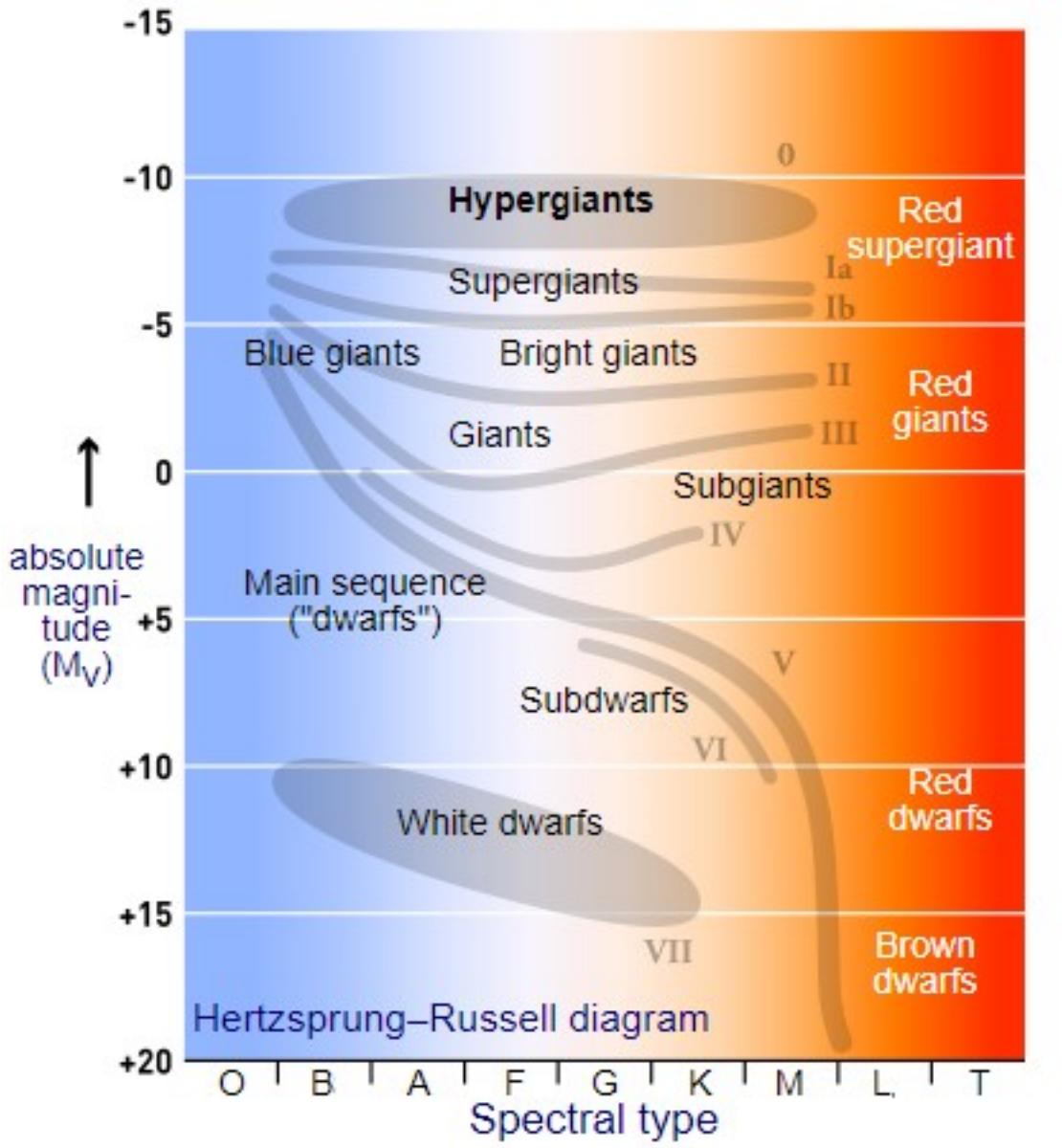
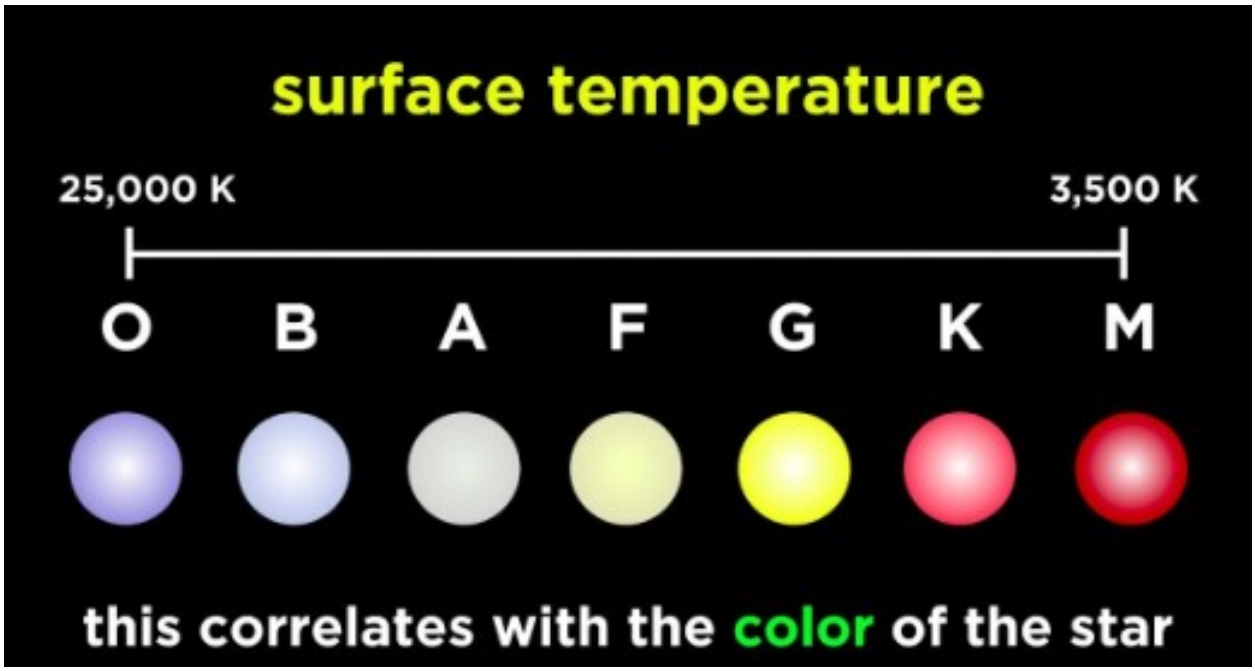
Astronomicala Data (Tabular) - Visualizations

Exploring and Understanding the Star Type Data

- CSV data link:- https://drive.google.com/uc?id=1BQVc6MHjQFtDC9iP1isT_K4ojVe_Oil-






```
# Importing the libraries
import os
import pandas as pd
import numpy as np
import seaborn as sns
# import matplotlib.pyplot as plt
from IPython.display import Image, display

# Peek into the data by creating pandas dataframe
star_df = pd.read_csv('https://drive.google.com/uc?
id=1BQVc6MHjQFtDC9iP1isT_K4ojVe_Oil-')
star_df.sample(10) # Random 10 samples from star_df
```

```
     Temperature (K)  Luminosity(L/Lo)  Radius(R/Ro)  Absolute
magnitude(Mv)  \
31             30000       28840.00000       6.30000
-4.20
2               2600           0.00030       0.10200
18.70
125             3225           0.00076       0.12100
19.63
56              3660      363000.00000    1673.00000
-11.92
131             3607           0.00023       0.38000
10.34
140            13420           0.00059       0.00981
13.67
90              5300           0.59000       0.91000
5.49
53              3749      550000.00000    1648.00000
```

```
-8.05
198              3324           0.00650       0.47100
12.78
109             33421      352000.00000      67.00000
-5.79
```

```
     Star type    Star color Spectral Class
31           3    Blue-White              B
2            0           Red              M
125          0           Red              M
56           5           Red              M
131          1           Red              M
140          2    Blue-White              B
90           3  Yellow-White              F
53           5           Red              M
198          1           Red              M
109          4          Blue              O
```

```
# Check general information about the dataframe
star_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 240 entries, 0 to 239
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Temperature (K)         240 non-null    int64  
 1   Luminosity(L/Lo)        240 non-null    float64
 2   Radius(R/Ro)            240 non-null    float64
 3   Absolute magnitude(Mv)  240 non-null    float64
 4   Star type               240 non-null    int64  
 5   Star color              240 non-null    object 
 6   Spectral Class          240 non-null    object 
dtypes: float64(3), int64(2), object(2)
memory usage: 13.2+ KB
```

Observations

1) Dataset consists of 240 rows, 6 feature and 1 target columns,

- *Absolute Temperature (in K)*

- *Relative Luminosity (L/Lo)*
- *Relative Radius (R/Ro)*
- *Absolute Magnitude (Mv)*
- *Star Color*
- *Spectral Class*
- *Star Type (Target classes)*

Here,

- *Lo = 3.828 x 10^26 Watts (Avg Luminosity of Sun)*
- *Ro = 6.9551 x 10^8 m (Avg Radius of Sun)*

