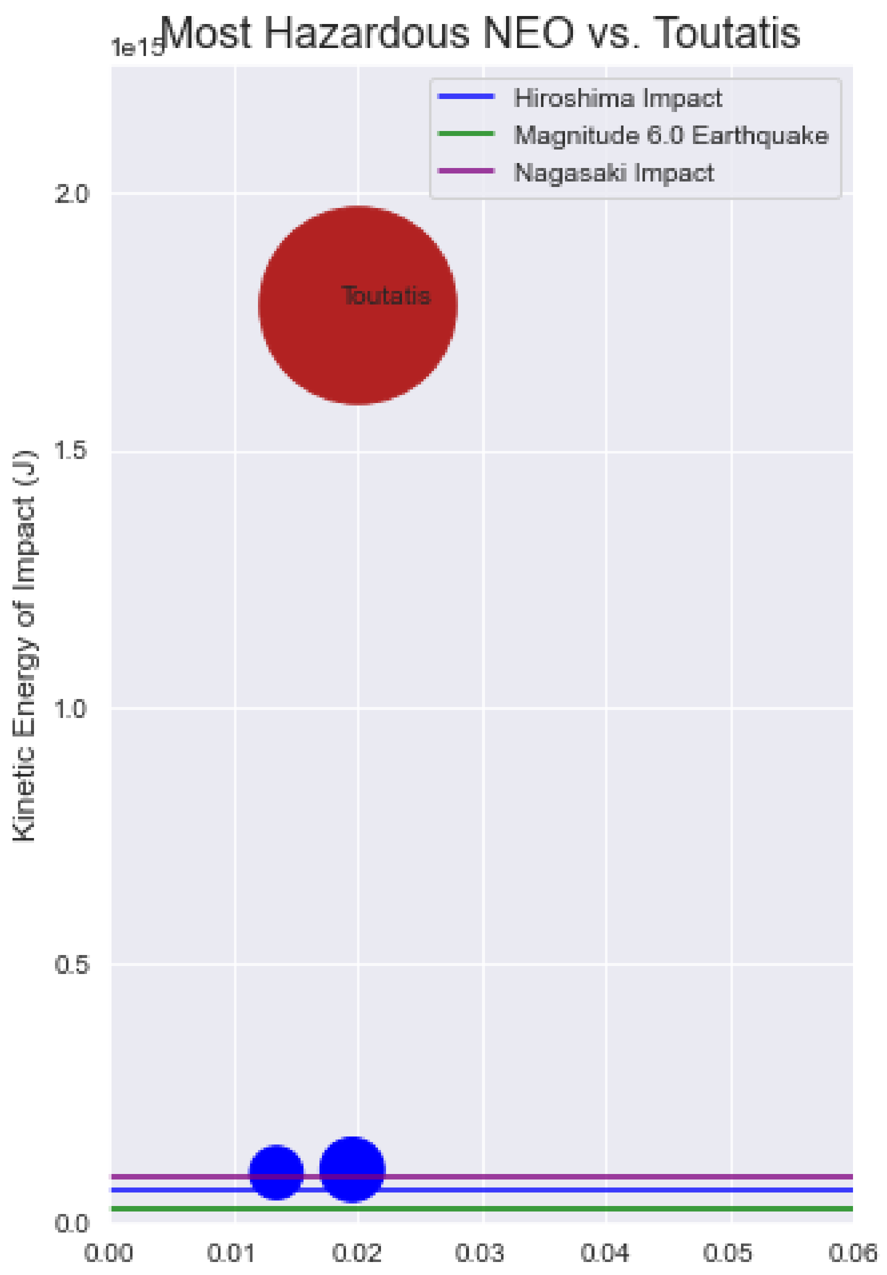
[41]:
```
#creating the same plot as above, only exluding the NEOs that would burn up in␣
↪the atmosphere.

plt.figure(figsize=(20,8))

plt.subplot(1,3,1)
plt.scatter(wontburn.iloc[:,6],wontburn.iloc[:,10], s=wontburn.iloc[:
↪,7],c=shortcolors, label='Diameter (m)')
plt.plot(x,littleboy_y,'b-', label='Hiroshima Impact')
plt.plot(x,earthquake6_y,'g-', label='Magnitude 6.0 Earthquake')
plt.ylabel('Kinetic Energy of Impact (J)', fontsize='12')
plt.title('Density of Class C', fontsize='14')
plt.legend()
plt.axis([0,0.06,(-.5*10**13),(6.5*10**13)])
plt.subplot(1,3,2)
plt.scatter(wontburn.iloc[:,6],wontburn.iloc[:,11], s=wontburn.iloc[:
 ↪,7],c=shortcolors, label='Diameter (m)')
plt.plot(x,littleboy_y,'b-', label='Hiroshima Impact')
plt.plot(x,earthquake6_y,'g-', label='Magnitude 6.0 Earthquake')
plt.ylabel('Kinetic Energy of Impact (J)', fontsize='12')
plt.title('Density of Class S', fontsize='14')
plt.legend()
plt.axis([0,0.06,(-.5*10**13),(6.5*10**13)])
plt.xlabel('Distance from Earth at closest approach (au)',fontsize='16')

plt.subplot(1,3,3)
plt.scatter(wontburn.iloc[:,6],wontburn.iloc[:,12], s=wontburn.iloc[:
 ↪,7],c=shortcolors, label='Diameter (m)')
plt.plot(x,littleboy_y,'b-', label='Hiroshima Impact')
plt.plot(x,earthquake6_y,'g-', label='Magnitude 6.0 Earthquake')
plt.plot(x,fatman_y,'purple', label='Nagasaki Impact')
plt.ylabel('Kinetic Energy of Impact (J)', fontsize='12')
plt.title('Density of Class M', fontsize='14')
plt.legend()
plt.axis([0,0.06,(-.1*10**14),(1.2*10**14)])

plt.savefig('energycomparisonpdangerous.jpeg')
```



