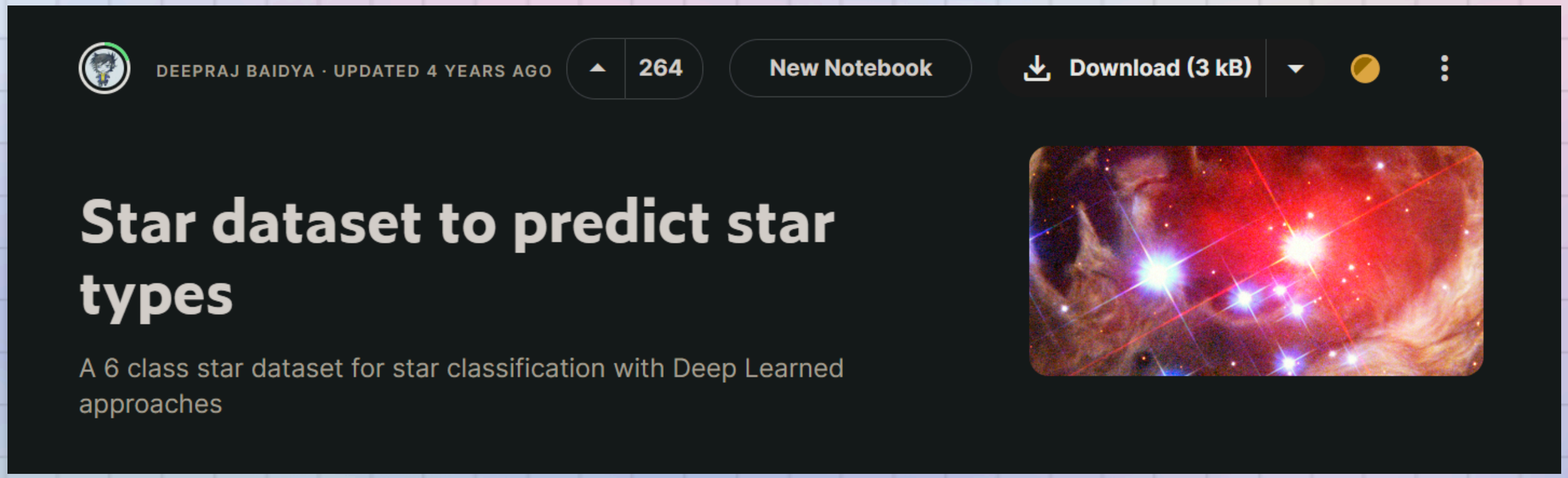
April 16, 2024

STAR TYPES CLASSIFICATION

By:
Duong Thu Phuong
Pham Ngoc Cuong
Nguyen Quang Huy

Source of data



Source of data

data.csv — Open with

	A	B	C	D	E	F	G
1	Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(	Star type	Star color	Spectral Class
2	3068	0.0024	0.17	16.12	0	Red	M
3	3042	0.0005	0.1542	16.6	0	Red	M
4	2600	0.0003	0.102	18.7	0	Red	M
5	2800	0.0002	0.16	16.65	0	Red	M
6	1939	0.000138	0.103	20.06	0	Red	M
7	2840	0.00065	0.11	16.98	0	Red	M
8	2637	0.00073	0.127	17.22	0	Red	M
9	2600	0.0004	0.096	17.4	0	Red	M
10	2650	0.00069	0.11	17.45	0	Red	M
11	2700	0.00018	0.13	16.05	0	Red	M
12	3600	0.0029	0.51	10.69	1	Red	M
13	3129	0.0122	0.3761	11.79	1	Red	M
14	3134	0.0004	0.196	13.21	1	Red	M
15	3628	0.0055	0.393	10.48	1	Red	M
16	2650	0.0006	0.14	11.782	1	Red	M
17	3340	0.0038	0.24	13.07	1	Red	M
18	2799	0.0018	0.16	14.79	1	Red	M
19	3692	0.00367	0.47	10.8	1	Red	M
20	3192	0.00362	0.1967	13.53	1	Red	M
21	3441	0.039	0.351	11.18	1	Red	M
22	25000	0.056	0.0084	10.58	2	Blue White	B
23	7740	0.00049	0.01234	14.02	2	White	A
24	7220	0.00017	0.011	14.23	2	White	F
25	8500	0.0005	0.01	14.5	2	White	A
26	16500	0.013	0.014	11.89	2	Blue White	B

Source of data

1. Brown Dwarf → Star Type = 0

2. Red Dwarf → Star Type = 1

3. White Dwarf→ Star Type = 2

4. Main Sequence → Star Type = 3

5. Supergiant → Star Type = 4

6. Hypergiant → Star Type = 5

The Luminosity and radius of each star is calculated w.r.t. that of the values of Sun.

Lo = 3.828 × 10^26 Watts

Ro = 6.9551 × 10^8 m

Import Libraries

```
[ ]  import numpy as np
     import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import MinMaxScaler
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
     import seaborn as sns
     import matplotlib.pyplot as plt
```

Import Data

```
data = pd.read_csv('data.csv')
data.head()
```

	Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class
0	3068	0.002400	0.1700	16.12	0	Red	M
1	3042	0.000500	0.1542	16.60	0	Red	M
2	2600	0.000300	0.1020	18.70	0	Red	M
3	2800	0.000200	0.1600	16.65	0	Red	M
4	1939	0.000138	0.1030	20.06	0	Red	M

