Artificial Intelligence

In Astronomy

What is AI

AI, or artificial intelligence, refers to the development of computer systems that can perform tasks that typically require human intelligence, such as learning, reasoning, and problem-solving. AI systems use algorithms and data to recognize patterns and make predictions, often improving their accuracy over time as they learn from more data. Machine learning is a subset of AI that involves training algorithms on large amounts of data to make predictions or take actions without being explicitly programmed.

Deep learning is a type of machine learning that uses neural networks, which are modeled on the structure of the human brain, to perform complex tasks like image and speech recognition. Natural language processing is another subset of AI that focuses on enabling machines to understand and process human language, both spoken and written. AI has many real-world applications, including in fields like healthcare, finance, transportation, and entertainment.

Impact of AI

in Astronomy

Image Analysis: With the vast amount of data that telescopes and satellites collect, it can be a daunting task for astronomers to sift through and analyze all of the images. AI algorithms can help automate the process of identifying and classifying objects in these images, such as galaxies, stars, and other celestial bodies. This has led to new discoveries and insights into the universe.

Data Processing: AI is also being used to process and analyze large data sets generated by telescopes and other instruments. AI algorithms can identify patterns and trends in the data that might be difficult or impossible for humans to detect. This has led to a better understanding of the universe and the processes that govern it.

Object Identification: AI can help identify and classify objects in the universe. For example, AI algorithms can analyze the light spectra of stars and galaxies to identify their chemical composition and age. This information can be used to study the formation and evolution of galaxies and other celestial bodies.

Search for Exoplanets: AI is being used to analyze data from telescopes to search for exoplanets, which are planets outside our solar system. AI algorithms can identify subtle changes in the light emitted by a star, which could indicate the presence of a planet orbiting around it. This has led to the discovery of many new exoplanets.

Simulation and Modeling: AI is also being used to simulate and model the behavior of celestial bodies, such as stars, galaxies, and black holes. This helps astronomers better understand how these objects interact with each other and their environment.

Overall, AI has revolutionized the way astronomers analyze and understand the universe. It has enabled astronomers to process and analyze vast amounts of data, identify new objects and phenomena, and simulate complex astronomical systems.

Use of AI in Astronomy

- Artificial intelligence (AI) has become an increasingly important tool in astronomy over the last few years. Here are some of the different applications of AI in astronomy:

- **Image analysis:** AI can help analyze images taken by telescopes to identify patterns and structures that might be difficult for humans to detect. For example, AI algorithms can be trained to identify gravitational lenses, which are often difficult to spot in astronomical images.

- **Data processing:** Astronomy produces vast amounts of data that need to be processed and analyzed. AI can help automate this process by identifying and categorizing data, reducing noise, and enhancing signal-to-noise ratios.

- **Exoplanet discovery:** AI algorithms can be used to search for exoplanets by analyzing the data from telescopes. These algorithms can detect subtle changes in a star's brightness or position that might indicate the presence of an orbiting planet.

- **Galaxy classification:** AI can help classify galaxies based on their shape, size, and other properties. This can help astronomers better understand the structure and evolution of galaxies.

- **Astronomical simulations:** AI can be used to create simulations of astronomical phenomena, such as the formation of galaxies, the evolution of stars, and the behavior of black holes.

- **Prediction of astronomical events:** AI can be used to predict astronomical events such as supernovae or solar flares, which can be helpful in planning observations or in preparing for potential space weather events.

Our Plan

We Are Planning of Making a Software That will Predict the Type of Star Using The AI. It will do so based on the data base it have from past few years to increase its accuracy for the same.

Predicting astronomical events such as meteor showers, lunar and solar eclipses, and planetary alignments can be very challenging, but AI algorithms can certainly help improve accuracy.

Type Of Star Prediction

We Are Planning of Making a Software That will Predict the Type of Star Using The AI to generate specific Date & time for the event that's gonna take place. It will do so based on the data base it have from past few years to increase its accuracy for the event.