This graph shows the impact that each of the NEO's would have were they to collide with the earth. There are varying levels of density because the dataset I was able to find did not include mass or density values, and I was not able to find that data elsewhere. Therefore, I decided to calculate the outcomes of multiple different scenarios.

```
[42]: #plotting the asteroid Toutais at its approach in 2069 against the top 2 most␣
       ↪hazardous NEOs at close approach
      #in the next year.

      plt.figure(figsize=(5,8))

      plt.scatter(tou_distance, tou_impactenergy, s=tou_diameter, color='firebrick')
      plt.scatter(top_2.iloc[:,6], top_2.iloc[:,12], s=top_2.iloc[:,7],c='blue')
      plt.plot(x,littleboy_y,'b-', label='Hiroshima Impact')
      plt.plot(x,earthquake6_y,'g-', label='Magnitude 6.0 Earthquake')
      plt.plot(x,fatman_y,'purple', label='Nagasaki Impact')
      plt.ylabel('Kinetic Energy of Impact (J)', fontsize='12')
      plt.title('Most Hazardous NEO vs. Toutatis', fontsize='16')
      plt.annotate(s='Toutatis', xy=(.0185, tou_impactenergy)) #labeling the toutatis␣
       ↪asteroid on the graph
      plt.legend()
      plt.axis([0,0.06,(-.5*10**13),(2.25*10**15)])

      plt.savefig('toutatiscomparison.jpeg')
```

```
<ipython-input-42-2ea1986b687c>:13: MatplotlibDeprecationWarning: The 's'
parameter of annotate() has been renamed 'text' since Matplotlib 3.3; support
for the old name will be dropped two minor releases later.
  plt.annotate(s='Toutatis', xy=(.0185, tou_impactenergy)) #labeling the
toutatis asteroid on the graph
```