Check for missing values

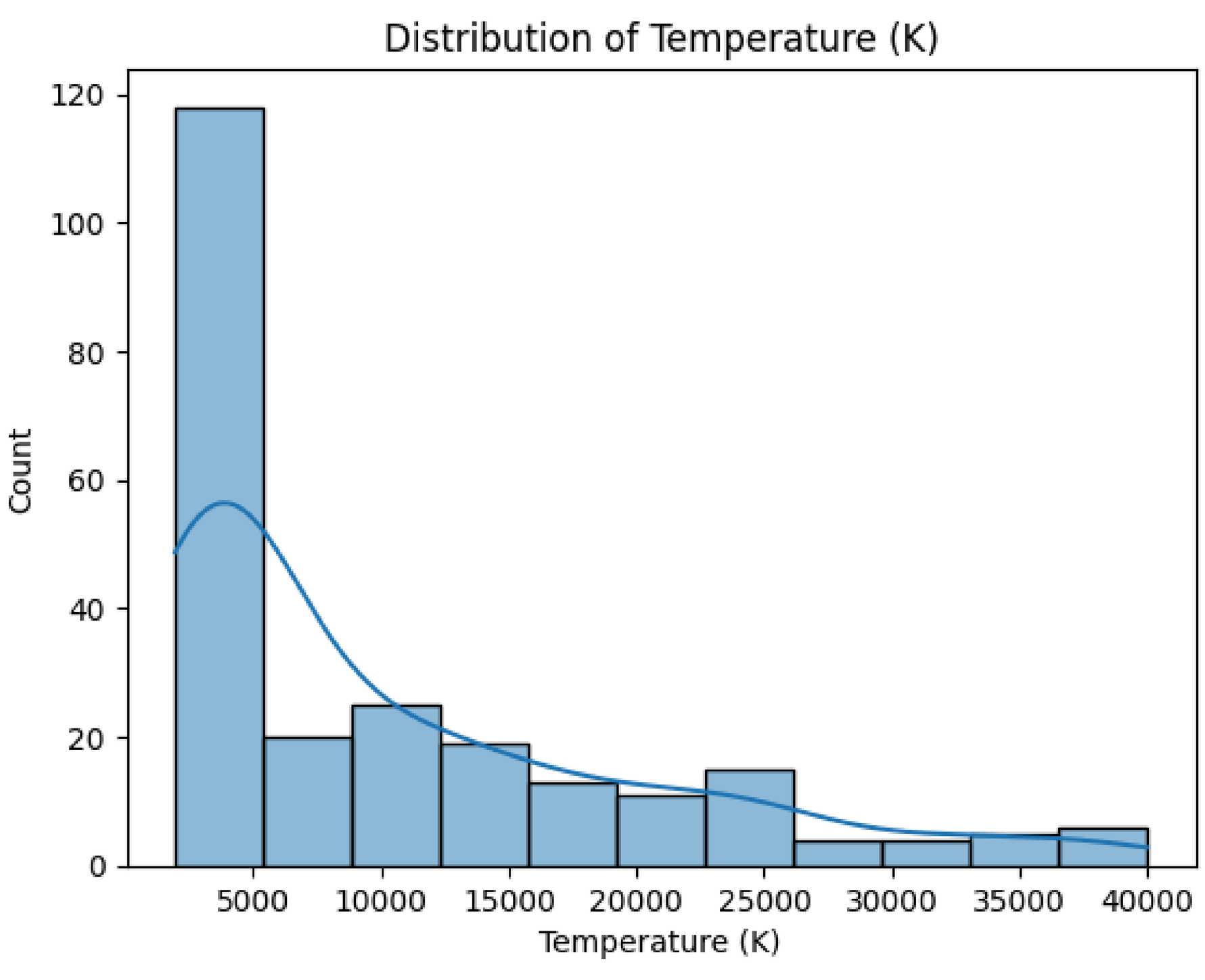
```
data.info()
```

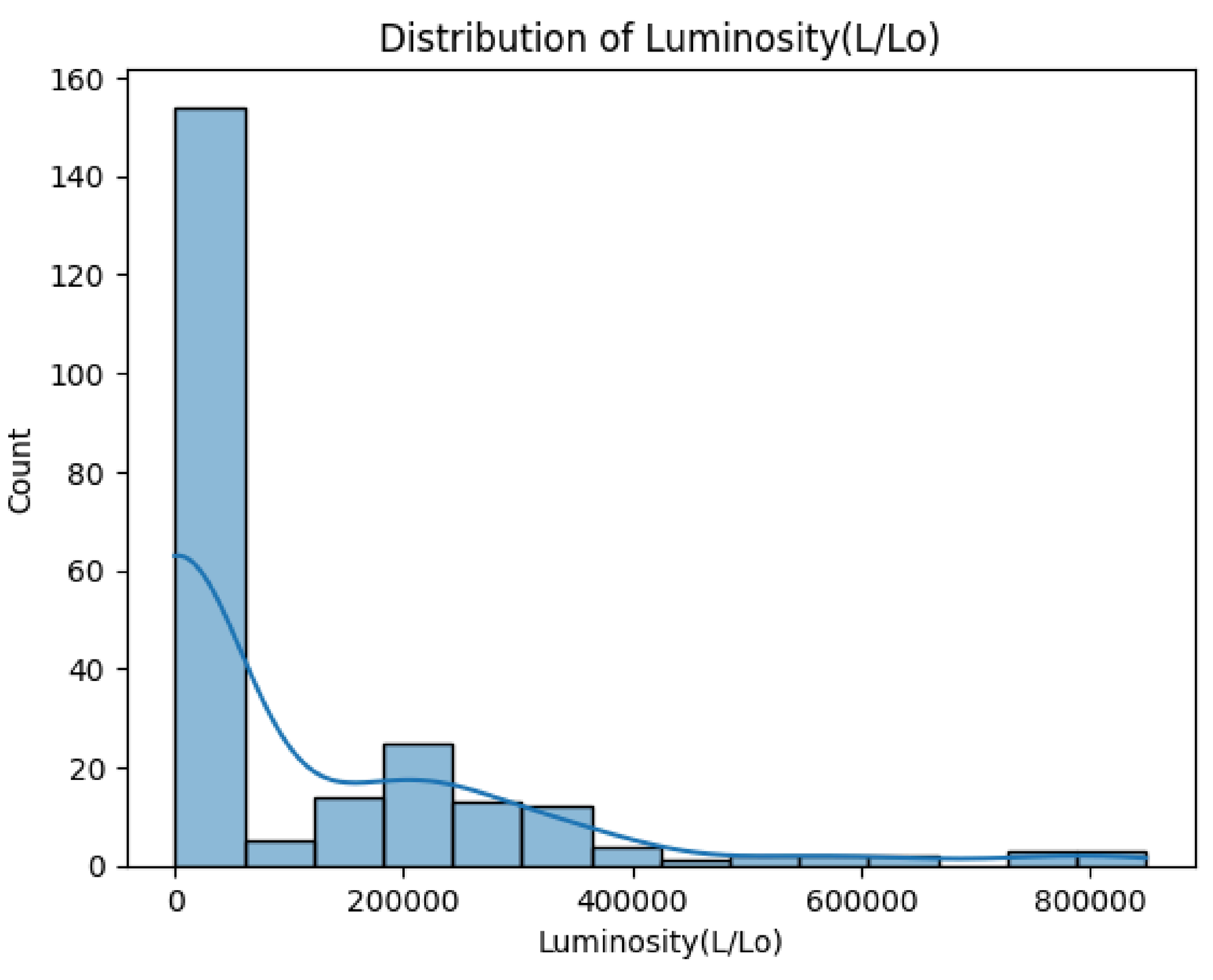
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 240 entries, 0 to 239
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Temperature (K)         240 non-null    int64
 1   Luminosity(L/Lo)        240 non-null    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
```

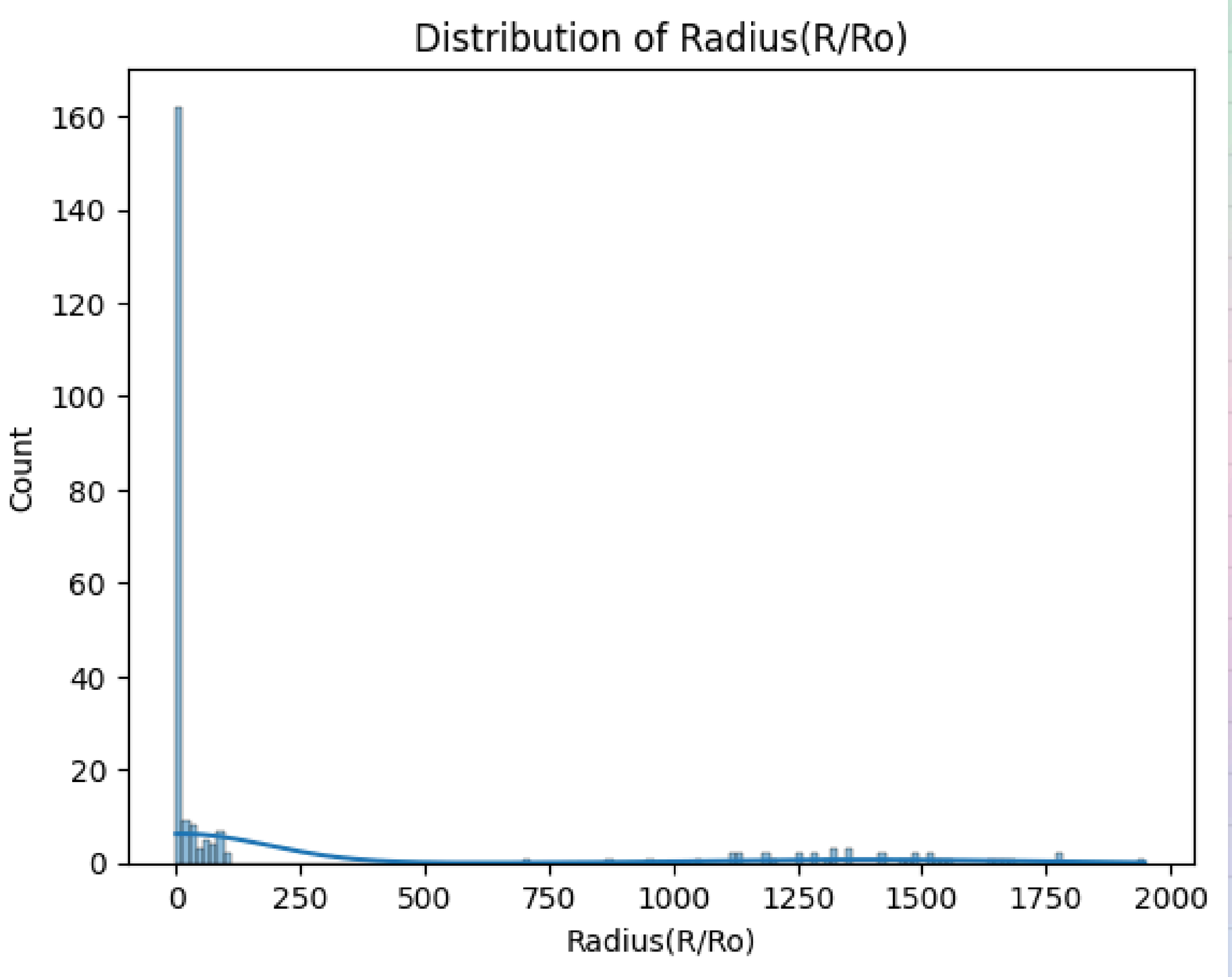
Distribution of Features

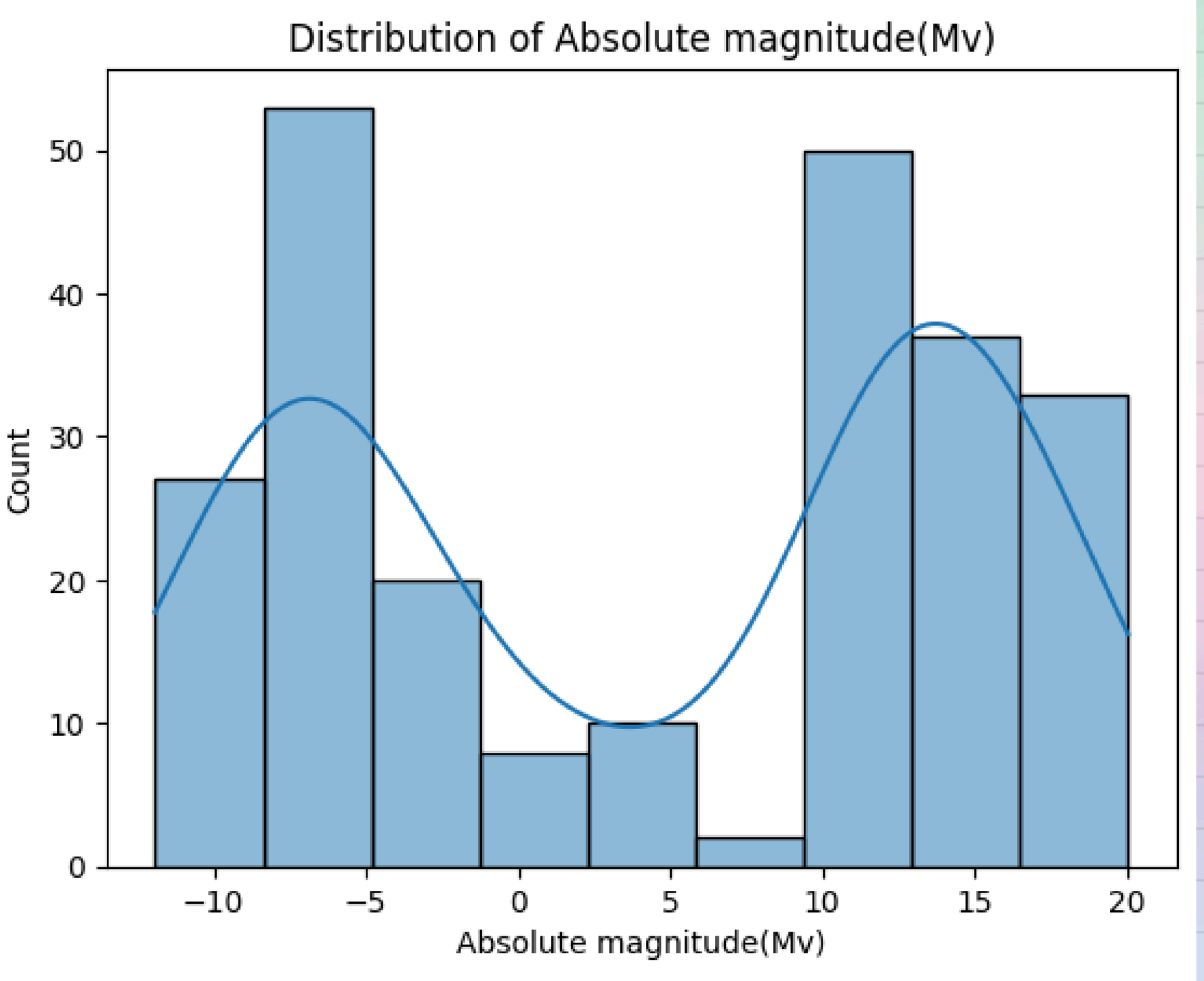
```
feature_cols = data.columns