AI in Astronomy ☆

File Edit View Insert Runtime Tools Help Changes will not be saved

+ Code + Text Copy to Drive

```
# Import required modules
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import plotly.express as pex
import tensorflow as tf
import re
import math
import joblib
from mpl_toolkits.axes_grid1.inset_locator import inset_axes

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_predict
```

```
[ ] # Load the dataset
    hipparcos = pd.read_csv("/content/hipparcos-voidmain.csv")
    star_df = pd.read_csv("/content/hipparcos-voidmain.csv")
    star_df2 = pd.read_csv("/content/6class.csv")
```

```
[ ] # Show the first 5 rows of both the dataframe
    star_df.head()
```

	Catalog	HIP	Proxy	RAhms	DEdms	Vmag	VarFlag	r_Vmag	RAdeg	DEdeg	...	Survey	Chart	Notes	HD	BD	CoD	CPD	(V-I)red	SpType	r_SpType
0	H	1	NaN	00 00 00.22	+01 05 20.4	9.10	NaN	H	0.000912	1.089013	...	S	NaN	NaN	224700.0	B+00 5077	NaN	NaN	0.66	F5	S
1	H	2	NaN	00 00 00.91	-19 29 55.8	9.27	NaN	G	0.003797	-19.498837	...	NaN	NaN	NaN	224690.0	B-20 6688	NaN	NaN	1.04	K3V	4
2	H	3	NaN	00 00 01.20	+38 51 33.4	6.61	NaN	G	0.005008	38.859286	...	S	NaN	NaN	224699.0	B+38 5108	NaN	NaN	0.00	B9	S
3	H	4	NaN	00 00 02.01	-51 53 36.8	8.06	NaN	H	0.008382	-51.893546	...	S	NaN	NaN	224707.0	NaN	NaN	P-52 12237	0.43	F0V	2
4	H	5	NaN	00 00 02.39	-40 35 28.4	8.55	NaN	H	0.009965	-40.591224	...	NaN	NaN	NaN	224705.0	NaN	C-41 15372	P-41 9991	0.95	G8III	2

5 rows × 78 columns

5 rows × 78 columns

```
# Show info the hippocoras df
star_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 118218 entries, 0 to 118217
Data columns (total 78 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   Catalog    118218 non-null  object
 1   HIP        118218 non-null  int64
 2   Proxy      10925 non-null   object
 3   RAhms      118218 non-null  object
 4   DEdms      118218 non-null  object
 5   Vmag       118217 non-null  float64
 6   VarFlag    11562 non-null   float64
 7   r_Vmag     118217 non-null  object
 8   RAdeg      117955 non-null  float64
 9   DEdeg      117955 non-null  float64
 10  AstroRef   13734 non-null   object
 11  Plx        117955 non-null  float64
 12  pmRA       117955 non-null  float64
 13  pmDE       117955 non-null  float64
 14  e_RAdeg    117955 non-null  float64
 15  e_DEdeg    117955 non-null  float64
 16  e_Plx      117955 non-null  float64
 17  e_pmRA     117955 non-null  float64
 18  e_pmDE     117955 non-null  float64
 19  DE:RA      117955 non-null  float64
 20  Plx:RA     117955 non-null  float64
 21  Plx:DE     117955 non-null  float64
 22  pmRA:RA    117955 non-null  float64
 23  pmRA:DE    117955 non-null  float64
 24  pmRA:Plx   117955 non-null  float64
 25  pmDE:RA    117955 non-null  float64
 26  pmDE:DE    117955 non-null  float64
 27  pmDE:Plx   117955 non-null  float64
 28  pmDE:pmRA  117955 non-null  float64
 29  F1         117955 non-null  float64
 30  F2         116392 non-null  float64
 31  ---        118218 non-null  int64
 32  BTmag      114820 non-null  float64
 33  e_BTmag    114820 non-null  float64
 34  VTmag      114879 non-null  float64
 35  e_VTmag    114879 non-null  float64
 36  m_BTmag    12796 non-null   object
 37  B-V        116927 non-null  float64
```

function.

Apparent magnitude is a measure of the brightness of a celestial object as observed from Earth and can vary due to distance from the object and atmospheric conditions. Absolute magnitude is a distance-independent measure and is useful for comparing the brightness of different celestial objects.