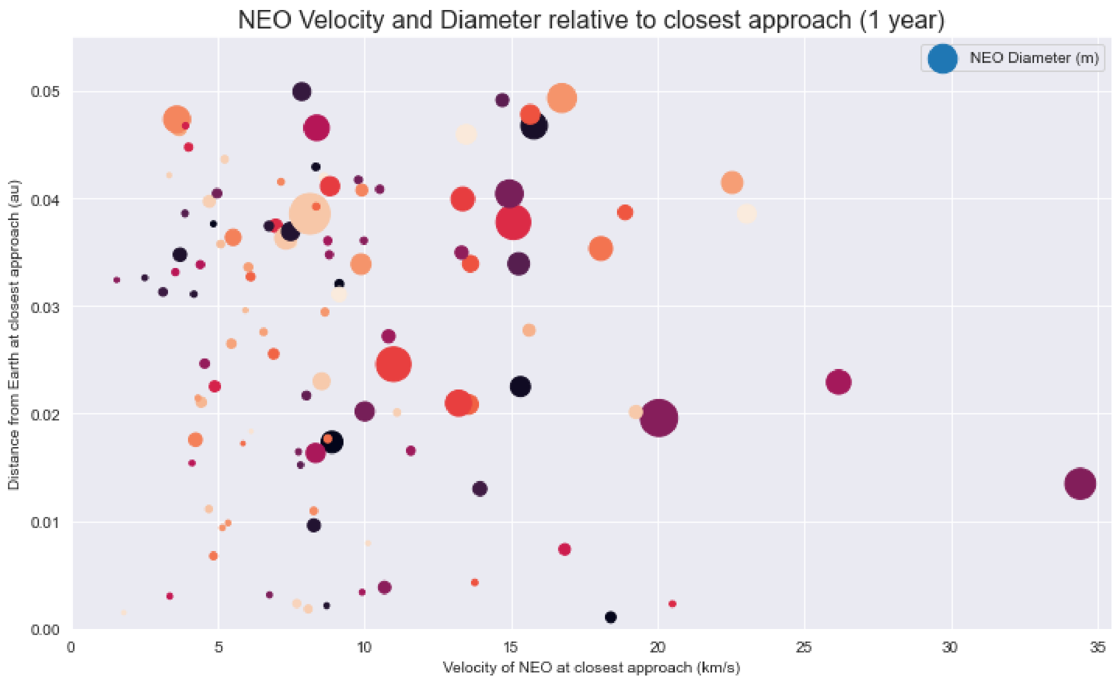
This is the data of two NEO's that would create the most damage upon impact graphed against the asteroid Toutatis. It shows just how big an asteroid would have to be in order to create regional damage, let alone worldwide damage.

## 1.4 Discussions and Conclusions

One of the very first problems I ran into with this project was finding sufficient and up-to-date data. I ended up finding a couple of good data sets, however the data and the format itself were too old to be combatable with my project. This was one of the main issues I had with figuring out what question I wanted to solve. Once I found a good dataset I was able to sift through the data and find something that interested me. How an impact from an NEO would affect the earth.

```
[43]: plt.figure(figsize = (12,7))
      sns.set_style('darkgrid')
      plt.scatter(neo_1year.iloc[:,0],neo_1year.iloc[:,6], s=neo_1year.iloc[:
       ↪,7],c=colors, label='NEO Diameter (m)')
      plt.xlabel('Velocity of NEO at closest approach (km/s)')
      plt.ylabel('Distance from Earth at closest approach (au)')
      plt.axis([0,35.5,0,.055])
      plt.title('NEO Velocity and Diameter relative to closest approach (1 year)',␣
       ↪fontsize='16')
      plt.legend()
```

```
[43]: <matplotlib.legend.Legend at 0x7fbc07563a90>
```