2) Two categorical features (object type) which will need some kind of encoding,

- *Star Color*
- *Spectral Class*

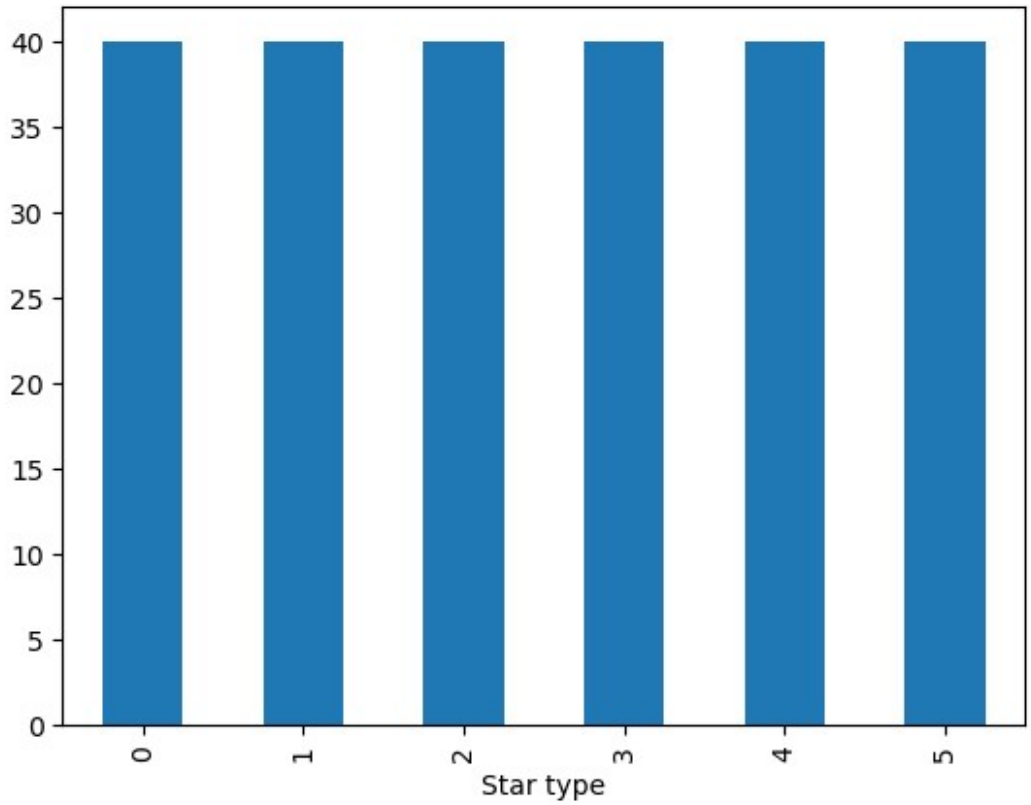
3) It consists of some different features of stars. Information on the star type is given below,

- **0** → Brown Dwarf
- **1** → Red Dwarf
- **2** → White Dwarf
- **3** → Main Sequence
- **4** → Supergiants
- **5** → Hypergiants

```
# Create a directory to save the visualizations
folder_name = 'star_plots'                      # directory name
os.makedirs(folder_name, exist_ok=True)         # Create directory if
not exisiting
base_dir = f'{folder_name}/'           # Store the path of this
directory as base_dir to use it further

star_types_count = star_df['Star type'].value_counts()
print(star_types_count)
```

```
Star type
0    40
1    40
2    40
3    40
4    40
5    40
Name: count, dtype: int64
```

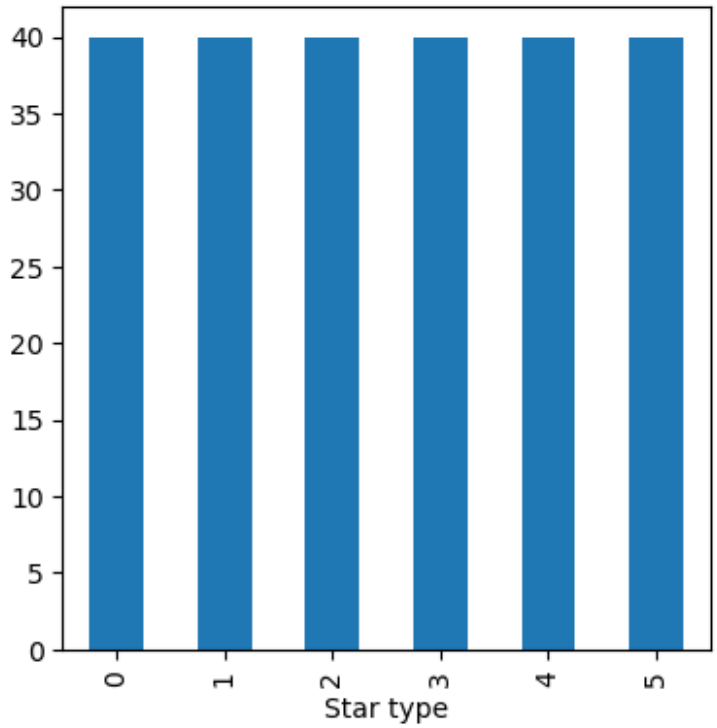
```
import matplotlib.pyplot as plt

# Bar chart to visualize the count of stars with respective type

plt.style.use('default')

star_df['Star type'].value_counts().plot(kind='bar')
plt.show()
```



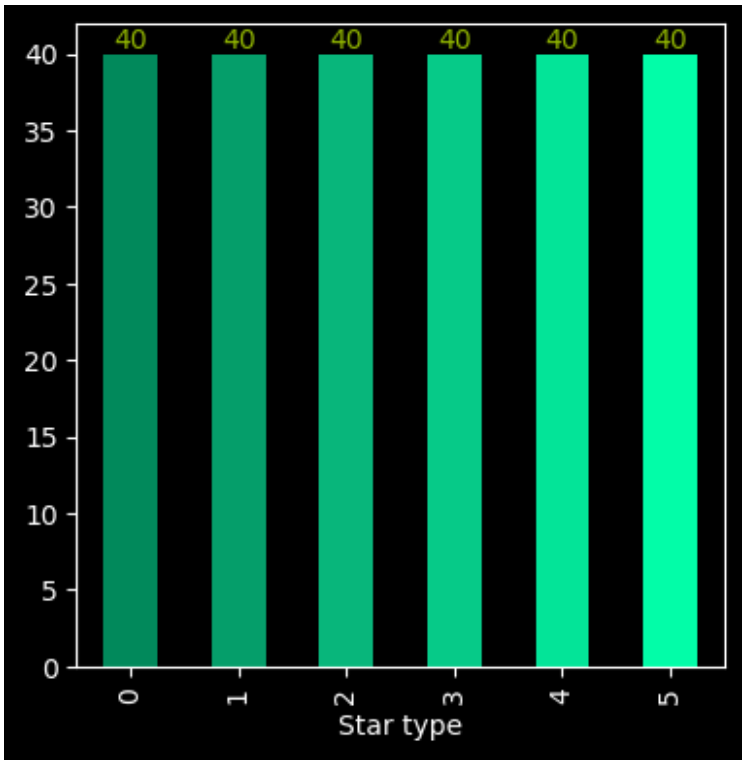

```
# Calculate the count of each star type
star_counts = star_df['Star type'].value_counts().reset_index()
star_counts.columns = ['Star type', 'Count']  # Rename columns for
clarity

print(star_counts)
print(star_counts.columns)
```

```
   Star type  Count
0          0     40
1          1     40
2          2     40
3          3     40
4          4     40
5          5     40
Index(['Star type', 'Count'], dtype='object')
```

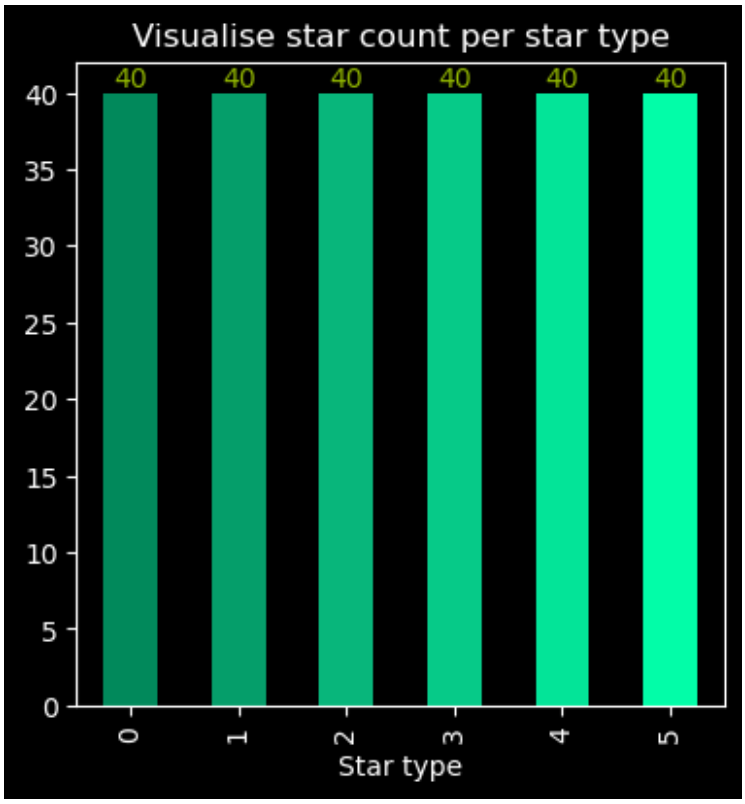
```
import matplotlib.pyplot as plt

plt.figure(figsize=(4.3, 4.3))

plt.style.use('default')

star_df['Star type'].value_counts().plot(kind='bar')
plt.show()
```