The formula is based on the fact that the apparent magnitude of an object is inversely proportional to its distance. As the distance increases, the apparent magnitude decreases. Therefore, subtracting the logarithm of the distance in parsecs divided by 10 from the apparent magnitude, gives the absolute magnitude.

temperature of a star:

$$T = 4600K\left(\frac{1}{0.92(B-V)+1.7}+\frac{1}{1.5(B-V)+0.62}\right)$$

```
[ ] # Filter Parallax
    star_df = star_df[star_df["Plx"]>0.000001]
    star_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 113710 entries, 0 to 118217
Data columns (total 78 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   Catalog    113710 non-null  object
 1   HIP        113710 non-null  int64
 2   Proxy      10553 non-null   object
 3   RAhms      113710 non-null  object
 4   DEdms      113710 non-null  object
 5   Vmag       113710 non-null  float64
 6   VarFlag    10508 non-null   float64
 7   r_Vmag     113710 non-null  object
 8   RAdeg      113710 non-null  float64
 9   DEdeg      113710 non-null  float64
 10  AstroRef   13035 non-null   object
 11  Plx        113710 non-null  float64
 12  pmRA       113710 non-null  float64
 13  pmDE       113710 non-null  float64
 14  e_RAdeg    113710 non-null  float64
 15  e_DEdeg    113710 non-null  float64
 16  e_Plx      113710 non-null  float64
 17  e_pmRA     113710 non-null  float64
 18  e_pmDE     113710 non-null  float64
 19  DE:RA      113710 non-null  float64
 20  Plx:RA     113710 non-null  float64
```

+ Code + Text

```
[ ] # Remove null values of B-V
    star_df = star_df.dropna(subset = ["B-V"])
    star_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 112823 entries, 0 to 118217
Data columns (total 78 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   Catalog    112823 non-null  object
 1   HIP        112823 non-null  int64
 2   Proxy      10291 non-null   object
 3   RAhms      112823 non-null  object
 4   DEdms      112823 non-null  object
 5   Vmag       112823 non-null  float64
 6   VarFlag    10312 non-null   float64
 7   r_Vmag     112823 non-null  object
 8   RAdeg      112823 non-null  float64
 9   DEdeg      112823 non-null  float64
 10  AstroRef   12660 non-null   object
 11  Plx        112823 non-null  float64
 12  pmRA       112823 non-null  float64
 13  pmDE       112823 non-null  float64
 14  e_RAdeg    112823 non-null  float64
 15  e_DEdeg    112823 non-null  float64
 16  e_Plx      112823 non-null  float64
 17  e_pmRA     112823 non-null  float64
 18  e_pmDE     112823 non-null  float64
 19  DE:RA      112823 non-null  float64
 20  Plx:RA     112823 non-null  float64
 21  Plx:DE     112823 non-null  float64
 22  pmRA:RA    112823 non-null  float64
 23  pmRA:DE    112823 non-null  float64
 24  pmRA:Plx   112823 non-null  float64
 25  pmDE:RA    112823 non-null  float64
 26  pmDE:DE    112823 non-null  float64
 27  pmDE:Plx   112823 non-null  float64
 28  pmDE:pmRA  112823 non-null  float64
 29  F1         112823 non-null  float64
 30  F2         111503 non-null  float64
 31  ---        112823 non-null  int64
 32  BTmag      110719 non-null  float64
 33  e_BTmag    110719 non-null  float64
 34  VTmag      110772 non-null  float64
 35  e_VTmag    110772 non-null  float64
 36  m_BTmag    11970 non-null   object
 37  B-V        112823 non-null  float64
```

AI in Astronomy ☆

File Edit View Insert Runtime Tools Help Changes will not be saved

+ Code + Text Copy to Drive

```
[ ] # Get values of bmv, m, p
    bmv = np.array(star_df["B-V"], dtype=float)
    m = np.array(star_df["Vmag"], dtype=float)
    p = np.array(star_df["Plx"], dtype=float)
```

```
# Calculate Distance with Parallax in milliarcseconds
# Calculate the temperature
# Calculate the absolute magnitude

d = 1/p*1000
t = np.array(4600 * (1/(0.92 * bmv + 1.7) + 1/(0.92 * bmv + 0.62)))
M = m - 5 * np.log10(d/10)

print(d, t, M, sep="\n")
```

```
[282.48587571  45.66210046 355.87188612 ... 200.         52.02913632
 114.81056257]
[ 6471.66782641  4745.14042459 10368.59558776 ...  4745.14042459
  5608.54490151 11168.81296033]
[ 1.84501631  5.97222057 -1.1464684  ...  1.08485002  5.61876692
 -0.80990922]
```

```
# Line that shows the flow of movement in the graph of the stars throughout their lives.
sequence_principal_x = [-0.1,0, .2,0.5, 1, 1.4, 1.713]
sequence_principal_y = [0,-1.42, -2, -4, -6.5, -9, -13.21]

whiteDwarfs_x = [-0.2, -0.03, 0.5,1.5]
whiteDwarfs_y = [-10, -11.18,-13, -14]

SubGiants_x = [-0.1, .5 ,1.613]
SubGiants_y = [1, -2.5 , -4]

SuperGiants_x = [-0.2, 1.613]
SuperGiants_y = [9.5, 10]

giants_x = [-0.1,0.35 ,0.8 ,1.2 ,1.713]
giants_y = [2,0 ,-0.7, 0,1.231]

giantsBrillant_x = [-0.1,0.8 ,1.713]
giantsBrillant_y = [5, 4 ,5]
```

