"Data Science Analysis on Exo-planetary Data"

Source : Inspired by this presentation!

REPORT BY : KOIDALA SURYA PRAKASH _ EE18BTECH11026
TEJASREE GUNTURU _ EE20BTECH11049

INTRODUCTION:

Any planet beyond our solar system is termed as an exoplanet. These exoplanets are free floating and do not orbit other stars. They orbit the galactic center. The composition of the exoplanets can be determined by measuring their sizes(diameters) and masses(weights), thereby classifying them from rocky to gas-rich. The first exoplanets were discovered in the 1990s and since then thousands were identified using a variety of detection methods. The information (dataset) of such discovered exoplanets can be found here

OBJECTIVE:

Based on the planetary mass and orbital period, the K-Means clustering method should be used to classify three large groups of exoplanets: ***Hot Jupiters*** (a class of gas giant exoplanets that are inferred to be physically similar to Jupiter but that have very short orbital periods ($P < 10$ days), ***Long Period Giants*** *and* ***Small Planets***.

In order to take into account more planetary and stellar parameters, Uniform Manifold Approximation and Projection (UMAP) technique is used to visualize data on a 2D map, aiming to find structures within the high dimensional parameter space. Finally, we explore the relation between the chosen parameters.

METHODS:

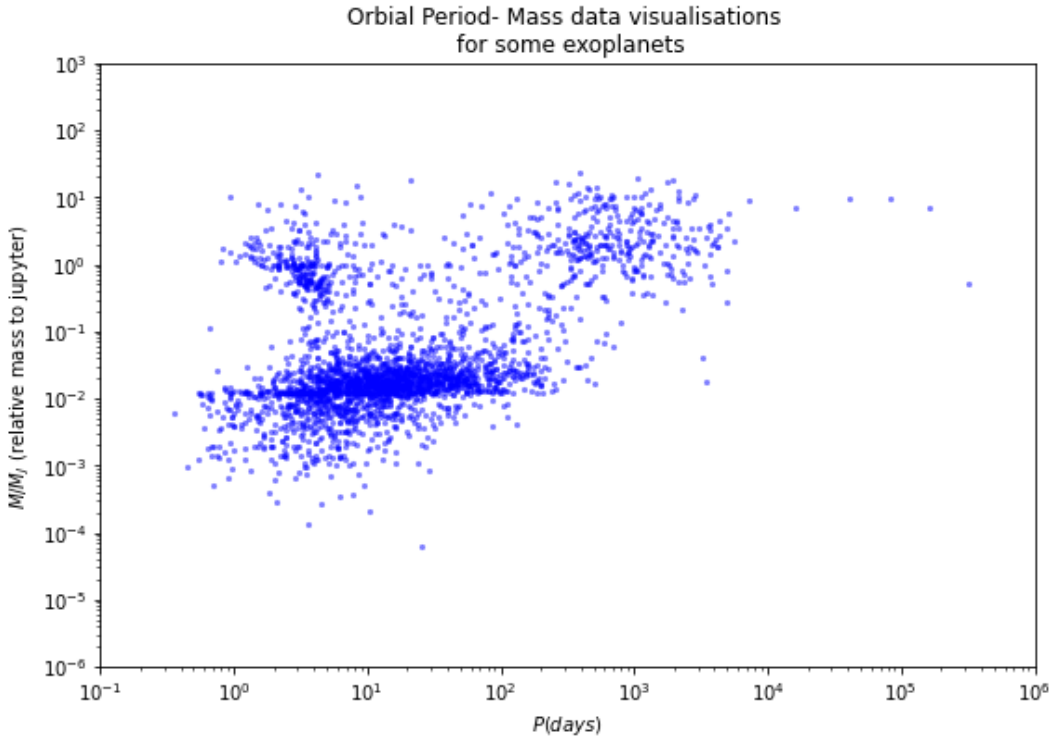
PART-1:

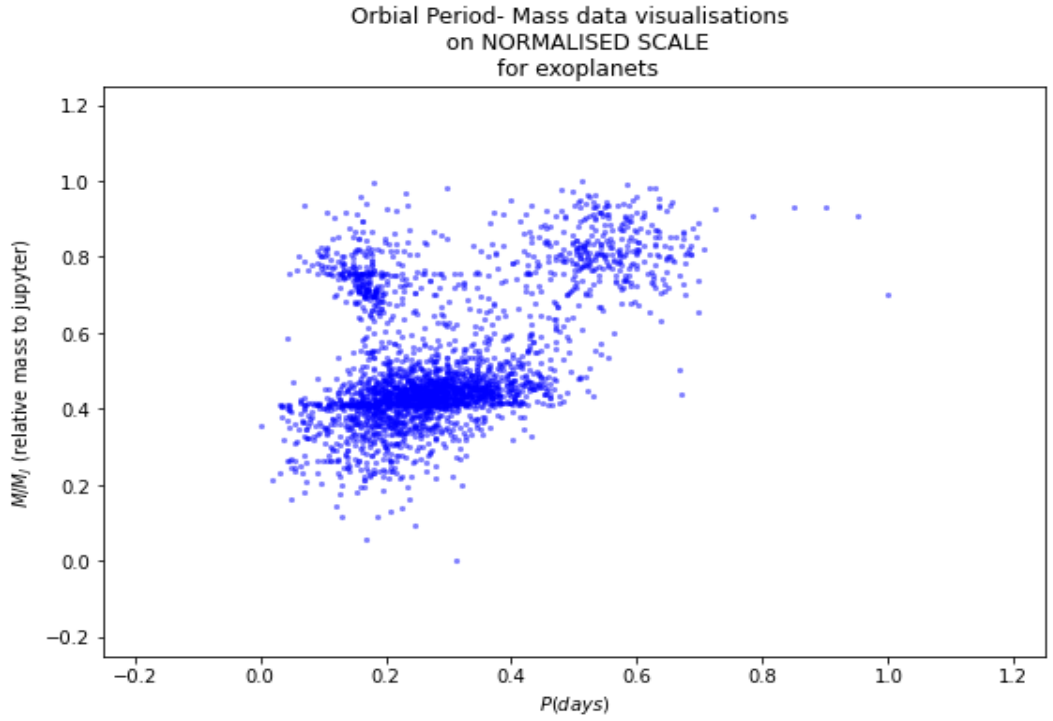
- **K-Means Clustering:**

It is a type of unsupervised learning applied to unlabeled data (data without defined categories or groups). This technique automatically labels the groups within the data those of which are visibly distinct from each other.

K-Means follows a simple procedure where the given data set is classified into a number of clusters, defined by the letter "k," which is fixed beforehand. The clusters are then positioned as points and all observations or data points are associated with the nearest cluster, computed, adjusted and then the process starts over using the new adjustments until a desired result is reached.

K-means clustering has uses in search engines, market segmentation, statistics and even astronomy.





From the above graphs, we can observe the data being segregated into three clusters, which provides us with an idea to group them into three different categories.

In this project K-Means Clustering is used in an attempt to resemble a plot of planetary mass as a function of the orbital period on logarithmic scale (where, data belonging to the three different groups can be observed), thereby clustering the data into the following three groups:

- ➢ Hot Jupiters: M ~ $1M_J$, P ~ 3 days.
- ➢ Long Period Giants: M ~ $1M_J$, P > 100 days.
- ➢ Non-Giants: all the other exoplanets.

Contributions : Implemented K-means algorithm from scratch and visualized the clusters and classified them into 3 specific groups

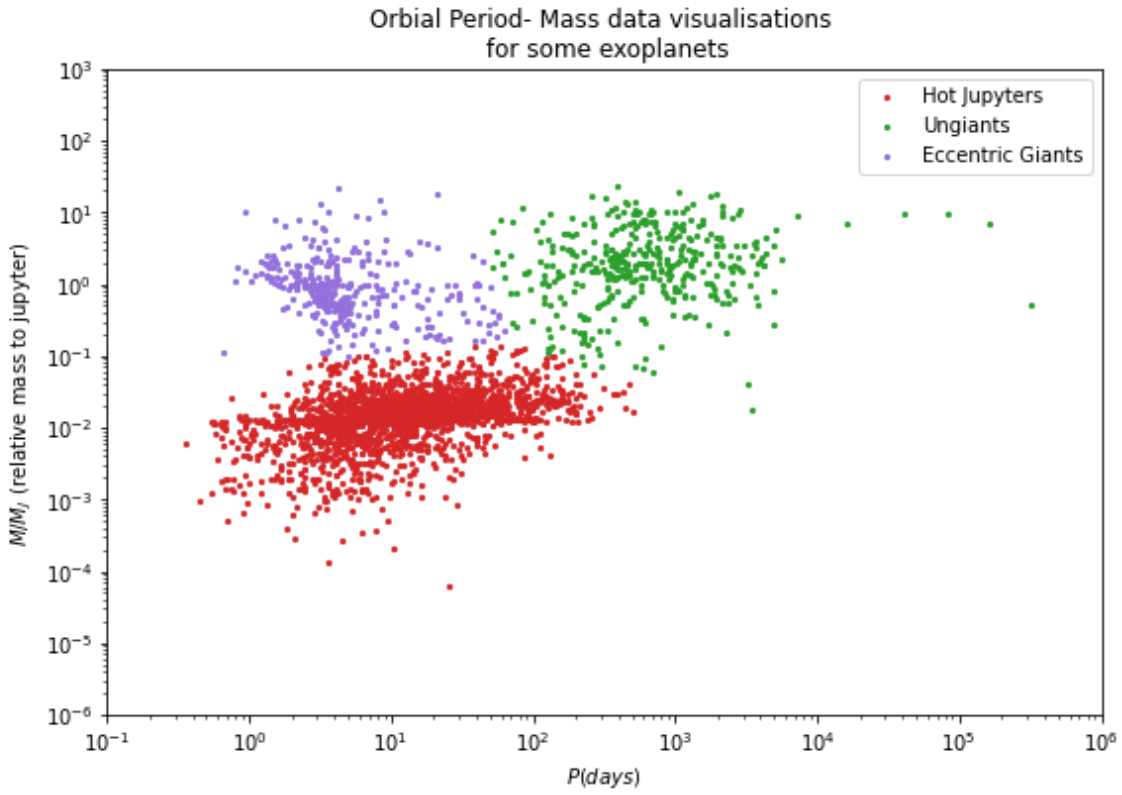
PSEUDO CODE:

for $k = 1$ **to** K **do**
  $\mu_k \leftarrow$ some random location // randomly initialize mean for kth cluster
end for
repeat
  for $n = 1$ **to** N **do**
    $z_n \leftarrow \text{argmin}_k ||\mu_k - x_n||$ // assign example n to closest center
  end for
  for $k = 1$ **to** K **do**
    $\mu_k \leftarrow \text{MEAN}(\{ x_n : z_n = k \})$ // re-estimate mean of cluster k
  end for
until converged
return z // return cluster assignments

OUR IMPLEMENTATION:

```
def kmeans_scratch(data,K, iter_no):
    idx = np.random.choice(len(data), K, replace=False)

    '''
    Step 01 : randomly initialise mean for kth cluster
    '''
    centroids = data[idx, :]

    '''
    Step 02 : assigning n closest centers by finding the distance
    '''

    #finding the distance between centroids and all the data points
    distances = cdist(data, centroids ,'euclidean')
```

```
    #Centroid with the minimum Distance
    points = np.array([np.argmin(i) for i in distances])

    #Repeating the above steps for a defined number of iterations
    for _ in range(iter_no):
        centroids = []
        for idx in range(K):

            '''
            Step 03 :  re-estimate mean of cluster k
            '''
            temp_cent = data[points==idx].mean(axis=0)
            centroids.append(temp_cent)

        centroids = np.vstack(centroids) #Updated Centroids

        distances = cdist(data, centroids ,'euclidean')
        points = np.array([np.argmin(i) for i in distances])

    '''
    Step 04 : return cluster assigments
    '''
    return points
```



Hence, the exoplanets have been classified into three clusters using K-Means.

PART-2:

- **Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP):**

UMAP is a novel manifold learning technique for dimension reduction. It is constructed from a theoretical framework based in Riemannian geometry and algebraic topology and has no computational restrictions on embedding dimension, making it viable as a general purpose dimension reduction technique for machine learning. Dimensionality Reduction helps in effective visualization of high dimensional data.

UMAP was chosen for dimensionality reduction because it is based on a mathematical foundation that allows for the preservation of global and local structure so that the following features are satisfied:
1. Distance between points is a measure of similarity.
2. It enables the search of correlations between groups and features.

On further exploring the non giants (Dimensionality reduction using UMAP), the following groups are obtained:

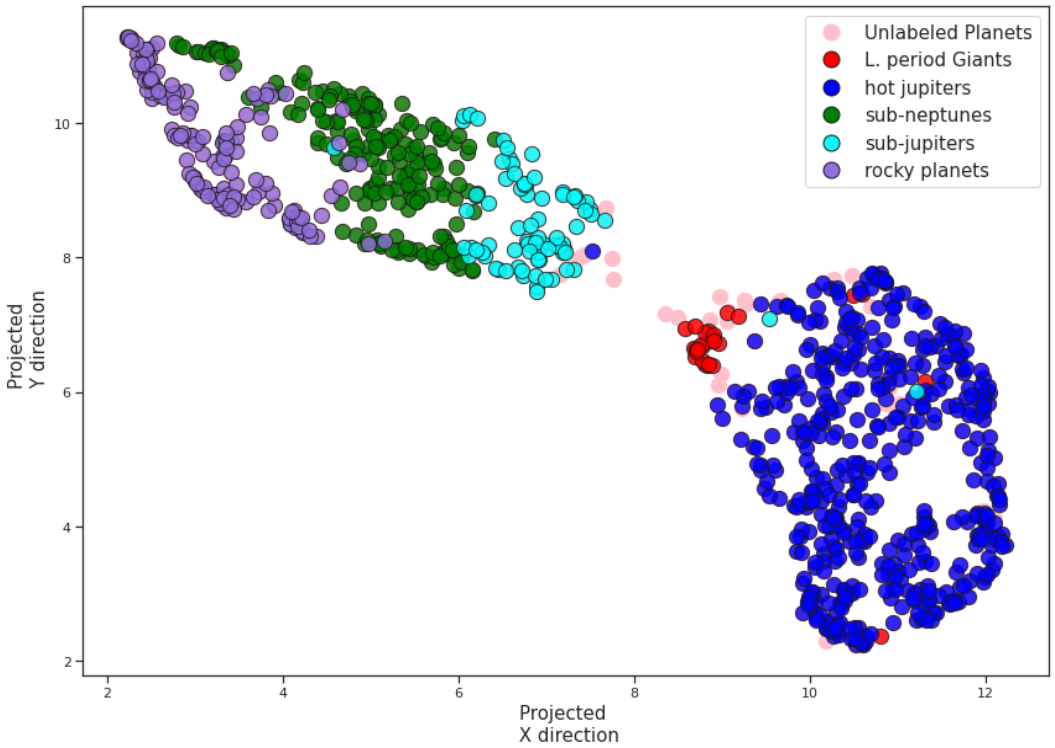
- ➢ Rocky Planets
- ➢ Sub-Neptunes
- ➢ Sub-Jupiters
- ➢ Longer Period Giants
- ➢ Hot Jupiters

In this project, we have chosen six input parameters (the least amount of planetary features that allow for a fuller classification, so as to maximize the amount of data points)
to cluster exoplanetary groups using UMAP. They are:

M* **:** Stellar Mass (Mass of the Star)
R* **:** Stellar Radius (Radius of the star)
T* **:** Stellar Temperature (Temperature of the star)
Mp **:** Planetary Mass
Rp **:** Planetary Radius
P **:** Orbital Period of the Planet

```
### Umap visualisation ...

data_EU2 = data_EU[params2]
x = data_EU2.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
data_US_scaled = pd.DataFrame(x_scaled)
```

```
new_data = data_EU[params2]

### min max scaling
x = new_data.values
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
new_data_scaled = pd.DataFrame(x_scaled)

### UMAP transformation
umap_trans = umap.UMAP(n_neighbors=80, min_dist=0.05, metric='euclidean')
embedding_scaled = umap_trans.fit_transform(new_data_scaled)
```



However, K-Means cannot be used to look for the extraplanetary types, since it requires more parameters, which is why UMAP is used.

Exploring the relationship between the parameters:

M* : Stellar Mass (Mass of the Star)
R* : Stellar Radius (Radius of the star)
T* : Stellar Temperature (Temperature of the star)
Mp : Planetary Mass
Rp : Planetary Radius
P : Orbital Period of the Planet

```
for param_y in params2:
    fig, axes = plt.subplots(1, 5, figsize=(25, 5), sharey=True)
    i = 0
    for param_x in params2:
        if(param_x != param_y):
            sns.scatterplot(ax = axes[i], data=data_labeled, x=param_x, y=param_y, hue = 'tag', style='tag')
            axes[i].set_title('{} vs {}'.format(param_x, param_y))
            i = i+1
```

The plots for visualizing the relation between the six parameters is shown in the github link that we provided at the end of the report.

Conclusions from the relation graphs:

- Rocky Planets show the largest density of all groups
- sub-Neptune's' density centers around 2.84 gcm−1 which goes according to what is established in [Mousis et al., 2020].
- Hot Jupiters, apart from being the group with the lowest central planetary density, seem to orbit around the largest, more luminous and massive stars.

CONCLUSIONS:

- Two machine learning algorithms (K-means and UMAP) were used to study exoplanet parameters.
- Using orbital period and planetary mass, the K-Means algorithm leads to the classification in three different groups: Hot Jupiters, Long Period Giants and Non-Giants.
- To study the impact of other parameters, the UMAP dimensionality reduction method was used. This resulted in the classification in five different groups. In order of similarity: Hot Jupiters, Longer Period Giants, sub-Jupiters, sub-Neptunes and Rocky Planets.
- In the process of exploring the relation between the input parameters, Long Period Giants and Hot Jupiters showed an average metallicity higher than the non giant planets.

Github link to code and plots: DSA project

“ THANK YOU! ”