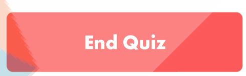
WhiteHat Jr
Live online Coding for Kids

Topic	Data-Science 2	
Class Description	**Students will be applying machine learning algorithms and statistics to filter out more planets.**	
Class	**C132**	
Class time	**45 mins**	
Goal	● Plot various charts on our data ● Filter out more planets ● Apply machine learning algorithms to real world data for multiple use cases	
Resources Required	● Teacher Resources ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ● Student Resources ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen	
Class structure	**Warm Up** **Teacher-led Activity** **Student-led Activity** **Wrap up**	**5 mins** **15 min** **15 min** **5 min**

CONTEXT

- **Review the concepts learned in the earlier classes**

Class Steps	Teacher Action	Student Action
Step 1: **Warm Up** **(5 mins)**	Hi <Student Name>! In the last class, we learned about gravity, found out the gravity for all the exo-planets and filtered out data based on that!	**ESR:** It is the mass of the planet

	Can you tell me what is the formula of Gravity that we applied in last class?	divided by the square of the radius of the planet. We then multiply this with the gravitational constant
	I have an exciting quiz question for you! Are you ready to answer this question? Teacher click on the  button on the bottom right corner of your screen to start the In-Class Quiz. A quiz will be visible to both you and the student. Encourage the student to answer the quiz question. The student may choose the wrong option, help the student to think correctly about the question and then answer again. After the student selects the correct option, the  button will start appearing on your screen. Click the End quiz to close the quiz pop-up and continue the class.	
	Excellent! Now in today's class, we will try to filter out some more planets by applying machine learning algorithms and statistics, and also we will try to find if there are any	**ESR:** "Yes!"

	interesting relations between the data points. Are you excited?	
	Let's start!	
Teacher Initiates Screen Share		
<u>CHALLENGE</u> ● **Plotting charts and applying statistics** ● **Making student understand the different planet types** ● **Applying Machine Learning Algorithm (Clustering) on the data**		
Step 2: Teacher-led Activity (15 min)	*(Before beginning the class, make sure to use the same colab that you used in the last class. This is the continuation of that.)*	
	Let's revisit what we did in the last class. We found out the gravity of all the planets and then based on the fact that our body can withstand 90 times more gravity than what Earth offers, we found out a list of 3,951 planets that have a gravity of 10 times or lower than what we have on Earth, just to be comfortable.	**ESR:** varied

```
[10] low_gravity_planets = []
     for index, gravity in enumerate(planet_gravity):
       if gravity < 10:
         low_gravity_planets.append(planet_data_rows[index])

     print(len(low_gravity_planets))

     1012

[11] low_gravity_planets = []
     for index, gravity in enumerate(planet_gravity):
       if gravity < 100:
         low_gravity_planets.append(planet_data_rows[index])

     print(len(low_gravity_planets))

     3951
```

Now, if we look at our headers, we can see that we have a header as planet_type.

```
[12] print(headers)

     ['row_num', 'name', 'light_years_from_earth', 'planet_mass', 'stellar_magnitude', 'discovery_date', 'planet_type',
```

Let’s try to find out different values of planet_type.

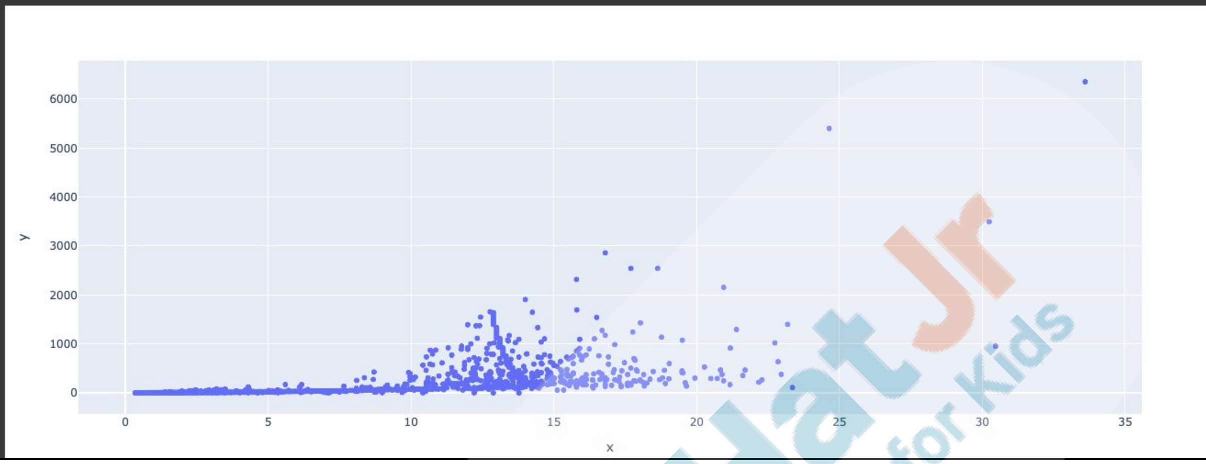
```
planet_type_values = []
for planet_data in planet_data_rows:
  planet_type_values.append(planet_data[6])

print(list(set(planet_type_values)))
```