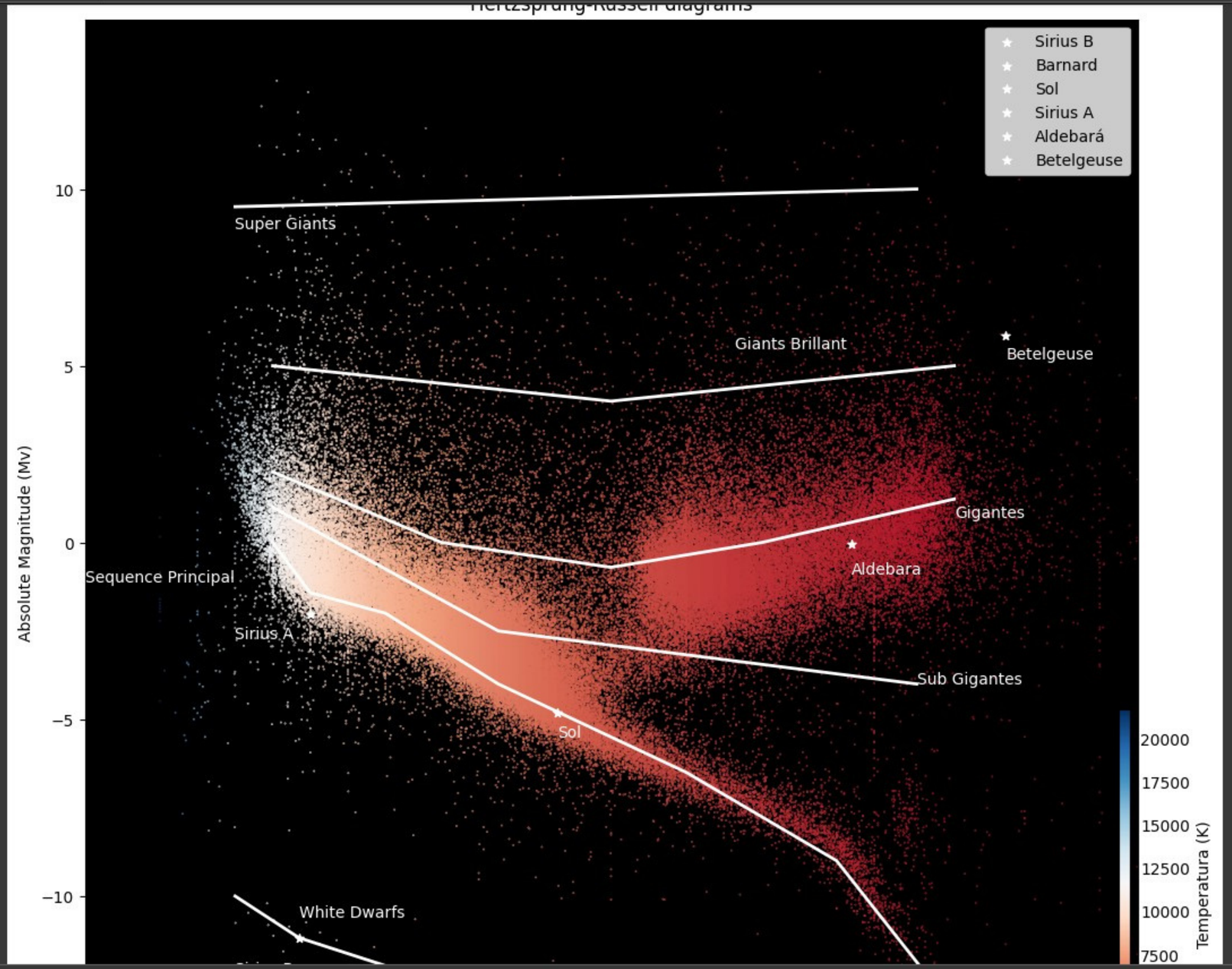
```
[ ] # Plot the graph
```

```
# Plot the graph
fig, ax = plt.subplots(figsize=(12, 12))
scatter = ax.scatter(bmv, -M, c=t, cmap="RdBu", marker='.', s=.5)

ax.scatter(x=-0.03, y=-11.18, c="white", s=30, label="Sirius B", marker="*")
ax.scatter(x=1.713, y=-13.21, c="white", s=30, label="Barnard", marker="*")
ax.scatter(x=0.656, y=-4.83, c="white", s=30, label="Sol", marker="*")
ax.scatter(x=0, y=-2.02, c="white", s=30, label="Sirius A", marker="*")
ax.scatter(x=1.44, y=-0.031, c="white", s=30, label="Aldebará", marker="*")
ax.scatter(x=1.85, y=5.85, c="white", s=30, label="Betelgeuse", marker="*")

# Set xlabel, ylabel and title
plt.xlabel("Color Index (B-V)")
plt.ylabel("Absolute Magnitude (Mv)")
plt.title("Hertzsprung-Russell diagrams")
plt.legend()

# Set x and y limit
plt.xlim(-0.6, 2.2)
plt.ylim(-14.8, 14.8)

# Plot the stars
plt.plot(sequence_principal_x, sequence_principal_y, c="white", linewidth=2)
plt.plot(whiteDwarfs_x, whiteDwarfs_y, c="white", linewidth=2)
plt.plot(SubGiants_x, SubGiants_y, c="white", linewidth=2)
plt.plot(giants_x, giants_y, c="white", linewidth=2)
plt.plot(SuperGiants_x, SuperGiants_y, c="white", linewidth=2)
plt.plot(giantsBrillant_x, giantsBrillant_y, c="white", linewidth=2)

# Write the name of stars
plt.annotate("Super Giants", (-0.2, 8.9), c="white")
plt.annotate("Giants Brillant", (1.13, 5.5), c="white")
plt.annotate("Sequence Principal", (-0.6, -1.1), color="white")
plt.annotate("White Dwarfs", (-0.03, -10.6), color="white")
plt.annotate("Sub Gigantes", (1.613, -4), color="white")
plt.annotate("Gigantes", (1.713, 0.7), color="white")
plt.annotate("Sol", (0.656, -5.5), color="white")
plt.annotate("Betelgeuse", (1.85, 5.2), color="white")
plt.annotate("Aldebara", (1.44, -0.9), color="white")
plt.annotate("Barnard", (1.713, -14.2), color="white")
plt.annotate("Sirius B", (-0.2,-12.2), color="white")
plt.annotate("Sirius A", (-0.2, -2.72), c="white")
```