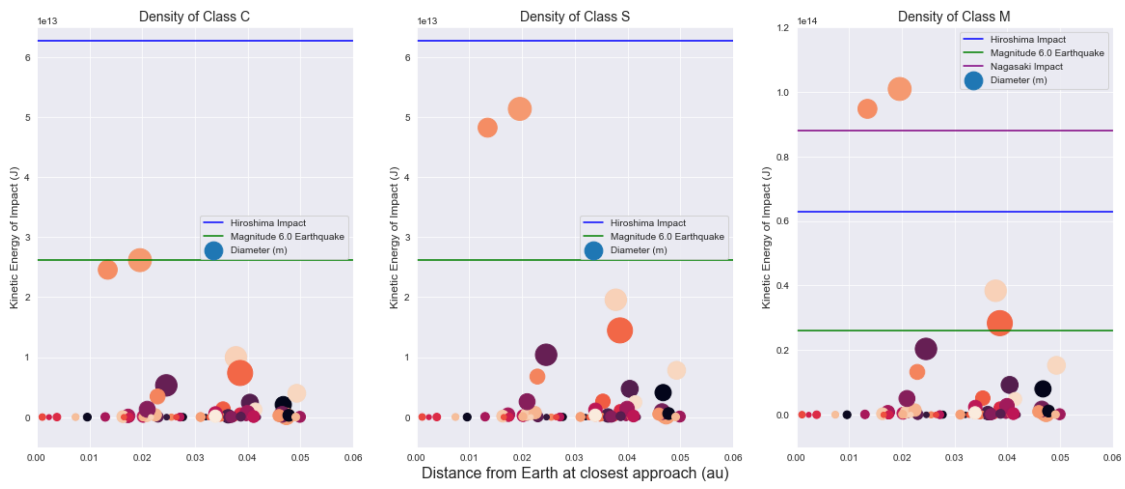


In the end I was able to learn a lot from this project. One of the things I found particularly interesting came about in the first graph I was able to produce. I could not help but notice the correlation between each aspect of the data on this graph. For example, it appears as though the closer the NEO gets to the earth, the smaller and slower it gets. There are also very few NEO's of small diameter that exceed 15 km/s at their closest approach. This created a lot more questions I would have loved to delve much deeper into. However, many of them require data that I do not have access to. When it came time to calculate the energy upon impact I realized that the data set I had found was lacking quite a bit that I would need. Therefore, a lot of the calculations are theoretical estimates.

The bar graph of size distribution in the NEO's show that the majority of them are very small. Another one of the bar graphs shows that most of the NEO's are farther than .195 NEO's away from earth at their closest approach. About 42 of the NEO's would burn up in the atmosphere if their orbits shifted enough to allow it. Out of the remaining 69 NEO's the majority of them would be so small they create little damage at best, if not they would get past the atmosphere as little more than dust. The graph below shows that data:

[44]:
```
plt.figure(figsize=(20,8))

plt.subplot(1,3,1)
plt.scatter(wontburn.iloc[:,6],wontburn.iloc[:,10], s=wontburn.iloc[:
 ↪,7],c=shortcolors, label='Diameter (m)')
plt.plot(x,littleboy_y,'b-', label='Hiroshima Impact')
plt.plot(x,earthquake6_y,'g-', label='Magnitude 6.0 Earthquake')
plt.ylabel('Kinetic Energy of Impact (J)', fontsize='12')
plt.title('Density of Class C', fontsize='14')
plt.legend()
plt.axis([0,0.06,(-.5*10**13),(6.5*10**13)])

plt.subplot(1,3,2)
plt.scatter(wontburn.iloc[:,6],wontburn.iloc[:,11], s=wontburn.iloc[:
 ↪,7],c=shortcolors, label='Diameter (m)')
plt.plot(x,littleboy_y,'b-', label='Hiroshima Impact')
plt.plot(x,earthquake6_y,'g-', label='Magnitude 6.0 Earthquake')
plt.ylabel('Kinetic Energy of Impact (J)', fontsize='12')
plt.title('Density of Class S', fontsize='14')
plt.legend()
plt.axis([0,0.06,(-.5*10**13),(6.5*10**13)])
plt.xlabel('Distance from Earth at closest approach (au)',fontsize='16')

plt.subplot(1,3,3)
plt.scatter(wontburn.iloc[:,6],wontburn.iloc[:,12], s=wontburn.iloc[:
 ↪,7],c=shortcolors, label='Diameter (m)')
plt.plot(x,littleboy_y,'b-', label='Hiroshima Impact')
plt.plot(x,earthquake6_y,'g-', label='Magnitude 6.0 Earthquake')
plt.plot(x,fatman_y,'purple', label='Nagasaki Impact')
plt.ylabel('Kinetic Energy of Impact (J)', fontsize='12')
plt.title('Density of Class M', fontsize='14')
plt.legend()
plt.axis([0,0.06,(-.1*10**14),(1.2*10**14)])
```

[44]:
```
(0.0, 0.06, -10000000000000.0, 120000000000000.0)
```



This data compares the energies of different events to that of the impact of NEO's approachings in the next year. As you can see there are two NEO's that would cause the most damage if they were to collide with the earth. The one that would cause the most damage is 1999 RM45. If it were to collide with the earth and it had a density of class M, the impact would be 3.844 times more energetic than a 6.0 magnitude earthquake.

Here are some of the other statistics I was able to calculate during this project:

Only 27.9 % of NEOs approaching earth in the next year will come within .01953 astronomical units or 1815694.72 miles of the earth.

25.2 % of the total NEO close approaches in the next year will be larger than 135 meters in diameter.