Here, we are creating an empty list to store the values of all the planet_types and we are iterating over our planet_data_rows to append each planet’s planet_type (7th element of the list) in the empty list.

	Finally, we are printing the result. We want to print the **planet_type_values** but it contains all the planet types and there will be a lot of duplicates. To get all the unique values from the list, we have the **set()** function. Finally, we are converting the result again into a list with the **list()** function.	
	Now that we have the types of planets that are out there, let's understand these terms: **Neptune-like => These planets are like neptune! They are big in size and they are also made of Ice.** **Super-Earth => These are the planets that have mass greater than earth but smaller than that of Neptune! (Neptune is 17 times Earth)** **Terrestrial => It is a planet that is composed primarily of silicate rocks or metals. (Like Earth, Mars)** **Gas Giant => These are the planets that are composed of Gas. (Hydrogen and Helium)** Based on this, let's try to do some clustering to see if there is any relation between planet type and mass of the planet. It looks like there is but let's see.	

	Let's code a machine learning model and plot it accordingly. Let's first plot all the planets with the planet_mass and planet_radius. `planet_masses = []` `planet_radiuses = []` `for planet_data in planet_data_rows:` `planet_masses.append(planet_data[3])` `planet_radiuses.append(planet_data[7])` `fig = px.scatter(x=planet_radiuses, y=planet_masses)` `fig.show()` Here, we are first collecting all the planet mass and radius and storing them in a list, then we are plotting a scatter plot just like how we did earlier, but this time by only providing the X and the Y coordinates.	

```
planet_masses = []
planet_radiuses = []
for planet_data in low_gravity_planets:
  planet_masses.append(planet_data[3])
  planet_radiuses.append(planet_data[7])

fig = px.scatter(x=planet_radiuses, y=planet_masses)
fig.show()
```