AI in Astronomy

File Edit View Insert Runtime Tools Help Changes will not be saved

+ Code + Text Copy to Drive



2.2. Color, Temperature and Spectral Classes

```
# Correct the naming for the values.
star_df2["Star color"] = star_df2["Star color"].apply(lambda x: re.sub(r"\s", "-", x))
star_df2["Star color"] = star_df2["Star color"].str.replace("Blue-white", "Blue-White")
star_df2["Star color"] = star_df2["Star color"].str.replace("white", "White")
star_df2["Star color"] = star_df2["Star color"].str.replace("Blue-", "Blue")
star_df2["Star color"] = star_df2["Star color"].str.replace("BlueWhite-", "BlueWhite")

star_df2["Star color"].value_counts(normalize=False, sort=True)
```

```
Red                 112
Blue                 56
BlueWhite            41
White                10
yellow-White          8
Yellowish-White       3
Whitish               2
Orange                2
yellowish             2
Pale-yellow-orange    1
White-Yellow          1
Yellowish             1
Orange-Red            1
Name: Star color, dtype: int64
```

```
[ ] # Show star color and temperature relation
Mv_color = pd.concat([star_df2["Star color"], star_df2["Temperature (K)"]], axis=1)

fig = plt.figure(figsize=(10, 8))
axes = sns.barplot(data= Mv_color, x= "Star color", y="Temperature (K)", palette= "Accent")
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12]),
 [Text(0, 0, 'Red'),
  Text(1, 0, 'BlueWhite'),
  Text(2, 0, 'White'),
  Text(3, 0, 'Yellowish-White'),
  Text(4, 0, 'Pale-yellow-orange'),
  Text(5, 0, 'Blue'),
  Text(6, 0, 'Whitish'),
  Text(7, 0, 'yellow-White'),
  Text(8, 0, 'Orange'),
  Text(9, 0, 'White-Yellow')
```