for feature in feature_cols:
    sns.histplot(data=data, x=feature, kde=True)
    plt.xlabel(feature)
    plt.ylabel('Count')
    plt.title(f'Distribution of {feature}')
    plt.show()
```


Distribution of Temperature (K)
Count
Temperature (K)
0
20
40
60
80
100
120
5000
10000
15000
20000
25000
30000
35000
40000

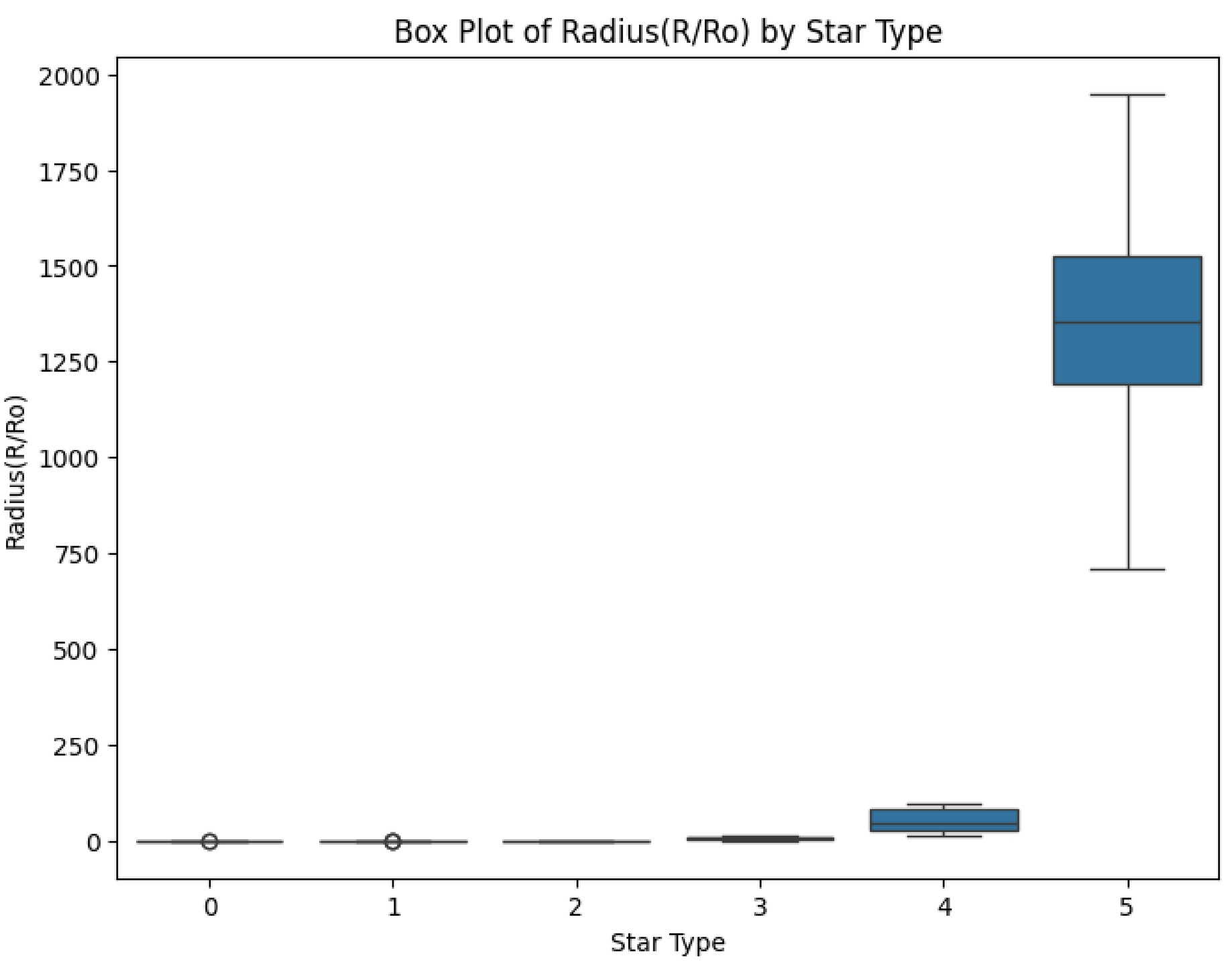

Distribution of Luminosity(L/Lo)
Count
160
140
120
100
80