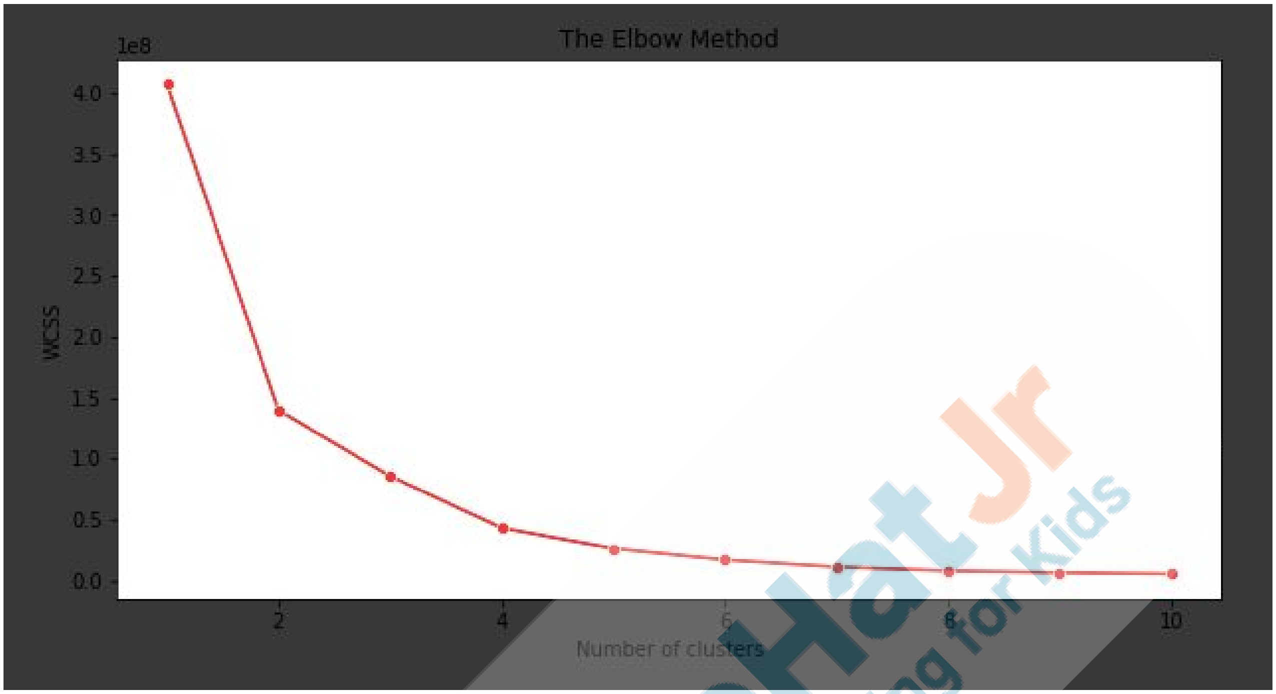


	With the following result, we don't know if there exists any clusters in this, and if yes, then how many? Let's find out the K for this cluster. Do you remember what K is in clustering and how do we find it out?	**ESR:** K signifies the number of clusters that algorithm finds in the dataset. To choose the right K, we use the WCSS perimeter to evaluate the choice of K. It stands for **Within Cluster Sum of Squares**. This means that we will be choosing the center point of the cluster from where all the points falling inside that cluster will be closest.
	Great!	
Teacher Stops Screen Share		
	Now it's your turn. Please share your	

	screen with me.	
	Ask Student to press ESC key to come back to panel **Guide Student to start Screen Share** **Teacher gets into Fullscreen**	
	ACTIVITY **Student code to build the clustering model** **Student plots graph** **Student filters out more data**	
Step 3: Student-Led Activity (15 min)	Okay, let's start by building a model for clustering. For this, we will have to merge our two lists into 1 as a list of lists. Then, we will find out the WCSS for our clusters. `from sklearn.cluster import KMeans` `import matplotlib.pyplot as plt` `import seaborn as sns` `X = []` `for index, planet_mass in enumerate(planet_masses):` `  temp_list = [` `    planet_radiuses[index],` `    planet_mass` `  ]` `  X.append(temp_list)` `wcss = []` `for i in range(1, 11):` `    kmeans = KMeans(n_clusters=i, init='k-means++', random_state = 42)`	*Student merges the two list **planet_masses and planet_radiuses** into a list of lists and then finds out the WCSS.*

	`` kmeans.fit(X) # inertia method returns wcss for that model wcss.append(kmeans.inertia_) plt.figure(figsize=(10,5)) sns.lineplot(range(1, 11), wcss, marker='o', color='red') plt.title('The Elbow Method') plt.xlabel('Number of clusters') plt.ylabel('WCSS') plt.show() ``` This is the code we have written earlier in the clustering class, but let's again go through it line by line: First, we are importing the KMeans from sklearn library, we are importing matplotlib and sns to prettify our plots. Then, we create an empty list **X** which will contain multiple lists for all the planets, having their mass and radius. We are doing the same in the first **for loop** where we are using the **enumerate()** function to get the index as well. Then, we are creating another empty list **wcss** which will contain all the WCSS values. We are then iterating 10 times (Since we expect our number of clusters to lie anywhere between 1 to 10).	

	We are preparing a **KMeans Model** where we are giving it the number of clusters, asking it to use the **k-means++** method and giving a random state of 42. **k-means++** is just the name of the algorithm that we want to use. It is the most famously used algorithm. We are then fitting our **X** in our model then appending the wcss of this into the wcss list. (inertia_ returns the value of wcss for a kmeans model) Finally, we are plotting the WCSS chart. We are giving the figure size to be 10 units wide and 5 units long. Do you remember what we call this method?	**ESR:** Elbow Method



Awesome! Here, we can see a descent up until 4, after which it is not significant.

When we found the number of unique planet types earlier, that came out to be 4 as well but when we looked at the scatter plot, we couldn't detect the number of clear clusters.

That's the power of machine learning!