Classification of stars in different evolutionary stages

```
[ ] # Get Vmag, Plx, B-V
    hipparcos = hipparcos.filter(["Vmag", "Plx", "B-V"])

    # Calculate Distance
    hipparcos["Distance"] = (1 / hipparcos["Plx"]) * 1000

    # Calculate Absolute Magnitude
    hipparcos["Absolute Magnitude"] =  hipparcos["Vmag"] - 5 * np.log10(hipparcos["Distance"]) + 5

    # Calculate effective temperature
    hipparcos["Teff"] = np.array(4600 * (1 / (0.92 * hipparcos["B-V"] + 1.7) + 1 / (0.92 * hipparcos["B-V"] + 0.62)))
```

```
/usr/local/lib/python3.9/dist-packages/pandas/core/arraylike.py:397: RuntimeWarning: invalid value encountered in log10
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

```
# Drop null values
hipparcos = hipparcos.dropna()
hipparcos = hipparcos.reset_index()
```

```
[ ] # Drop index column
    hipparcos.drop(columns="index", inplace=True)
```

```
[ ] # Filter the data
    hipparcos["SpClass"] = pd.cut(hipparcos["Teff"], bins= [1000, 3500, 5300, 6000, 7500, 10000, 20000, 40000], labels = ["M", "K", "G", "F", "A", "B", "O"])
    hipparcos["Star Type"] = pd.cut(hipparcos["Absolute Magnitude"], bins=[-15, -10, -5, 0, 4, 15], labels=[5, 4, 3, 2, 1])
```

```
[ ] # Set the conditions
    conditions = [((hipparcos["SpClass"]=="M") & (hipparcos["Star Type"]==1)),
                  ((hipparcos["SpClass"]=="K") & (hipparcos["Star Type"]==1)),
                  ((hipparcos["SpClass"]=="G") & (hipparcos["Star Type"]==1)),
                  ((hipparcos["SpClass"]=="F") & (hipparcos["Star Type"]==1)),
                  ((hipparcos["SpClass"]=="A") & (hipparcos["Star Type"]==1)),
                  ((hipparcos["SpClass"]=="B") & (hipparcos["Star Type"]==1)),
                  ((hipparcos["SpClass"]=="O") & (hipparcos["Star Type"]==1)),
                  ((hipparcos["SpClass"]=="M") & (hipparcos["Star Type"]==2)),
                  ((hipparcos["SpClass"]=="K") & (hipparcos["Star Type"]==2)),
                  ((hipparcos["SpClass"]=="G") & (hipparcos["Star Type"]==2)),
                  ((hipparcos["SpClass"]=="F") & (hipparcos["Star Type"]==2)),
```

	Vmag	Plx	B-V	Distance	Absolute Magnitude	Teff	SpClass	Star Type	Epochs
0	9.10	3.54	0.482	282.485876	1.845016	6471.667826	F	2	Main Sequence
1	9.27	21.90	0.999	45.662100	5.972221	4745.140425	K	1	Main Sequence
2	6.61	2.81	-0.019	355.871886	-1.146468	10368.595588	B	3	Main Sequence
3	8.06	7.75	0.370	129.032258	2.506509	7044.130880	F	2	Main Sequence
4	8.55	2.87	0.902	348.432056	0.839409	4991.060700	K	2	Red Giants
...	...	...	...	...	...	...	...	...	...
112862	6.99	1.92	1.595	520.833333	-1.593494	3655.993639	K	3	Red Giants
112863	8.23	10.63	0.639	94.073377	3.362666	5818.920292	G	2	Main Sequence
112864	7.59	5.00	0.999	200.000000	1.084850	4745.140425	K	2	Red Giants
112865	9.20	19.22	0.698	52.029136	5.618767	5608.544902	G	1	Main Sequence
112866	4.49	8.71	-0.075	114.810563	-0.809909	11168.812960	B	3	Main Sequence

112867 rows × 9 columns