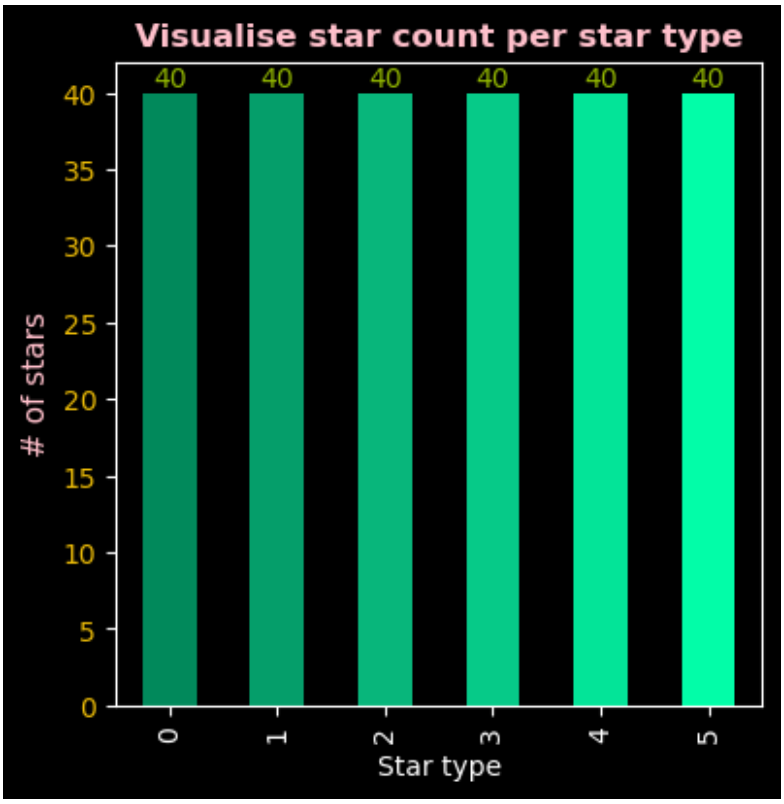

```
import matplotlib.pyplot as plt

plt.figure(figsize=(4.3, 4.3))

plt.style.use('dark_background')

ax = star_df['Star type'].value_counts().plot(kind='bar',
color=['#01895b', '#059e6a', '#08b67b', '#06ca88', '#03e498',
'#02fda8'])
ax.bar_label(ax.containers[0], color='#92b600')
plt.show()
```




```
import matplotlib.pyplot as plt

plt.figure(figsize=(4.3, 4.3))

plt.style.use('dark_background')

ax = star_df['Star type'].value_counts().plot(kind='bar',
color=['#01895b', '#059e6a', '#08b67b', '#06ca88', '#03e498',
'#02fda8'])
ax.bar_label(ax.containers[0], color='#92b600')
plt.title('Visualise star count per star type', color='white')
plt.show()
```

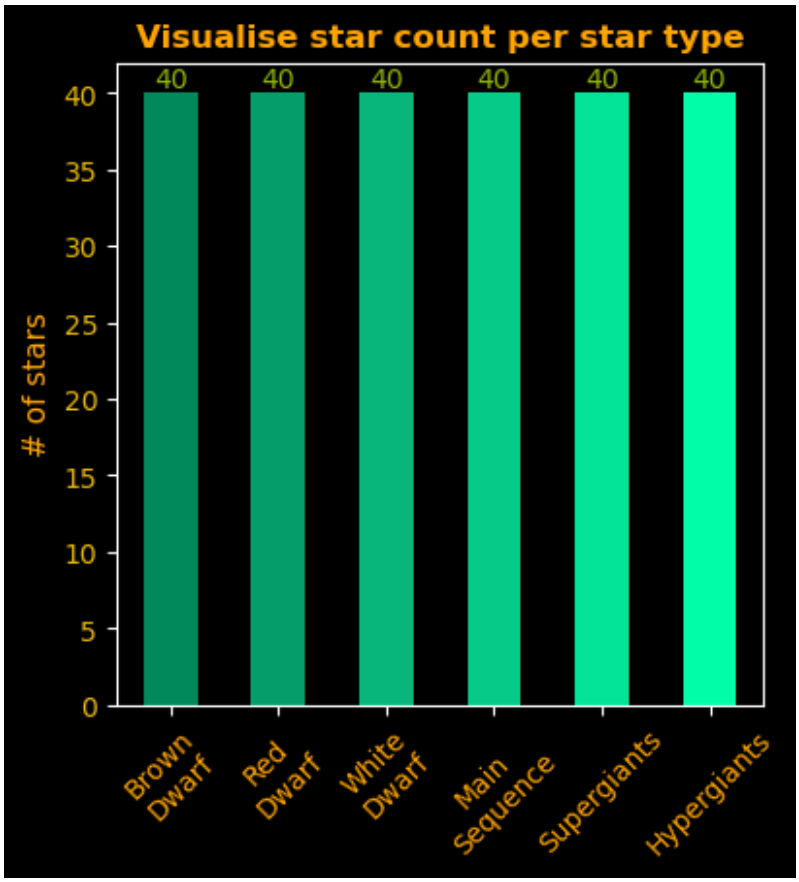



```
import matplotlib.pyplot as plt

plt.figure(figsize=(4.3, 4.3))

plt.style.use('dark_background')

ax = star_df['Star type'].value_counts().plot(kind='bar',
color=['#01895b', '#059e6a', '#08b67b', '#06ca88', '#03e498',
'#02fda8'])
ax.bar_label(ax.containers[0], color='#92b600')
plt.title('Visualise star count per star type', color='pink',
weight='bold')
plt.yticks(color='#eaba08')
plt.ylabel('# of stars', color='pink', fontsize=11)
plt.show()
```




```
import matplotlib.pyplot as plt

plt.figure(figsize=(4.3, 4.3))

plt.style.use('dark_background')

ax = star_df['Star type'].value_counts().plot(kind='bar',
color=['#01895b', '#059e6a', '#08b67b', '#06ca88', '#03e498',
'#02fda8'])
ax.bar_label(ax.containers[0], color='#92b600')
plt.title('Visualise star count per star type', color='orange',
weight='bold')
plt.yticks(color='#eaba08')
plt.xlabel('')
plt.ylabel('# of stars', color='orange', fontsize=11)
plt.xticks(
    ticks=[0, 1, 2, 3, 4, 5],
    labels=[
        'Brown\nDwarf',
        'Red\nDwarf',
        'White\nDwarf',
        'Main\nSequence',
        'Supergiants',
        'Hypergiants'
    ],
    rotation=45,
```

```
    color='orange'
)

plt.savefig(base_dir + 'barplot_star_count.png')
plt.show()
```



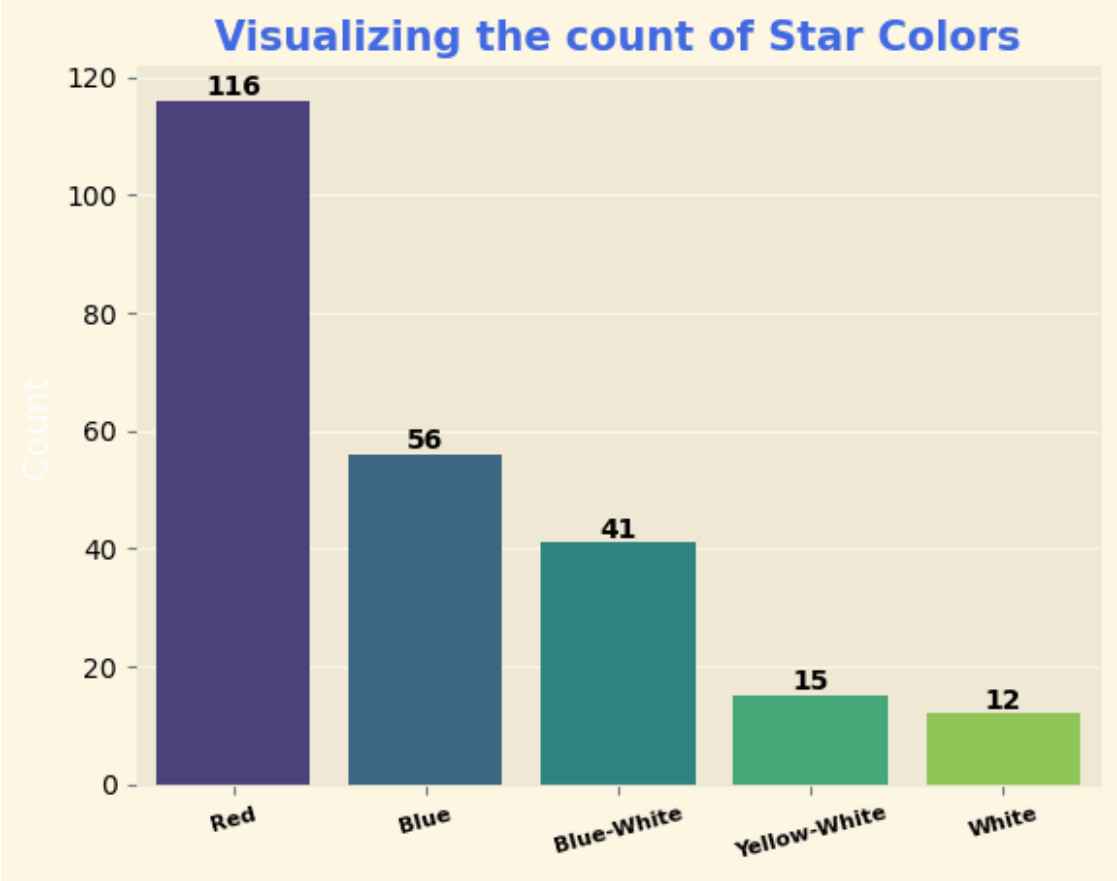
Final Presentation after edit

Use seaborns barplot to compare it with the above bar chart

(Matplotlib + Seaborn)

```
import matplotlib.pyplot as plt1

# Visualizing the Star color data
star_counts = star_df['Star color'].value_counts()
ax = sns.barplot(x=star_counts.index,
                 y=star_counts.values,
                 hue=star_counts.index,
                 palette='viridis',
```

```
                  legend=False)

plt1.style.use('Solarize_Light2')

# Use ax.containers to get all bar containers
for container in ax.containers:
    ax.bar_label(container, color='black', weight='bold')

plt1.title('Visualizing the count of Star Colors', color='royalblue',
fontsize=15, weight='bold')
plt1.xticks(rotation=15, color='black', fontsize=8, weight='bold')
plt1.xlabel('')
plt1.ylabel('Count', color='white', fontsize=13)
plt1.yticks(color='black')
plt1.savefig(base_dir + 'star_colors_viz.png')
plt1.show()
```




```
plt.style.available

['Solarize_Light2',
 '_classic_test_patch',
 '_mpl-gallery',
```

```
 '_mpl-gallery-nogrid',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn-v0_8',
 'seaborn-v0_8-bright',
 'seaborn-v0_8-colorblind',
 'seaborn-v0_8-dark',
 'seaborn-v0_8-dark-palette',
 'seaborn-v0_8-darkgrid',
 'seaborn-v0_8-deep',
 'seaborn-v0_8-muted',
 'seaborn-v0_8-notebook',
 'seaborn-v0_8-paper',
 'seaborn-v0_8-pastel',
 'seaborn-v0_8-poster',
 'seaborn-v0_8-talk',
 'seaborn-v0_8-ticks',
 'seaborn-v0_8-white',
 'seaborn-v0_8-whitegrid',
 'tableau-colorblind10']
```