60
40
20
0
0
200000
400000
600000
800000
Luminosity(L/Lo)

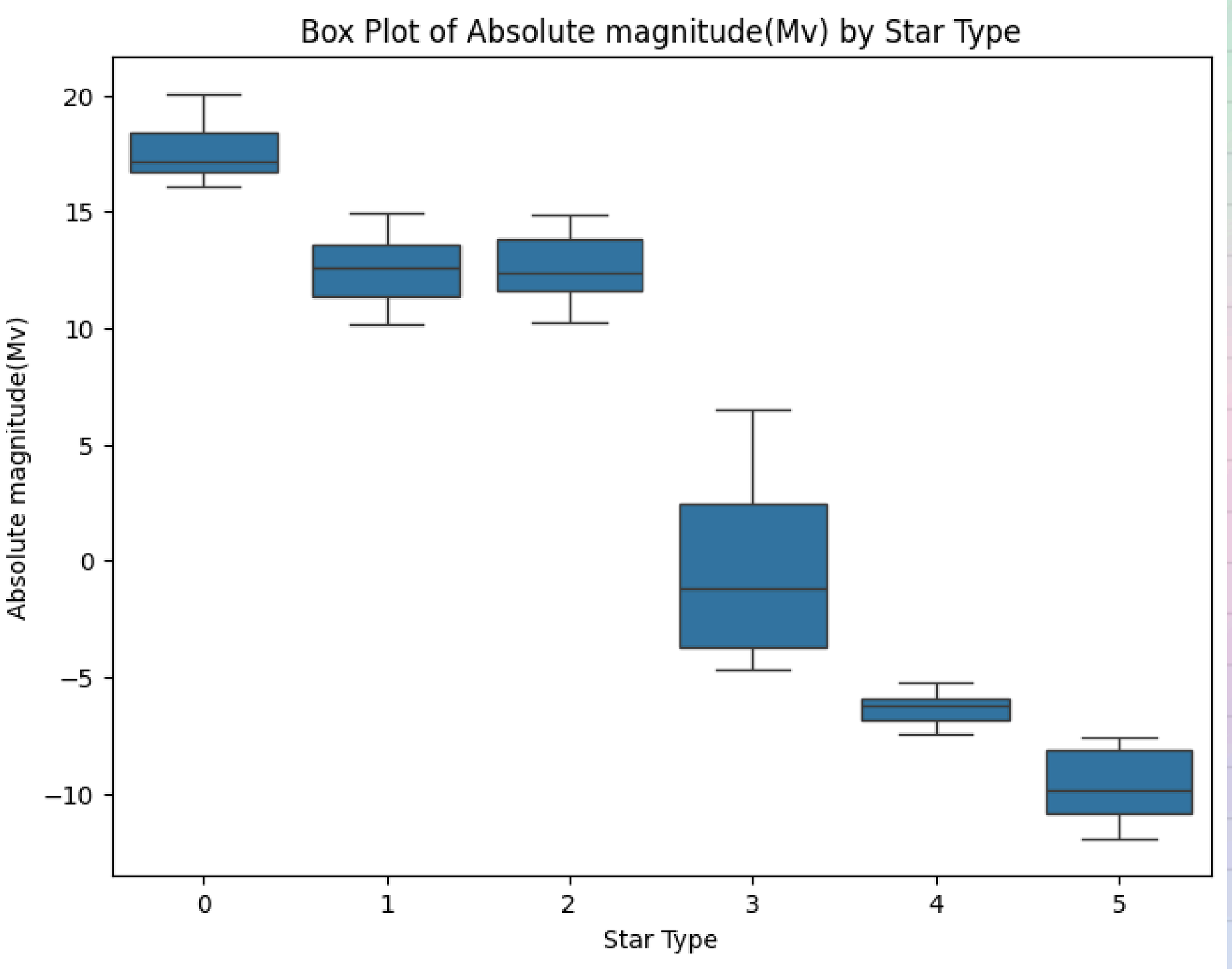

Distribution of Radius(R/Ro)
Count
0
20
40
60
80
100
120
140
160
0
250
500
750
1000
1250
1500
1750
2000
Radius(R/Ro)

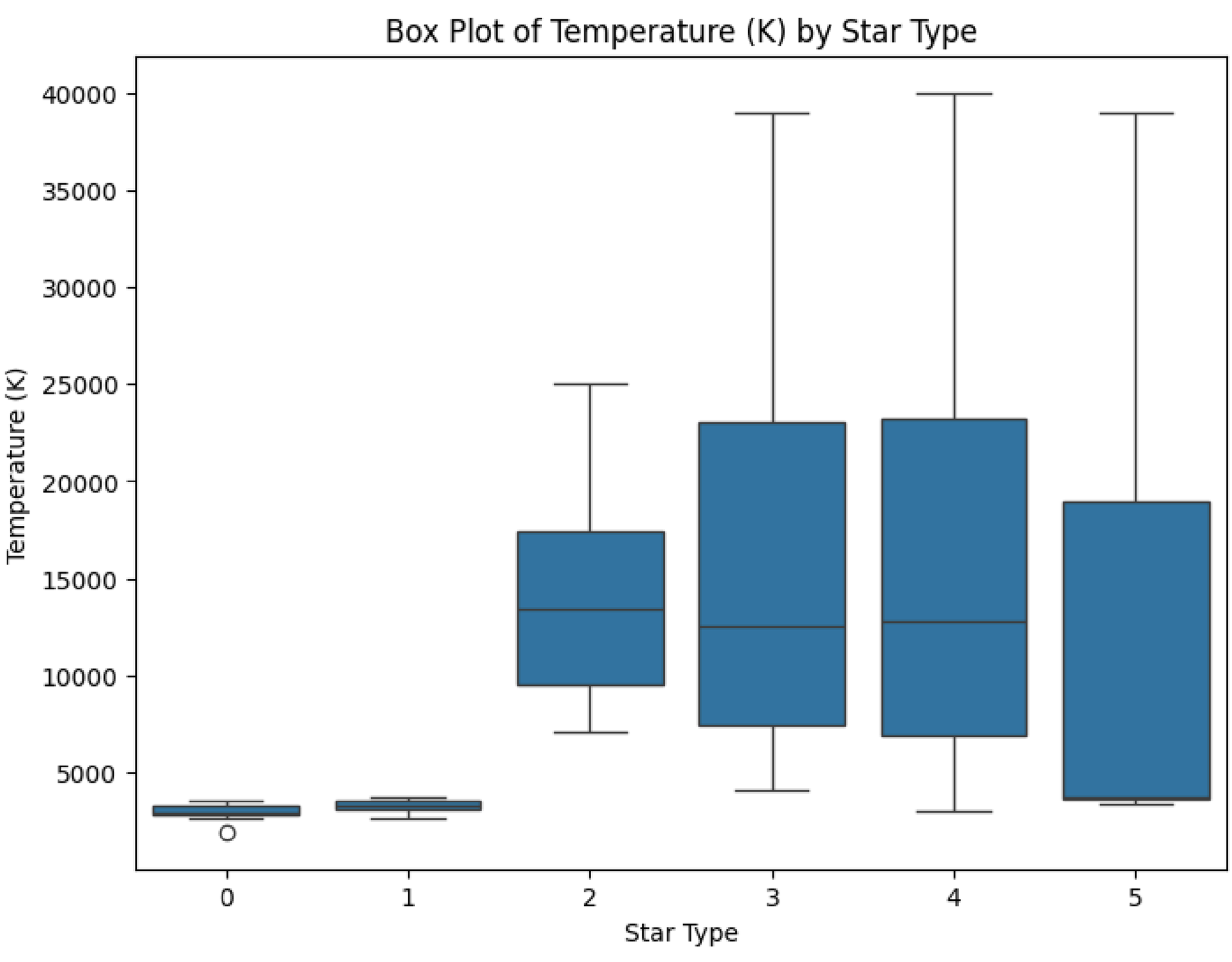

Distribution of Absolute magnitude(Mv)
Count
0
10
20
30
40
50
−10
−5
0
5
10
15
20
Absolute magnitude(Mv)

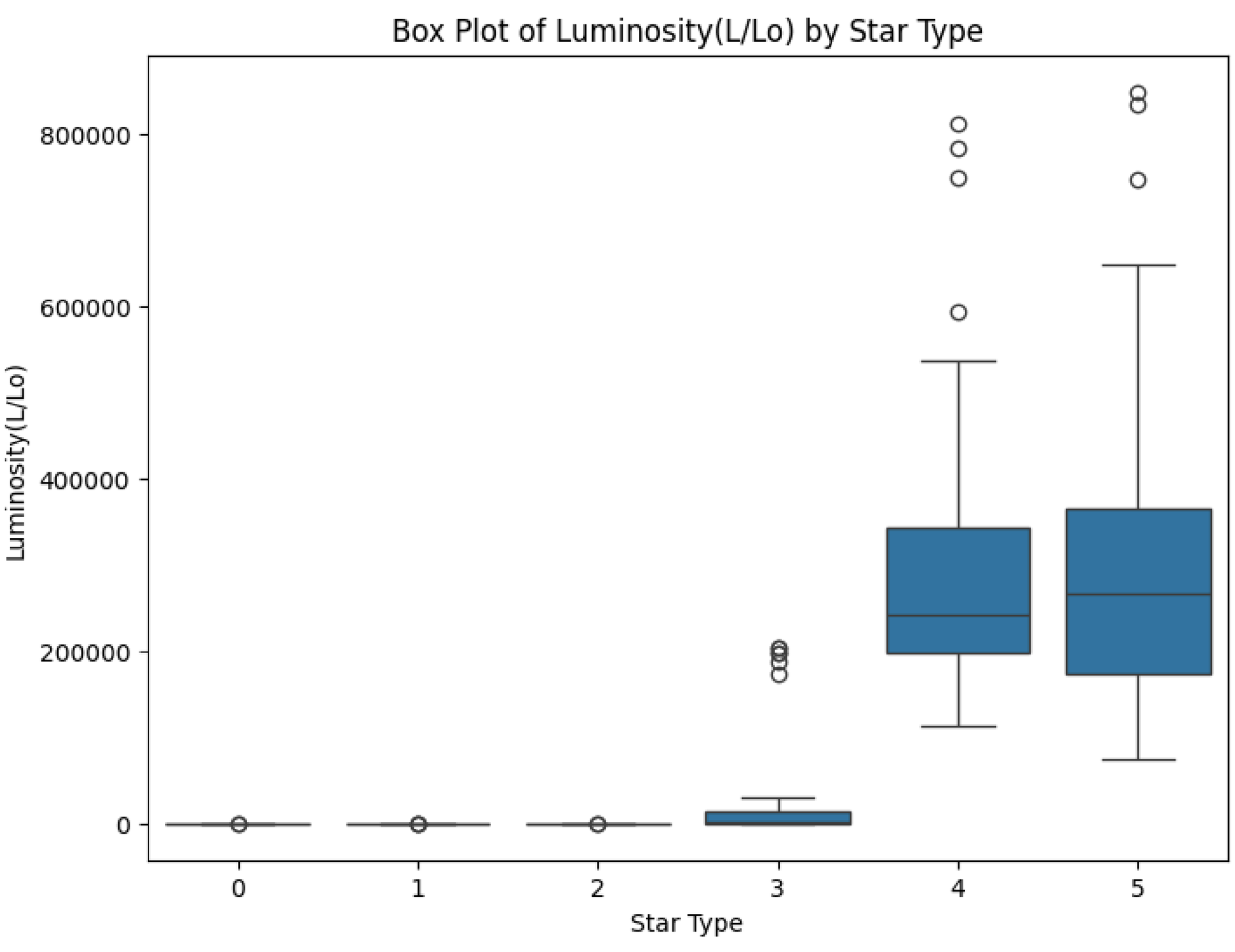
Box Plot