```
[ ] # Drop null values
    hipparcos.dropna(inplace=True)
```

```
[ ] # Create a LabelEncoder object
    le = LabelEncoder()

    # Convert categorical labels to unique numeric values
    hipparcos.Epochs = le.fit_transform(hipparcos.Epochs)
    hipparcos.SpClass = le.fit_transform(hipparcos.SpClass)
```

```
[ ] # Get target
    target = hipparcos.iloc[:, 8]
    hipparcos_t = hipparcos.iloc[:, :-1]
```

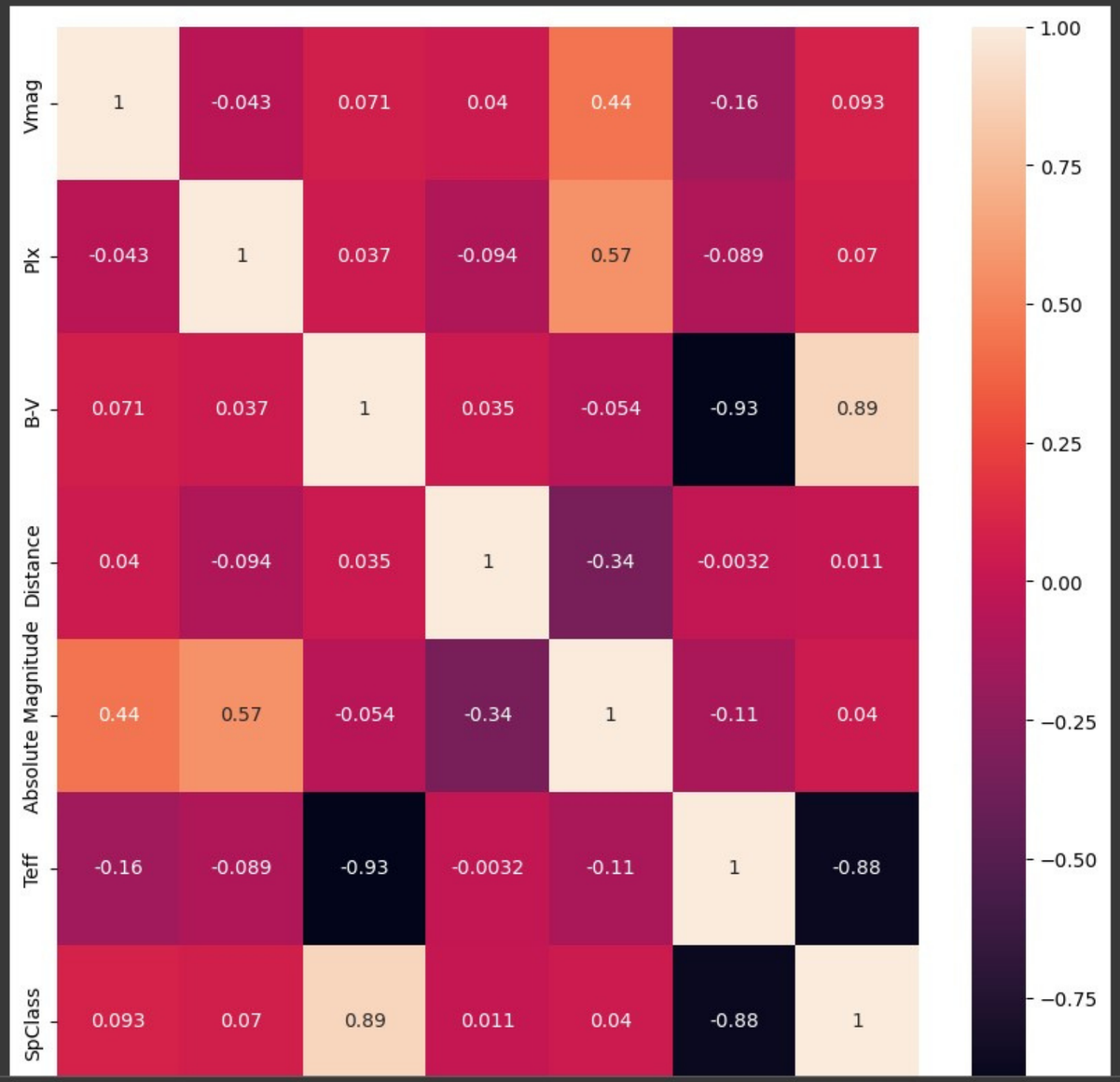
```
[ ] # Transform features
    ss = StandardScaler()
    features = ss.fit_transform(hipparcos_t)
```

AI in Astronomy

File Edit View Insert Runtime Tools Help Changes will not be saved

+ Code + Text Copy to Drive