Visualize outliers if any by creating boxplots for numeric features

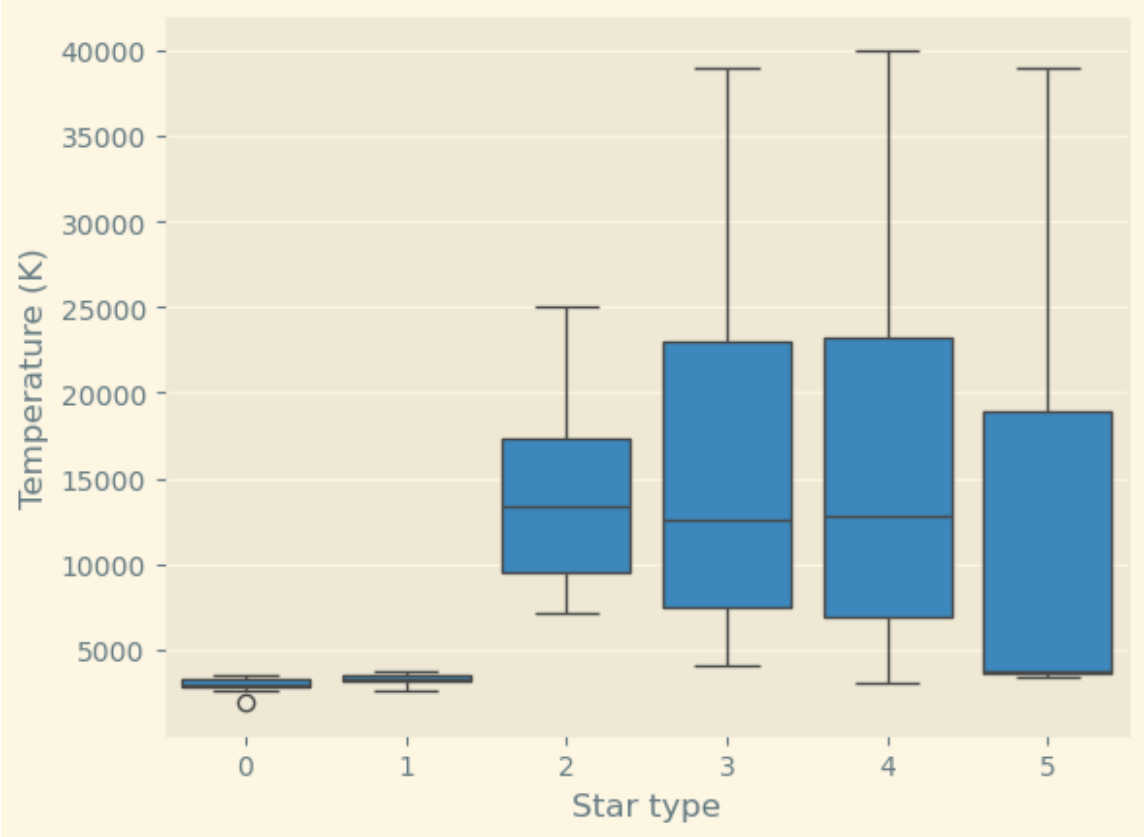
- We will be creating subplot for all the numeric features.
- When we create multiple plots inside one figure that is what we refer to as the subplot.
- It needs to know how many rows and columns to use to create different plots.
- `plt.subplot(rows, cols, position)` here position indicates out of all the rows & columns, which position to plot in.
- For example, `plt.subplot(2, 3, 2)` means that create a plot on the second postion of 6 available positions.

```
# Get a gist of the data again by checking the top 5 rows of the data
star_df.head(5)
```

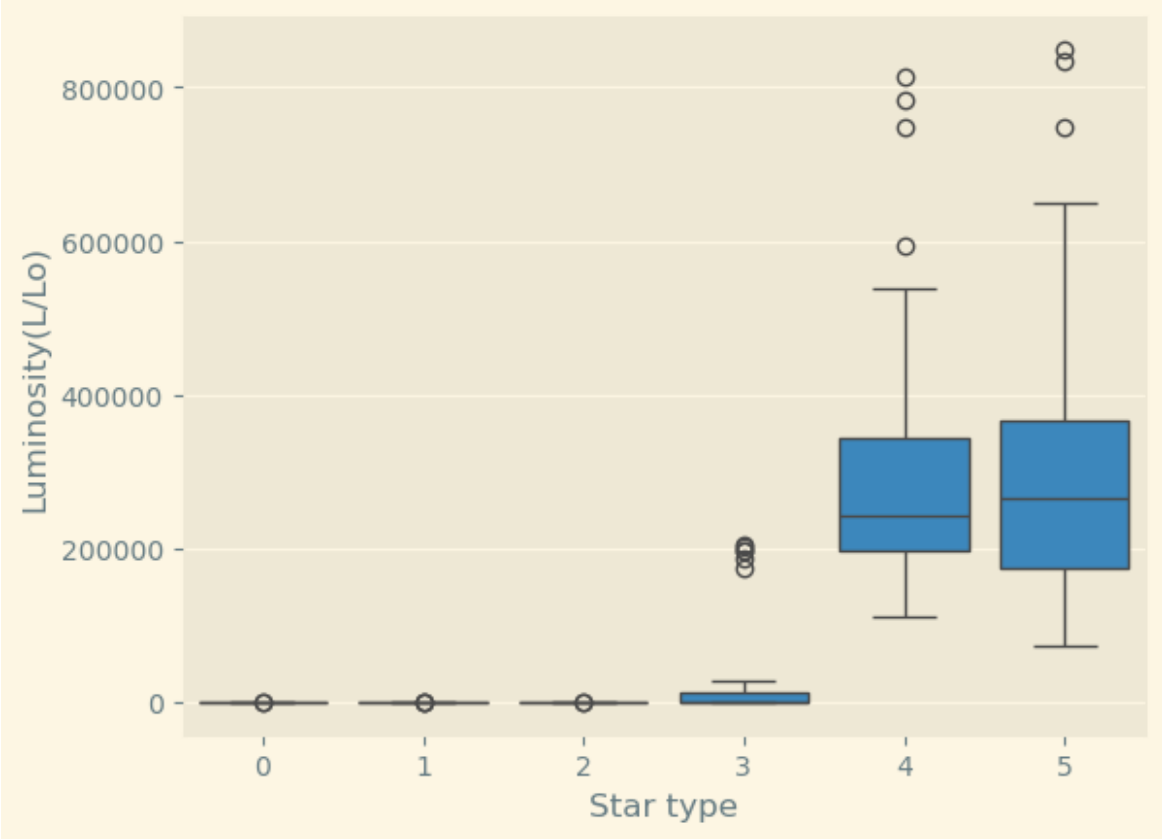
```
   Temperature (K)  Luminosity(L/Lo)  Radius(R/Ro)  Absolute
magnitude(Mv)  \
0             3068          0.002400        0.1700
16.12
1             3042          0.000500        0.1542
16.60
2             2600          0.000300        0.1020
18.70
```

```
3              2800           0.000200         0.1600
16.65
4              1939           0.000138         0.1030
20.06

   Star type Star color Spectral Class
0          0        Red               M
1          0        Red               M
2          0        Red               M
3          0        Red               M
4          0        Red               M
```

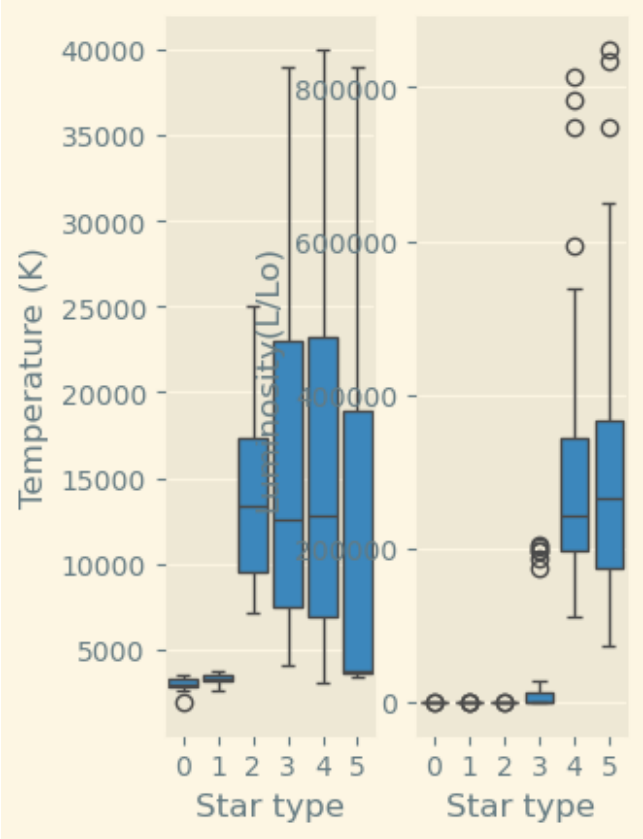
```
import matplotlib.pyplot as plt2

sns.boxplot(x=star_df['Star type'], y=star_df.iloc[:,0])
plt2.show()
```