```
for feature in feature_cols:
    plt.figure(figsize=(8, 6))
    sns.boxplot(x='Star type', y=feature, data=data)
    plt.xlabel('Star Type')
    plt.ylabel(feature)
    plt.title(f'Box Plot of {feature} by Star Type')
    plt.show()
```


Box Plot of Radius(R/Ro) by Star Type
Radius(R/Ro)
2000
1750
1500
1250
1000
750
500
250
0
0
1
2
3
4
5
Star Type


Box Plot of Absolute magnitude(Mv) by Star Type
Absolute magnitude(Mv)
20
15
10
5
0
−5
−10
0
1
2
3
4
5
Star Type


Box Plot of Temperature (K) by Star Type
Temperature (K)
40000
35000
30000
25000
20000
15000
10000
5000
0
1
2
3
4
5
Star Type


Box Plot of Luminosity(L/Lo) by Star Type
Luminosity(L/Lo)
800000
600000
400000
200000
0
0
1
2
3
4
5
Star Type

Create dummy data

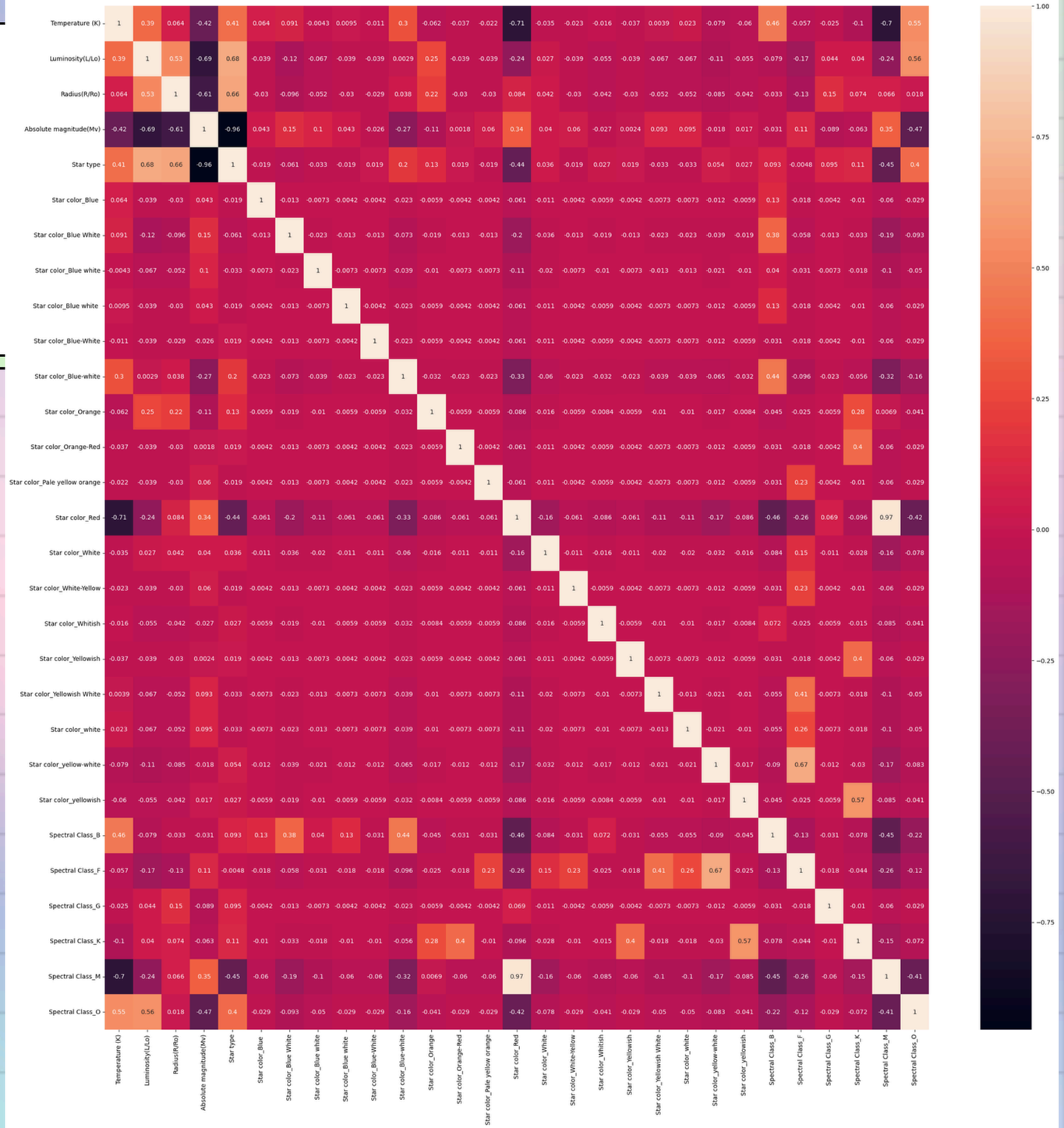
```
data = pd.get_dummies(data, drop_first=True)
data.head()
```

	Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color_Blue	Star color_Blue White	Star color_Blue white	Star color_Blue white	Star color_Blue-White	...
0	3068	0.002400	0.1700	16.12	0	False	False	False	False	False	...
1	3042	0.000500	0.1542	16.60	0	False	False	False	False	False	...
2	2600	0.000300	0.1020	18.70	0	False	False	False	False	False	...
3	2800	0.000200	0.1600	16.65	0	False	False	False	False	False	...
4	1939	0.000138	0.1030	20.06	0	False	False	False	False	False	...

5 rows × 29 columns

Correlation Matrix