```
# Show coorelation
fig = plt.figure(figsize=(10, 10))
matrizC = sns.heatmap(features.corr(), annot=True)
```



AI in Astronomy ☆

File Edit View Insert Runtime Tools Help Changes will not be saved

+ Code + Text Copy to Drive

```
[ ] # Drop null values
    hipparcos = hipparcos.dropna()
    hipparcos = hipparcos.reset_index()
```

```
[ ] # Drop index column
    hipparcos.drop(columns="index", inplace=True)
```

```
[ ] # Filter the data
    hipparcos["SpClass"] = pd.cut(hipparcos["Teff"], bins= [1000, 3500, 5300, 6000, 7500, 10000, 20000, 40000], labels = ["M", "K", "G", "F", "A", "B", "O"])
    hipparcos["Star Type"] = pd.cut(hipparcos["Absolute Magnitude"], bins=[-15, -10, -5, 0, 4, 15], labels=[5, 4, 3, 2, 1])
```

```
# Set the conditions
conditions = [((hipparcos["SpClass"]=="M") & (hipparcos["Star Type"]==1)),
              ((hipparcos["SpClass"]=="K") & (hipparcos["Star Type"]==1)),
              ((hipparcos["SpClass"]=="G") & (hipparcos["Star Type"]==1)),
              ((hipparcos["SpClass"]=="F") & (hipparcos["Star Type"]==1)),
              ((hipparcos["SpClass"]=="A") & (hipparcos["Star Type"]==1)),
              ((hipparcos["SpClass"]=="B") & (hipparcos["Star Type"]==1)),
              ((hipparcos["SpClass"]=="O") & (hipparcos["Star Type"]==1)),
              ((hipparcos["SpClass"]=="M") & (hipparcos["Star Type"]==2)),
              ((hipparcos["SpClass"]=="K") & (hipparcos["Star Type"]==2)),
              ((hipparcos["SpClass"]=="G") & (hipparcos["Star Type"]==2)),
              ((hipparcos["SpClass"]=="F") & (hipparcos["Star Type"]==2)),
              ((hipparcos["SpClass"]=="A") & (hipparcos["Star Type"]==2)),
              ((hipparcos["SpClass"]=="B") & (hipparcos["Star Type"]==2)),
              ((hipparcos["SpClass"]=="O") & (hipparcos["Star Type"]==2)),
              ((hipparcos["SpClass"]=="M") & (hipparcos["Star Type"]==3)),
              ((hipparcos["SpClass"]=="K") & (hipparcos["Star Type"]==3)),
              ((hipparcos["SpClass"]=="G") & (hipparcos["Star Type"]==3)),
              ((hipparcos["SpClass"]=="F") & (hipparcos["Star Type"]==3)),
              ((hipparcos["SpClass"]=="A") & (hipparcos["Star Type"]==3)),
              ((hipparcos["SpClass"]=="B") & (hipparcos["Star Type"]==3)),
              ((hipparcos["SpClass"]=="O") & (hipparcos["Star Type"]==3))]

# Set the choice
choices = [
  'Main Sequence',
  'Main Sequence',
  'Main Sequence',
  'Main Sequence',
  'white dwarfs',
```


```
[ ] # Create train test split
    X_train, X_test, y_train, y_test = train_test_split(features, target, random_state=42)
```

```
[ ] # Create RFC model
    rfc = RandomForestClassifier(random_state=42)
    rfc.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
[ ] # Use the confusion matrix
    y_train_pred = cross_val_predict(rfc, X_train, y_train, cv=3)
    confusion_matrix(y_train, y_train_pred)
```

```
array([[  656,     0,     0,     0],
       [    0, 53889,     0,     0],
       [    0,     0, 29952,     0],
       [    0,     1,     0,   118]])
```

```
[ ] # Calculate precision and recall with the new classifier
    p = precision_score(y_train, y_train_pred, average="macro")
    r = recall_score(y_train, y_train_pred, average="macro")
    p, r
```

```
(0.9999953609203934, 0.9978991596638656)
```

$$Precision = \frac{True Positives}{(True Positives + False Positives)}$$


Thank You!