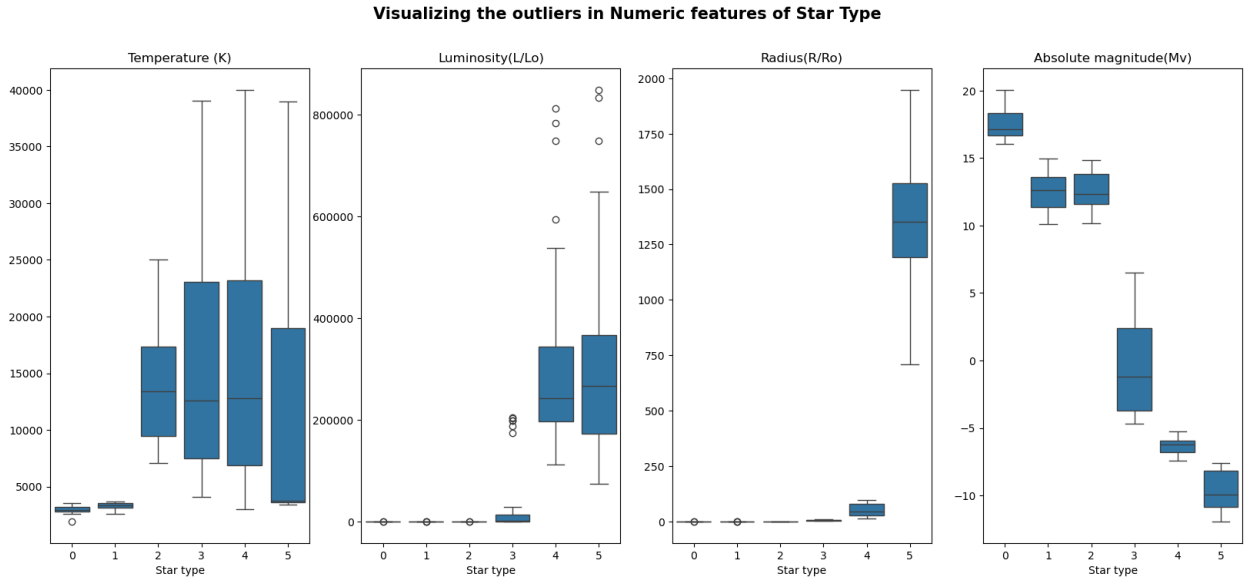

```
import matplotlib.pyplot as plt3

sns.boxplot(x=star_df['Star type'], y=star_df.iloc[:,1]);
plt3.show()
```



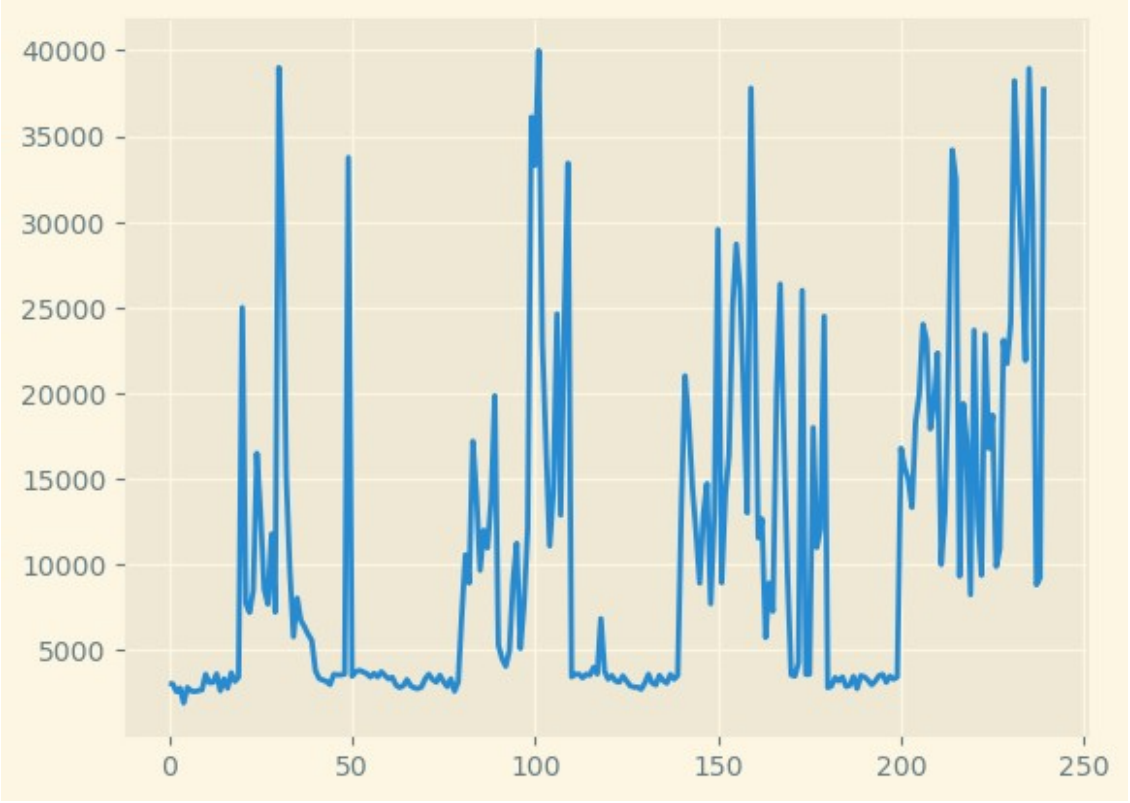

```
import matplotlib.pyplot as plt5

plt5.subplot(1, 4, 1);
sns.boxplot(x=star_df['Star type'], y=star_df.iloc[:,0])

plt5.subplot(1, 4, 2);
sns.boxplot(x=star_df['Star type'], y=star_df.iloc[:,1])

plt.show()
```




```
import matplotlib.pyplot as plt6

plt6.style.use('default')
plt6.figure(figsize=(20, 8))
plt6.suptitle('Visualizing the outliers in Numeric features of Star
Type',
              color='black', weight='bold', fontsize=15)

for i in range(4):
    plt6.subplot(1, 4, i+1)
    sns.boxplot(x=star_df['Star type'], y=star_df.iloc[:,i]);
    plt6.title(star_df.columns[i], color='black')
    plt6.ylabel('')

plt6.savefig(base_dir + 'boxplot_star_type.png');
```

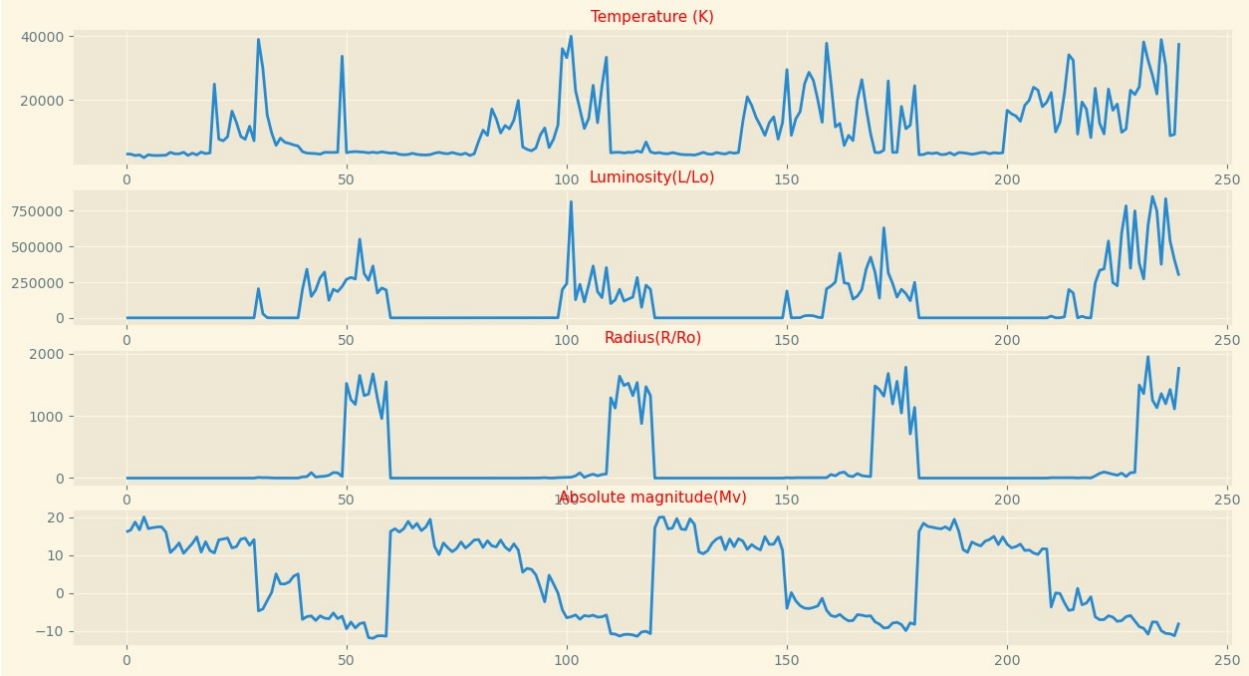



```
# Line Plots

plt.plot(star_df.iloc[:,0])
plt.show()
```