```
plt.figure(figsize=(30, 30))
sns.heatmap(data.corr(), annot=True)
plt.show()
```



Normalize Data

```
scaler = MinMaxScaler()
data_rescale = pd.DataFrame(scaler.fit_transform(data.drop('Star type', axis=1)), columns=data.drop('Star type', axis=1).columns)
data_rescale.head()
```

	Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star color_Blue	Star color_Blue White	Star color_Blue white	Star color_Blue white	Star color_Blue-White	color_Bl wh
0	0.029663	2.731275e-09	0.000083	0.876798	0.0	0.0	0.0	0.0	0.0	
1	0.028980	4.944550e-10	0.000075	0.891807	0.0	0.0	0.0	0.0	0.0	
2	0.017367	2.590003e-10	0.000048	0.957473	0.0	0.0	0.0	0.0	0.0	
3	0.022622	1.412729e-10	0.000078	0.893371	0.0	0.0	0.0	0.0	0.0	
4	0.000000	6.828189e-11	0.000049	1.000000	0.0	0.0	0.0	0.0	0.0	

5 rows × 28 columns

Split data into features and target

```
X = data_rescale
y = data['Star type']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.8, random_state=42)
```

Decision Tree Classifier

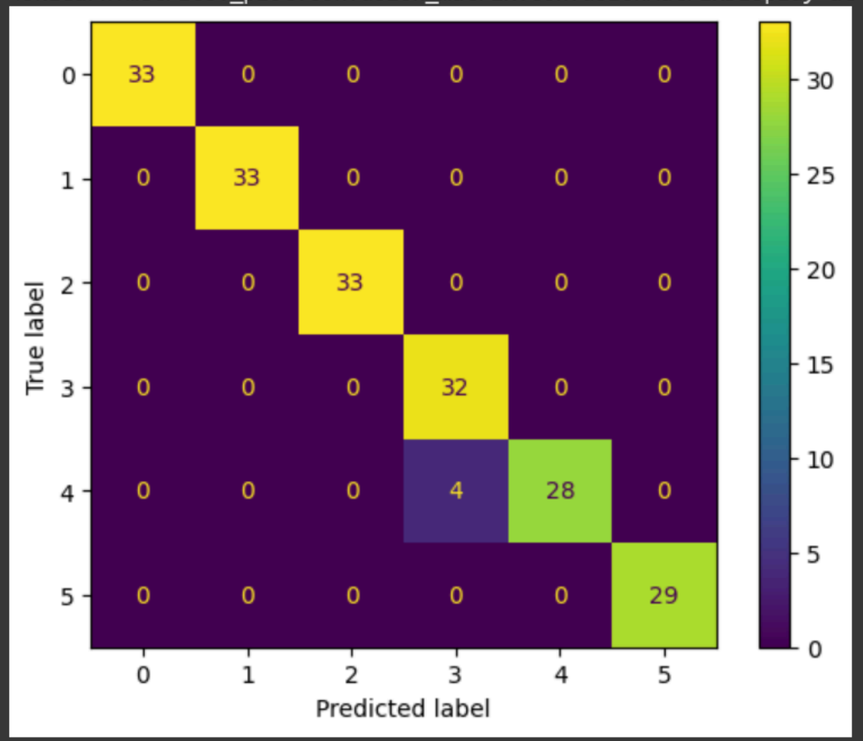
```
dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)
dtree.score(X_test, y_test)
```