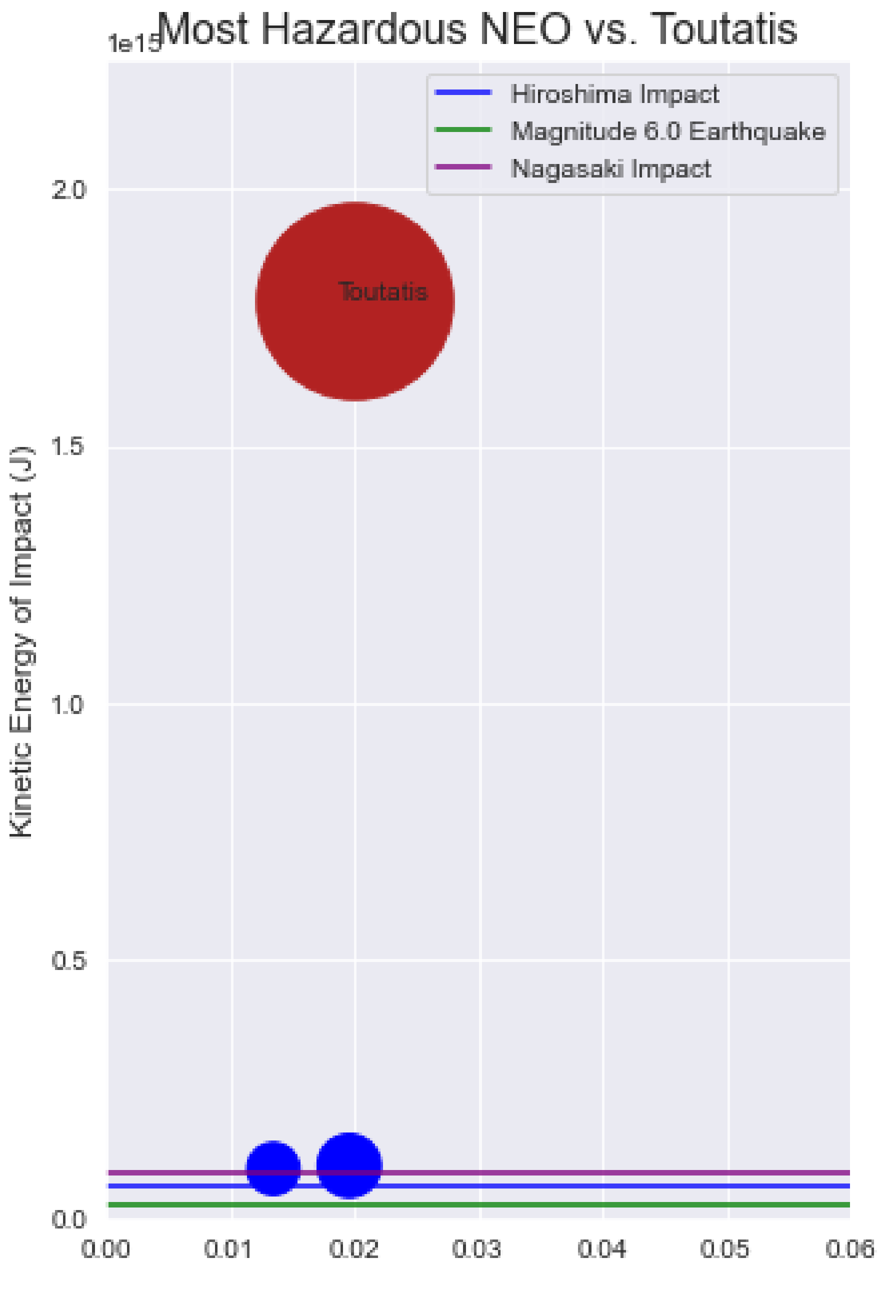
37.8 % of the NEOs will burn up completely in the atmosphere. This means that 62.2 % of NEOs with close approaches within the next year have the potential to reach the ground.

During this project I was also able to take some more definite data about a large asteroid and compare it to the data of close approaches within the next year. I chose the asteroid Toutatis to do this because it is large enough to create regional damage to the planet upon impact. Here are those comparisons:

[45]:
```
plt.figure(figsize=(5,8))

plt.scatter(tou_distance, tou_impactenergy, s=tou_diameter, color='firebrick')
plt.scatter(top_2.iloc[:,6], top_2.iloc[:,12], s=top_2.iloc[:,7],c='blue')
plt.plot(x,littleboy_y,'b-', label='Hiroshima Impact')
plt.plot(x,earthquake6_y,'g-', label='Magnitude 6.0 Earthquake')
plt.plot(x,fatman_y,'purple', label='Nagasaki Impact')
plt.ylabel('Kinetic Energy of Impact (J)', fontsize='12')
plt.title('Most Hazardous NEO vs. Toutatis', fontsize='16')
plt.annotate(s='Toutatis', xy=(.0185, tou_impactenergy)) #labeling the toutatis␣
 ↪asteroid on the graph
plt.legend()
plt.axis([0,0.06,(-.5*10**13),(2.25*10**15)])
```

```
<ipython-input-45-75a5999ad9a8>:10: MatplotlibDeprecationWarning: The 's'
parameter of annotate() has been renamed 'text' since Matplotlib 3.3; support
for the old name will be dropped two minor releases later.
  plt.annotate(s='Toutatis', xy=(.0185, tou_impactenergy)) #labeling the
toutatis asteroid on the graph
```

```
[45]: (0.0, 0.06, -5000000000000.0, 2250000000000000.0)
```



This graph goes to show that while a possible impact by NEO's over the next year would cause damage, it would ve very local to the impact area and would not be close to an extinction level event. Therefore over the next year, there is not much to be worried about. It is also important to note that the probability of one of these NEO's impacting the earth is very very slim. As I had stated in my background, there's only about one asteroid a year large enough to create a fireball in the atmosphere, and those tend to burn up completely before reaching the ground. In conclusion, the answers to my questions are this:

1. There are 69 NEO's that will be potential hazardous in the next year because they will not burn up in the atmosphere in the case of an impact.
2. The two NEO's who would cause the most damage, at worst case scenario, would be 3.84 times more energetic than a magnitude 6.0 earthquake, but less energetic than the H Bomb.

If I were to do anything differently in this project I would spend more time trying to find accurate data. That was a big problem of mine throughout the project. I would also like to have allotted more time to exploring some of the other interesting concepts that came up in the project such as how large the impact craters would be and investigating the correlation mentioned in the first graph.

## 1.5 Resources

"Scatter Plot¶." Scatter Plot - Matplotlib 3.3.3 Documentation, matplotlib.org/3.3.3/gallery/shapes_and_collections/scatter.html.

Curious2learnCurious2learn 24.5k3535 gold badges9494 silver badges120120 bronze badges, et al. "Setting y-Axis Limit in Matplotlib." Stack Overflow, 1 Nov. 1959, stackoverflow.com/questions/3777861/setting-y-axis-limit-in-matplotlib.

"Scatter Plot¶." Scatter Plot - Matplotlib 3.3.3 Documentation, matplotlib.org/3.3.3/gallery/shapes_and_collections/scatter.html.

"NEO Basics." NASA, NASA, cneos.jpl.nasa.gov/about/basics.html.

"Matplotlib - Bar Plot." Tutorialspoint, www.tutorialspoint.com/matplotlib/matplotlib_bar_plot.htm.

"In Depth." NASA, NASA, 19 Dec. 2019, solarsystem.nasa.gov/asteroids-comets-and-meteors/asteroids/in-depth/.

"Standard Asteroid Physical Characteristics." Wikipedia, Wikimedia Foundation, 22 Jan. 2020, en.wikipedia.org/wiki/Standard_asteroid_physical_characteristics.

Contributors, HowStuffWorks.com. "How Much Energy in a Hurricane, a Volcano, and an Earthquake?" HowStuffWorks Science, HowStuffWorks, 27 Jan. 2020, science.howstuffworks.com/environmental/energy/energy-hurricane-volcano-earthquake3.htm.

"Little Boy." Wikipedia, Wikimedia Foundation, 1 Dec. 2020, en.wikipedia.org/wiki/Little_Boy.

"NEO Earth Close Approaches." NASA, NASA, cneos.jpl.nasa.gov/ca/.

Dunbar, Brian. "Asteroid Fast Facts." NASA, NASA, 24 Mar. 2015, www.nasa.gov/mission_pages/asteroids/overview/fastfacts.html.

"4179 Toutatis." Wikipedia, Wikimedia Foundation, 29 Sept. 2020, en.wikipedia.org/wiki/4179_Toutatis.

"Toutatis." Asteroid Toutatis | Space Reference, www.spacereference.org/asteroid/4179-toutatis-1989-ac.

"Nuclear Weapon Yield." Wikipedia, Wikimedia Foundation, 12 Dec. 2020, en.wikipedia.org/wiki/Nuclear_weapon_yield.

```
[ ]:
```