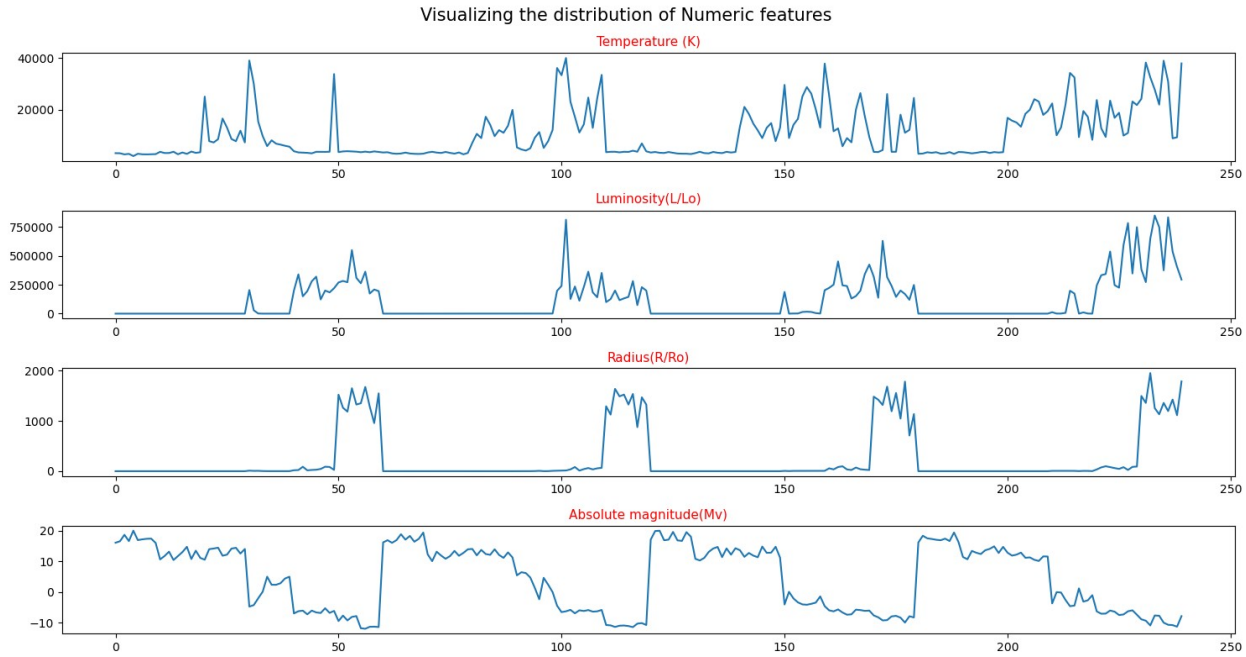


```
import matplotlib.pyplot as plt7
```

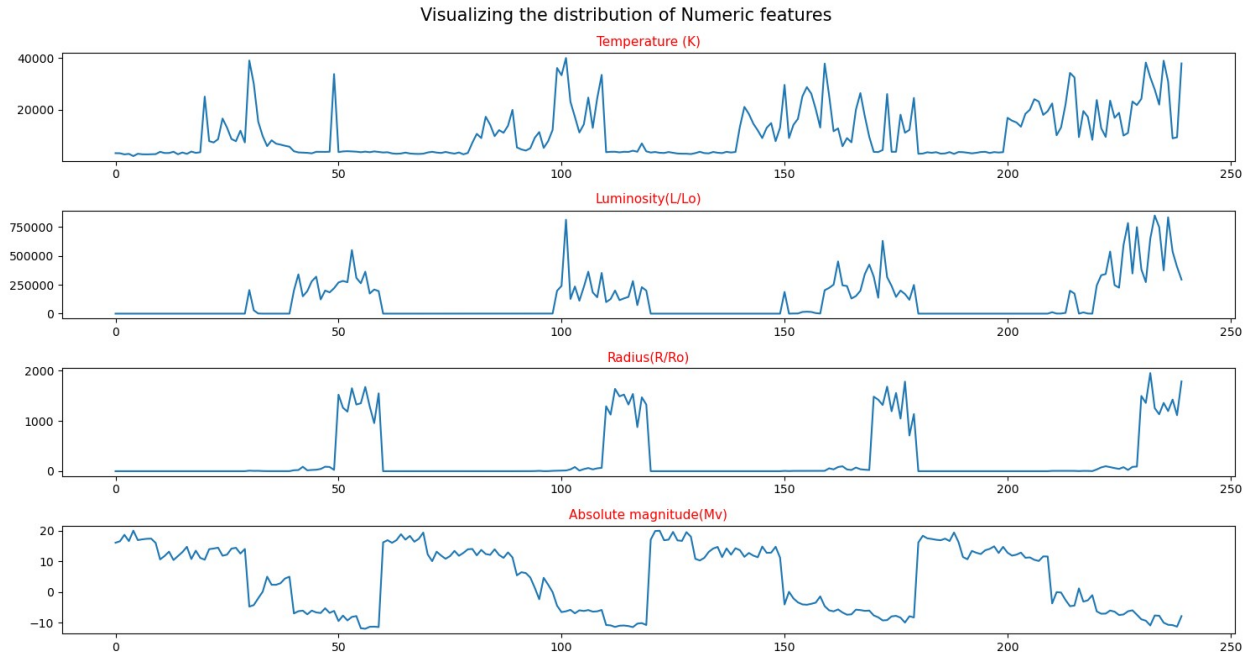
```
plt7.figure(figsize=(15, 8))

for i in range(4):
    plt7.subplot(4, 1, i+1)
    plt7.plot(star_df.iloc[:,i])
    plt7.title(star_df.columns[i], color='red', fontsize=11)

plt7.show()
```




```
import matplotlib.pyplot as plt8

plt8.style.use('default')

# color = ['royalblue', 'gold', 'lime', 'magenta']
plt8.figure(figsize=(15, 8))
plt8.suptitle('Visualizing the distribution of Numeric features',
fontsize=15, color='black')

for i in range(4):
    plt8.subplot(4, 1, i+1)
    plt8.plot(star_df.iloc[:,i])
    # plt8.plot(star_df.iloc[:,i], color=color[i])
    plt8.title(star_df.columns[i], color='red', fontsize=11)

plt8.tight_layout()
plt8.savefig(base_dir + 'distribution_numeric_features.png')
plt8.show()
```



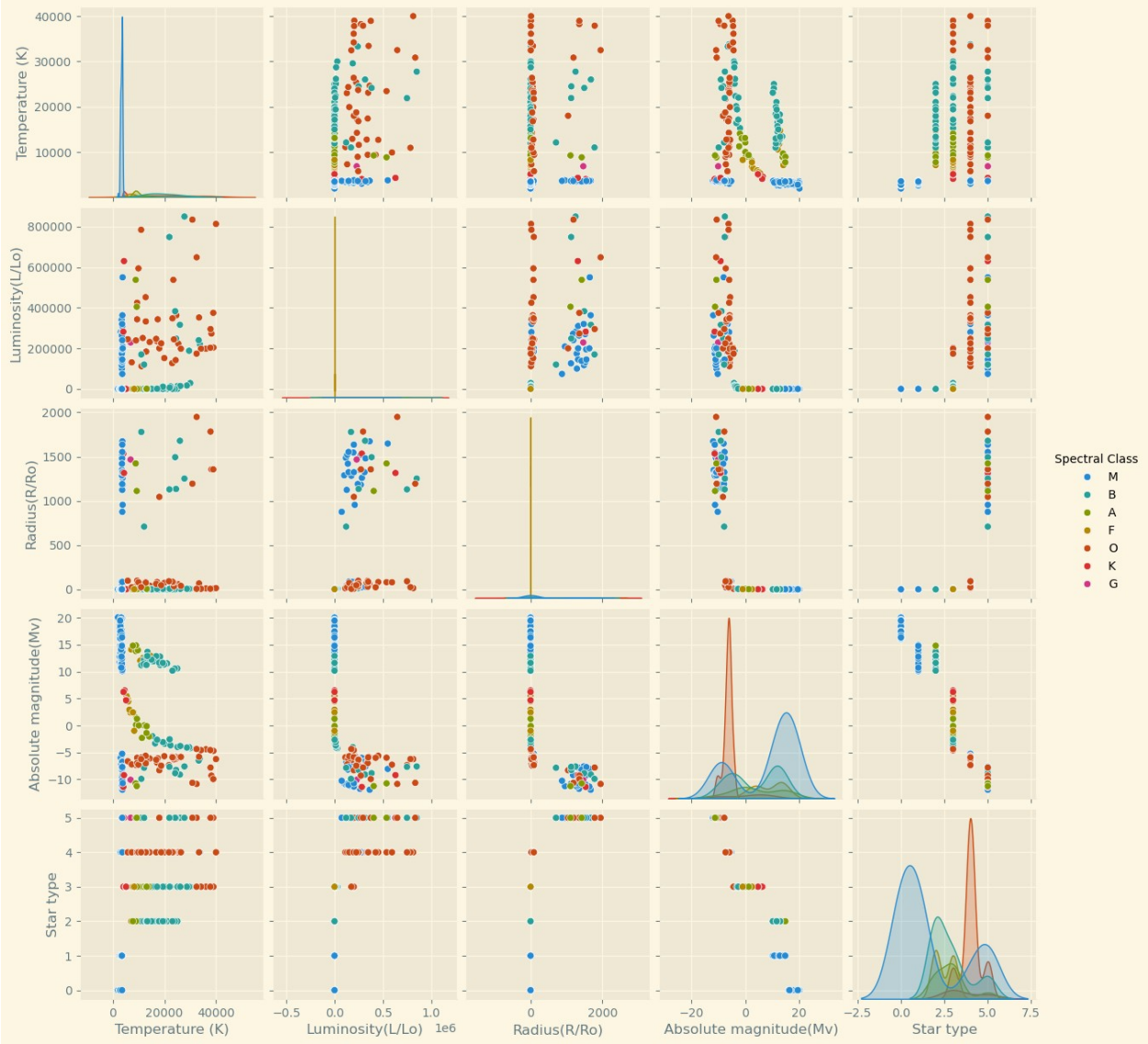

```
def line_subplot(star_df, plt, i):
    plt.subplot(4, 1, i+1)
    plt.plot(star_df.iloc[:,i])
    # plt.plot(star_df.iloc[:,i], color=color[i])
    plt.title(star_df.columns[i], color='red', fontsize=11)


import matplotlib.pyplot as plt9

plt9.style.use('default')

# color = ['royalblue', 'gold', 'lime', 'magenta']
plt9.figure(figsize=(15, 8))
plt8.suptitle('Visualizing the distribution of Numeric features',
fontsize=15, color='black')

for i in range(4):
    line_subplot(star_df, plt9, i)

plt8.tight_layout()
plt8.show()
```