```
0.96875
```

Confusion Matrix

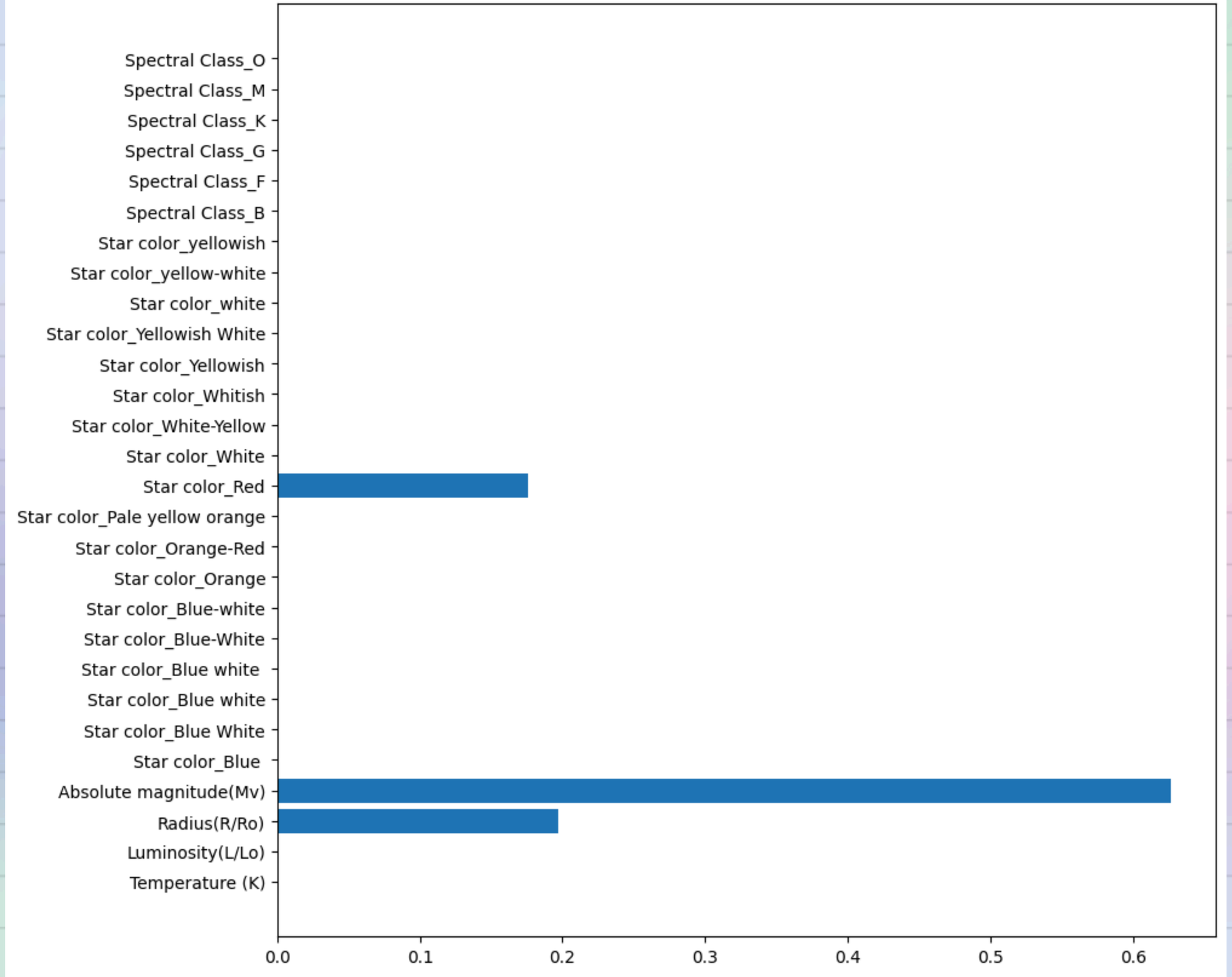
```
cfs_mat = confusion_matrix(y_test, dtree.predict(X_test))
disp = ConfusionMatrixDisplay(cfs_mat, display_labels=dtree.classes_)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x782f5a46ea10>
```



Features Weight

```
plt.figure(figsize=(10, 10))
plt.barh(X.columns, dtree.feature_importances_)
plt.show()
```


Spectral Class_O
Spectral Class_M
Spectral Class_K
Spectral Class_G
Spectral Class_F
Spectral Class_B
Star color_yellowish
Star color_yellow-white
Star color_white
Star color_Yellowish White
Star color_Yellowish
Star color_Whitish
Star color_White-Yellow
Star color_White
Star color_Red
Star color_Pale yellow orange
Star color_Orange-Red
Star color_Orange
Star color_Blue-white
Star color_Blue-White
Star color_Blue white
Star color_Blue white
Star color_Blue White
Star color_Blue
Absolute magnitude(Mv)
Radius(R/Ro)
Luminosity(L/Lo)
Temperature (K)
0.0
0.1
0.2
0.3
0.4
0.5
0.6

Remove features with high importance

```
X.keys()
```

```
Index(['Temperature (K)', 'Luminosity(L/Lo)', 'Radius(R/Ro)',
       'Absolute magnitude(Mv)', 'Star color_Blue ', 'Star color_Blue White',
       'Star color_Blue white', 'Star color_Blue white ',
       'Star color_Blue-White', 'Star color_Blue-white', 'Star color_Orange',
       'Star color_Orange-Red', 'Star color_Pale yellow orange',
       'Star color_Red', 'Star color_White', 'Star color_White-Yellow',
       'Star color_Whitish', 'Star color_Yellowish',
       'Star color_Yellowish White', 'Star color_white',
       'Star color_yellow-white', 'Star color_yellowish', 'Spectral Class_B',
       'Spectral Class_F', 'Spectral Class_G', 'Spectral Class_K',
       'Spectral Class_M', 'Spectral Class_O'],
      dtype='object')
```

```
X = data_rescale.drop(['Absolute magnitude(Mv)','Radius(R/Ro)'], axis=1)
```

Retrain

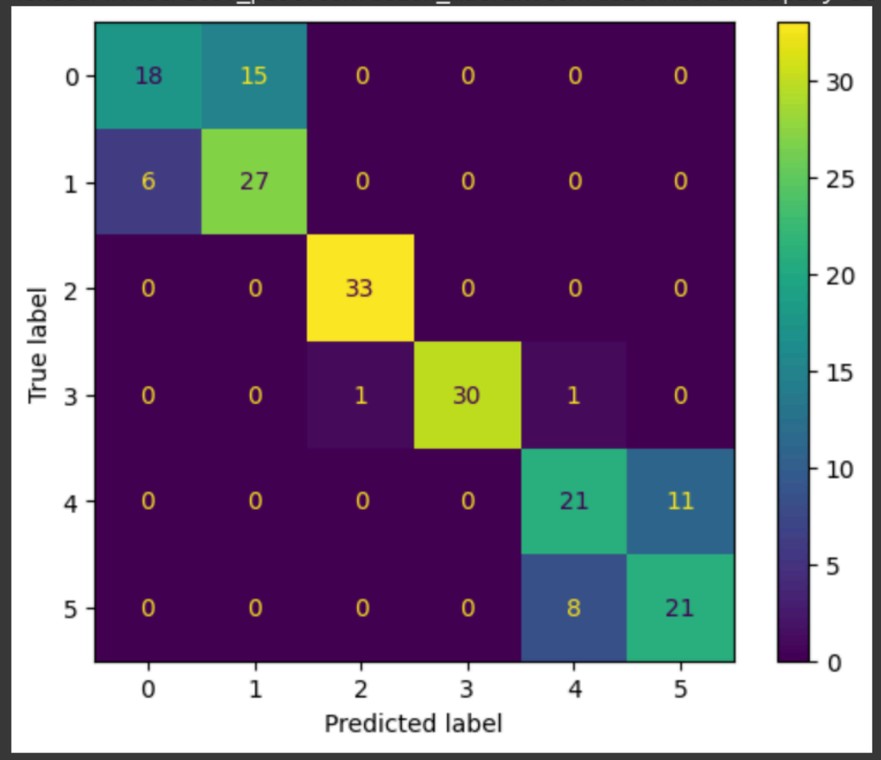
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.8, random_state=42)
dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)
dtree.score(X_test, y_test)
```

```
0.78125
```