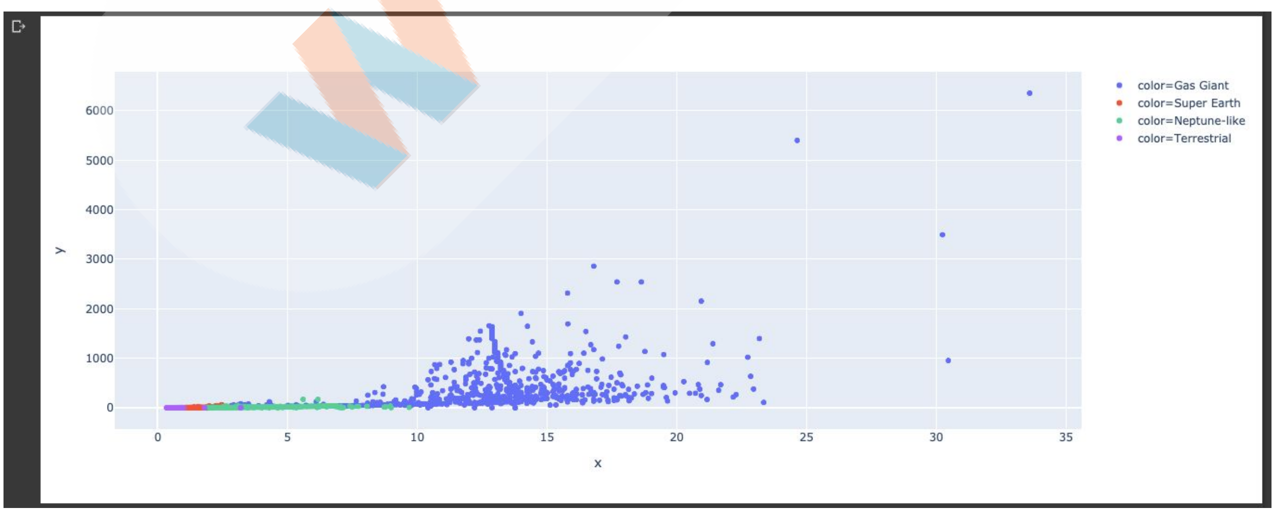
Let's plot the previous chart, this time with the color coding based on planet type.

```
planet_masses = []
planet_radiuses = []
planet_types = []
```

The student writes code to create the chart.

```
for planet_data in
low_gravity_planets:

planet_masses.append(planet_data
[3])

planet_radiuses.append(planet_da
ta[7])

planet_types.append(planet_data[
6])

fig =
px.scatter(x=planet_radiuses,
y=planet_masses,
color=planet_types)
fig.show()
```

Here, we have segregated all the planet masses, planet radiuses and the planet types. Then when we plot it, we have added an extra **color=planet_types**.



	Okay, and now the big question. Out of the 4 types of planet that we studied, which are the ones that can support life?	**ESR:** ~ Terrestrial ~ Super Earth
	Okay, then from our **low_gravity_planets** list, let's filter out planets with planet type as **Neptune-like & Gas Giant**. `suitable_planets = []` `for planet_data in low_gravity_planets:` `  if planet_data[6].lower() == "terrestrial" or planet_data[6].lower() == "super earth":` `    suitable_planets.append(planet_data)` `print(len(suitable_planets))` Here, we are creating a new list where we are checking all the low gravity planets, if they are terrestrial or super earth like, and if they are, we are adding it to a new list.	*The student creates a new list after filtering the planets!*

```
suitable_planets = []
for planet_data in low_gravity_planets:
  if planet_data[6].lower() == "terrestrial" or planet_data[6].lower() == "super earth":
    suitable_planets.append(planet_data)

print(len(suitable_planets))
```

```
1452
```

	Great! We are still left with 1,452 planets that still support us! But, were these the only factors that we needed in order to survive? What else do you think we need to survive on a planet that we need from nature?	**ESR:** Varied
Teacher Guides Student to Stop Screen Share		
FEEDBACK ● **Appreciate the student for their efforts** ● **Identify 2 strengths and 1 area of progress for the student**		
Step 4: **Wrap-Up** **(5 min)**	So, in this class, we learned about different types of planets and also, we witnessed how machine learning can help us identify things that we otherwise cannot do by ourselves! How was your experience?	**ESR:** varied
	Amazing. While working on this project, we also made sure that we are on top of all the concepts we have acquired so far. Next class, we will filter out more planets based on various other factors. We will also learn new concepts and techniques!	
Teacher Clicks ✖ End Class		

Activity	Activity Name	Links

Teacher Activity 1	Solution	https://colab.research.google.com/drive/1fSpkokhAWD_z2A1ZnNIQvjljcVWOmV8K?usp=sharing