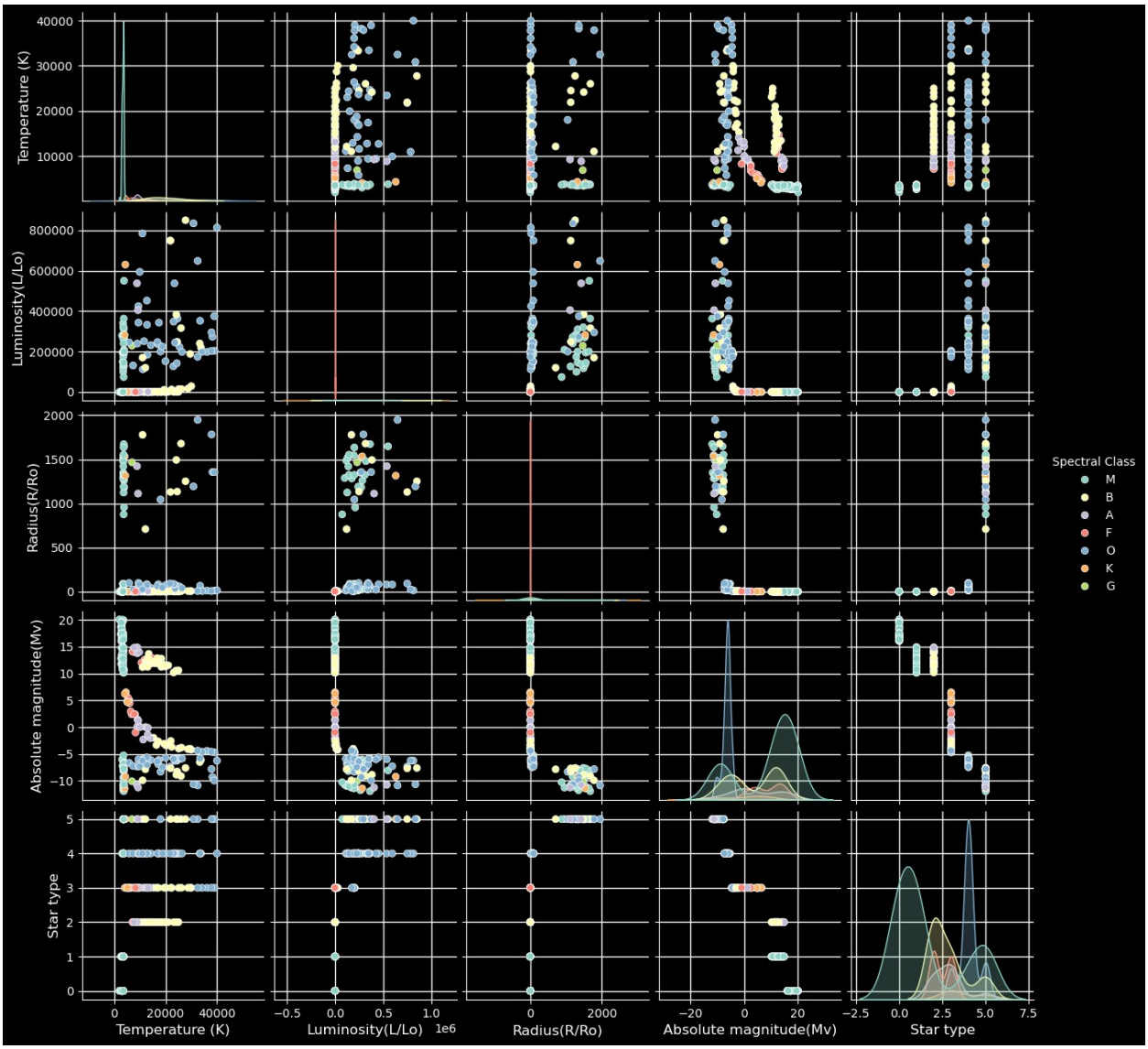
Create a pairplot

- It will give you the scatter plot by default and you can change the kind of plot you want for of all variables with each other.
- It gives all the plots together without using subplot manually.
- That's the power of seaborn! But it has a disadvantage as well.
- If you have a lot more features in your dataset then it would be too time consuming to run this + it won't be properly visible.
- However, as we have less features let's obtain scatter pairplot with hue being set to Spectral Class.
- Hue will allow us to compare two different features with respect to the spectral class.

```
import matplotlib.pyplot as plt10

# Get a pairplot - scatter

# sns.pairplot(star_df)
sns.pairplot(star_df, hue='Spectral Class')
plt10.show()
```




```
import matplotlib.pyplot as plt11

# Get a pairplot - scatter
plt11.style.use('dark_background')

# sns.pairplot(star_df)
sns.pairplot(star_df, hue='Spectral Class')
plt11.savefig(base_dir + 'pairplot.png')
plt11.show()
```



Create a scatter plot of HR diagram

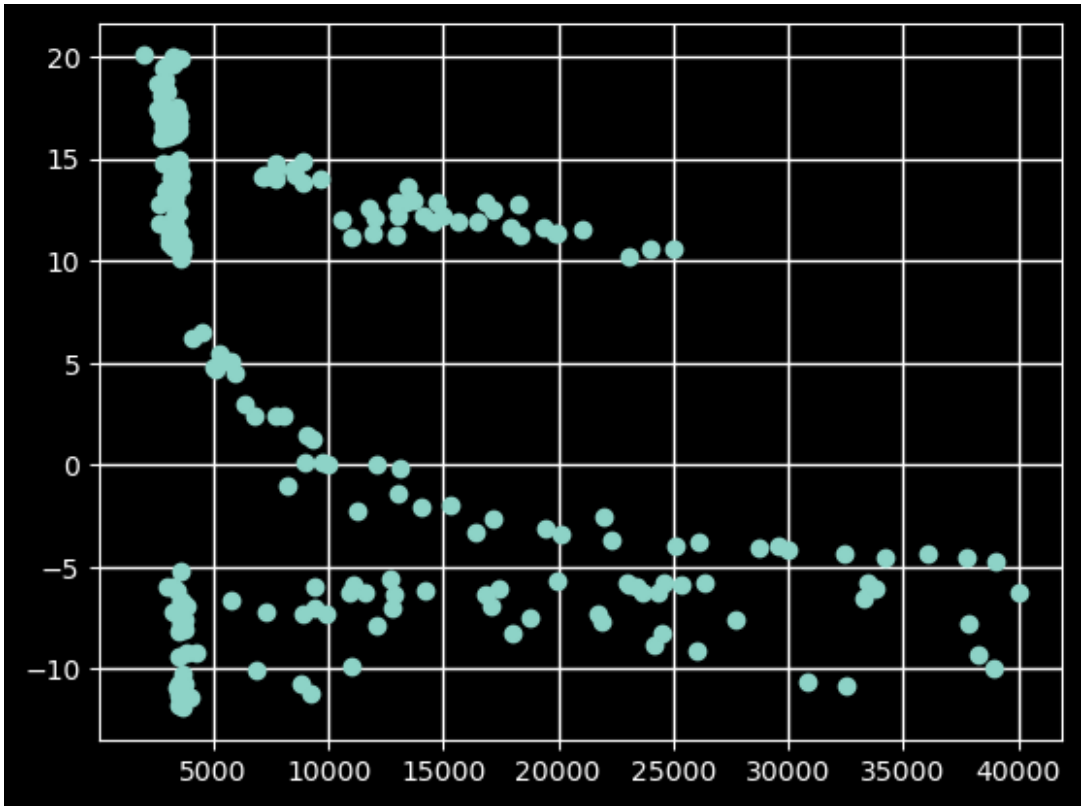
- Using the information we have in our star_df let us try to create something like this:-

0 → Brown Dwarf
1 → Red Dwarf
2 → White Dwarf
3 → Main Sequence
4 → Supergiants
5 → Hypergiants

```
temperature = star_df['Temperature (K)'].values

# print(temperature)
print(type(temperature))
```