Check confusion matrix

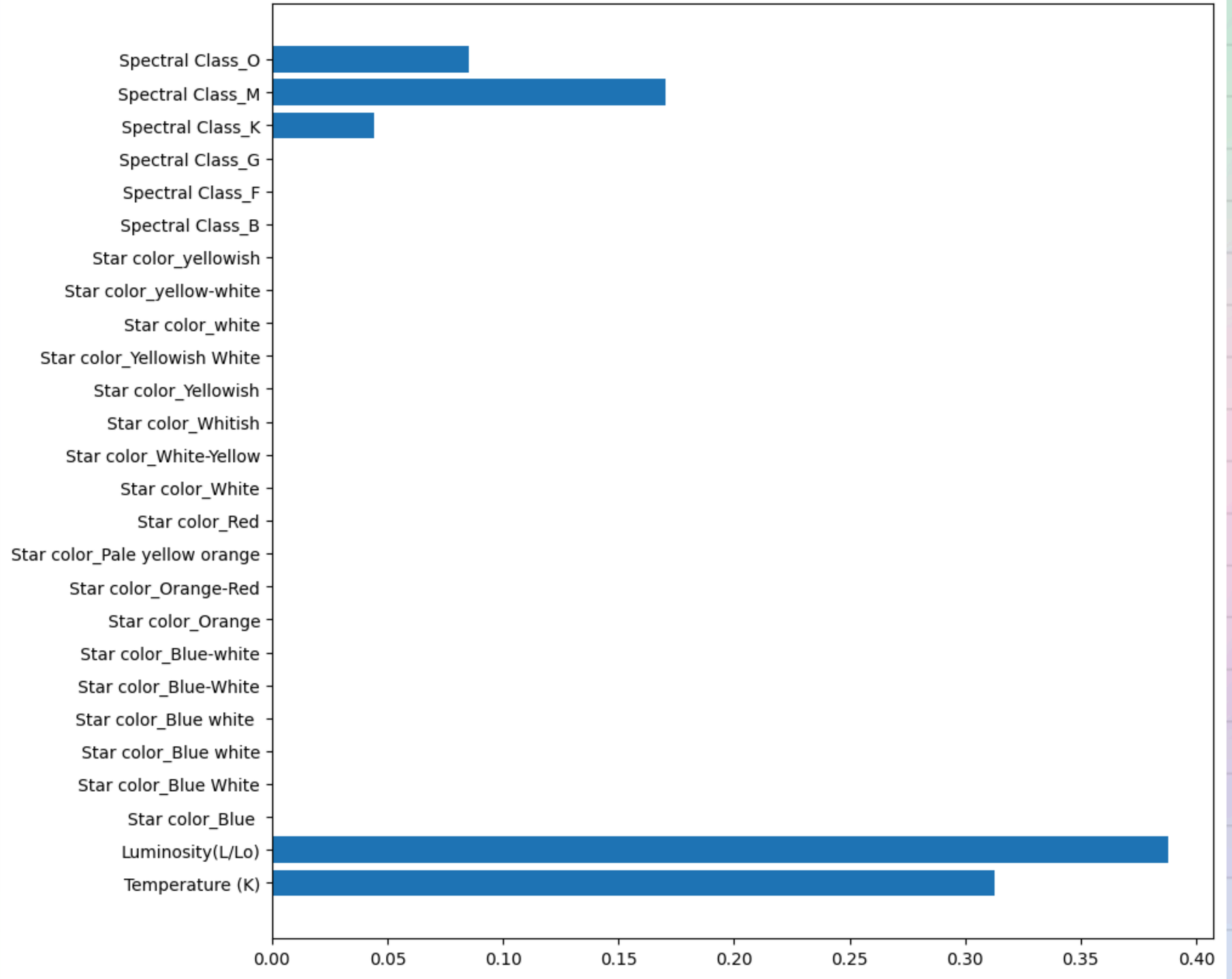
```
cfs_mat = confusion_matrix(y_test, dtree.predict(X_test))
disp = ConfusionMatrixDisplay(cfs_mat, display_labels=dtree.classes_)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7a0576b7a800>
```



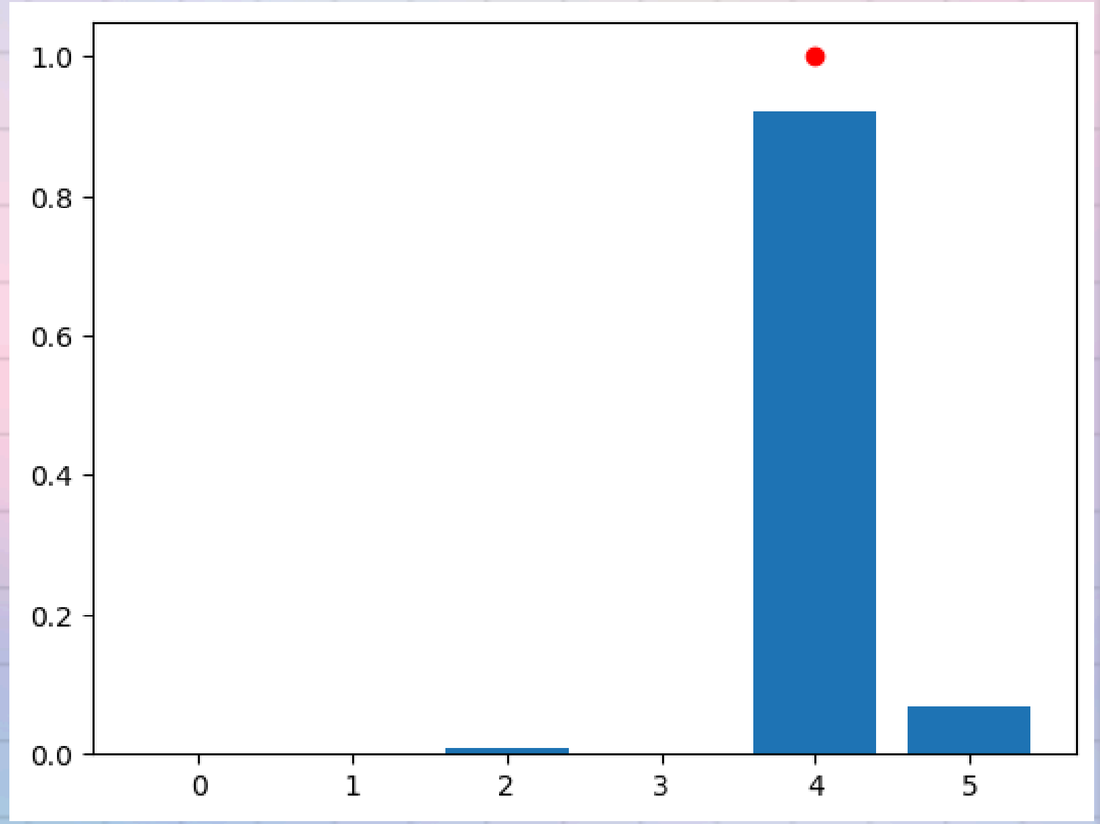
Check features weight

```
plt.figure(figsize=(10, 10))
plt.barh(X.columns, dtree.feature_importances_)
plt.show()
```


Spectral Class_O
Spectral Class_M
Spectral Class_K
Spectral Class_G
Spectral Class_F
Spectral Class_B
Star color_yellowish
Star color_yellow-white
Star color_white
Star color_Yellowish White
Star color_Yellowish
Star color_Whitish
Star color_White-Yellow
Star color_White
Star color_Red
Star color_Pale yellow orange
Star color_Orange-Red
Star color_Orange
Star color_Blue-white
Star color_Blue-White
Star color_Blue white
Star color_Blue white
Star color_Blue White
Star color_Blue
Luminosity(L/Lo)
Temperature (K)
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=200, random_state=42)
rf.fit(X_train, y_train)
print(rf.score(X_test, y_test))
x_prob = rf.predict_proba(X_test.iloc[42].values.reshape(1, -1))
plt.bar( rf.classes_,x_prob[0])
plt.scatter(y_test.iloc[42], 1, color='red', alpha=1)
```



Cross Validation Score

```python
from sklearn.model_selection import cross_val_score

scores = cross_val_score(dtree, X, y, cv=5)
mean_score = np.mean(scores)

print("Cross-Validation Scores:")
for fold, score in enumerate(scores, start=1):
    print(f"- Fold {fold}: {score}")
print(f"Mean Score: {mean_score}")
```

```
Cross-Validation Scores:
- Fold 1: 0.6666666666666666
- Fold 2: 0.7291666666666666
- Fold 3: 0.8333333333333334
- Fold 4: 0.875
- Fold 5: 0.7708333333333334
Mean Score: 0.775
```

tks 4 listenin