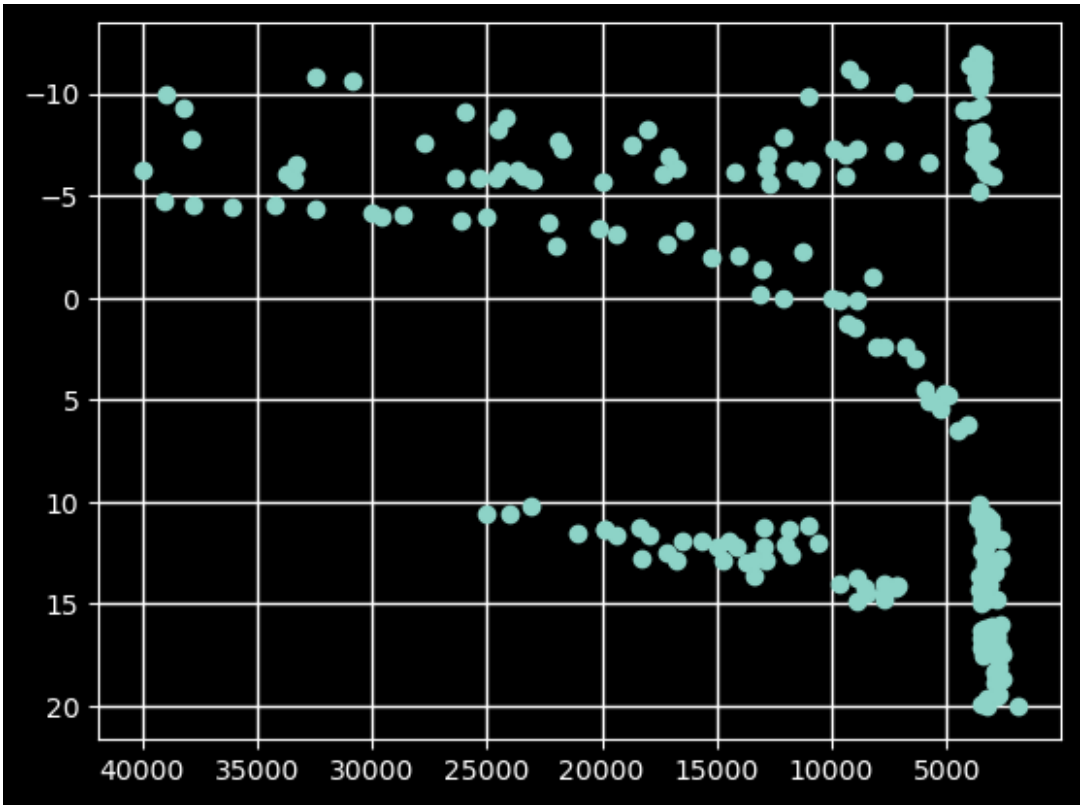
```
<class 'numpy.ndarray'>
```

```
# Plotting a HR Diagram for Temp vs Abs mag
import matplotlib.pyplot as plt12

temperature = star_df['Temperature (K)'].values
abs_mag = star_df['Absolute magnitude(Mv)'].values

plt12.scatter(temperature, abs_mag)
plt12.show()

# this is not working because in HR diagram, the luminosity decreases
with x and abs mag decreases with y
```




```
# Plotting a HR Diagram for Temp vs Abs mag correctly
import matplotlib.pyplot as plt13

temperature = star_df['Temperature (K)'].values
abs_mag = star_df['Absolute magnitude(Mv)'].values

plt13.scatter(temperature, abs_mag)

plt.gca().invert_xaxis()
plt.gca().invert_yaxis()
```

```
plt13.show()
```

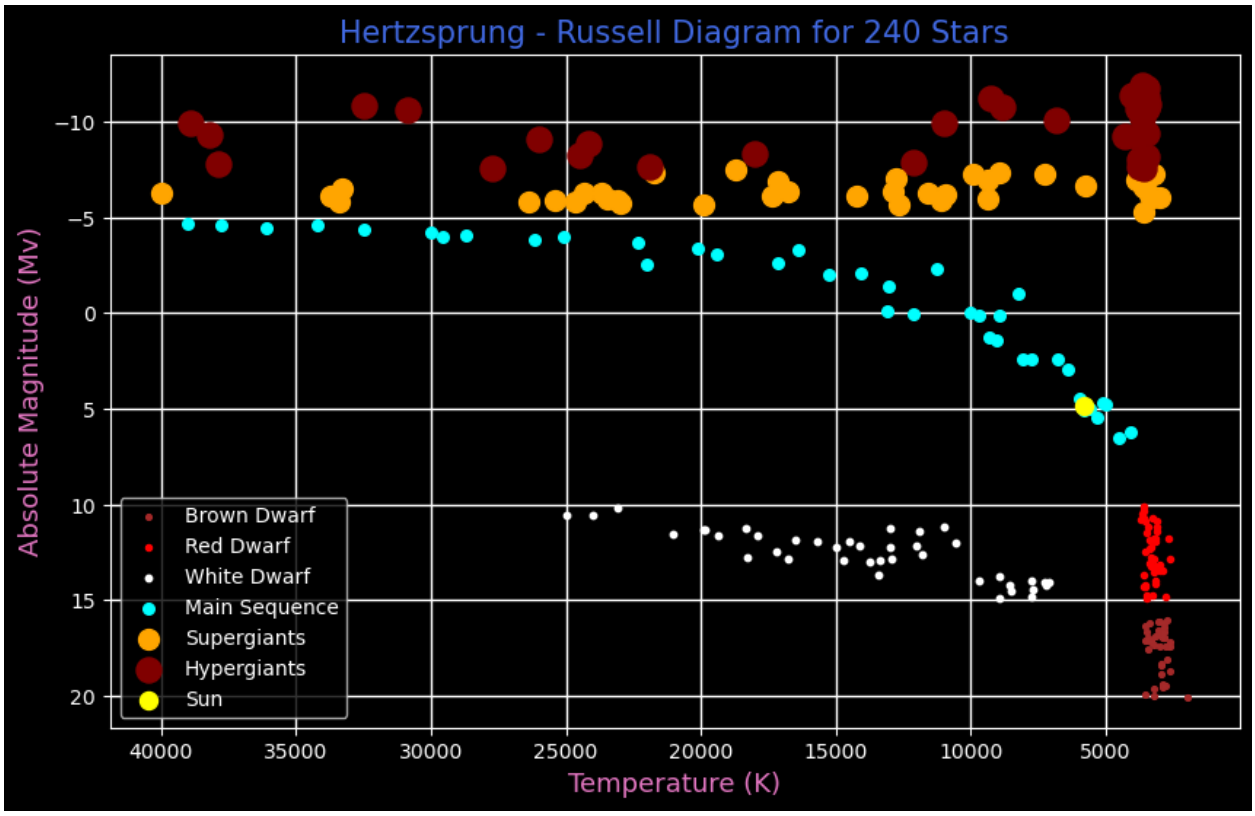



```
import matplotlib.pyplot as plt14

star_type = star_df['Star type'].values
temperature = star_df['Temperature (K)'].values
abs_mag = star_df['Absolute magnitude(Mv)'].values

star_types = {
    0: {'label': 'Brown Dwarf', 'color': 'brown', 'size': 30,
'marker': '.'},
    1: {'label': 'Red Dwarf', 'color': 'red', 'size': 35, 'marker':
'.'},
    2: {'label': 'White Dwarf', 'color': 'white', 'size': 40,
'marker': '.'},
    3: {'label': 'Main Sequence', 'color': 'cyan', 'size': 30,
'marker': 'o'},
    4: {'label': 'Supergiants', 'color': 'orange', 'size': 100,
'marker': 'o'},
    5: {'label': 'Hypergiants', 'color': 'maroon', 'size': 150,
'marker': 'o'}
}

star_types[0]
```

```
{'label': 'Brown Dwarf', 'color': 'brown', 'size': 30, 'marker': '.'}
```

```
axes = []
labels = set()

plt14.figure(figsize=(10, 6))

for i in range(len(star_type)):
    properties = star_types[star_type[i]]

    if properties['label'] not in labels:
        ax = plt.scatter(
                temperature[i],
                abs_mag[i],
                s=properties['size'],
                c=properties['color'],
                marker=properties['marker'],
                label=properties['label']
        )
        axes.append(ax)
        labels.add(properties['label'])
    else:
        plt.scatter(
                temperature[i],
                abs_mag[i],
                s=properties['size'],
                c=properties['color'],
                marker=properties['marker'],
                label=properties['label']
        )

# Adding data for sun
ax_sun = plt14.scatter(5778, 4.83, s=75, c="yellow", marker='o',
label='Sun')
axes.append(ax_sun)
labels.add("Sun")

# Add title
plt14.title(f"Hertzsprung - Russell Diagram for {len(star_type)}
Stars", fontsize=15, color='royalblue')

# Add labels
plt14.ylabel("Absolute Magnitude (Mv)", fontsize=13, color='tab:pink')
plt14.xlabel("Temperature (K)", fontsize=13, color='tab:pink')

# plt14.legend()  # this has a problem, it assigns the label for all
the star i.e., 240 legends
plt14.legend(handles=axes)
```

```
plt14.gca().invert_xaxis()
plt14.gca().invert_yaxis()

plt14.savefig(base_dir + 'hr_diagram.png')
plt14.show()
```




```
import os
import zipfile

def zipdir(path, ziph):
    # ziph is the zipfile handle
    for root, dirs, files in os.walk(path):
        for file in files:
            file_path = os.path.join(root, file)
            # Set the desired path inside the zip file
            arcname = os.path.join('content', 'star_plots',
os.path.relpath(file_path, path))
            ziph.write(file_path, arcname)

with zipfile.ZipFile('Star_Plots.zip', 'w', zipfile.ZIP_DEFLATED) as
zipf:
    zipdir('star_plots/', zipf)

print("Created zip file: Python.zip")
```

```
Created zip file: Python.zip
```