In [62]:
```
!matplotlib inline

import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [63]:
```
df = pd.read_csv("Stars.csv")
df.head()
```

Out[63]:

	Temperature	L	R	A_M	Color	Spectral_Class	Type
0	3068	0.002400	0.1700	16.12	Red	M	0
1	3042	0.000500	0.1542	16.60	Red	M	0
2	2600	0.000300	0.1020	18.70	Red	M	0
3	2800	0.000200	0.1600	16.65	Red	M	0
4	1939	0.000138	0.1030	20.06	Red	M	0

In [64]:
```
df.dtypes
```

Out[64]:
```
Temperature          int64
L                  float64
R                  float64
A_M                float64
Color               object
Spectral_Class      object
Type                 int64
dtype: object
```

In [65]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 240 entries, 0 to 239
Data columns (total 7 columns):
Temperature       240 non-null int64
L                 240 non-null float64
R                 240 non-null float64
A_M               240 non-null float64
Color             240 non-null object
Spectral_Class    240 non-null object
Type              240 non-null int64
dtypes: float64(3), int64(2), object(2)
memory usage: 13.2+ KB
```

In [66]:
```
df.describe()
```

Out[66]:

	Temperature	L	R	A_M	Type
count	240.000000	240.000000	240.000000	240.000000	240.000000
mean	10497.462500	107188.361635	237.157781	4.382396	2.500000
std	9552.425037	179432.244940	517.155763	10.532512	1.711394
min	1939.000000	0.000080	0.008400	-11.920000	0.000000
25%	3344.250000	0.000865	0.102750	-6.232500	1.000000
50%	5776.000000	0.070500	0.762500	8.313000	2.500000
75%	15055.500000	198050.000000	42.750000	13.697500	4.000000
max	40000.000000	849420.000000	1948.500000	20.060000	5.000000

In [67]:
```
df["Type"].mean()
```

Out[67]:
```
2.5
```

In [68]:
```
print(df.isna().sum())
print(df.isna().sum(axis=0))
```

```
Temperature       0
L                 0
R                 0
A_M               0
Color             0
Spectral_Class    0
Type              0
dtype: int64
Temperature       0
L                 0
R                 0
A_M               0
Color             0
Spectral_Class    0
Type              0
dtype: int64
```

In [69]:
```
df.skew()
```

Out[69]:
```
Temperature    1.321568
L              2.068069
R              1.946800
A_M           -0.121540
Type           0.000000
dtype: float64
```

In [70]:
```
df.kurtosis()
```

Out[70]:
```
Temperature    0.877352
L              4.465098
R              2.072935
A_M           -1.659888
Type          -1.269979
dtype: float64
```

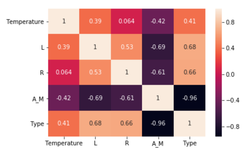
In [71]:
```
corr = df.corr()
corr.style.background_gradient(cmap='coolwarm')
```

Out[71]:

	Temperature	L	R	A_M	Type
Temperature	1	0.393404	0.064216	-0.420261	0.411129
L	0.393404	1	0.526516	-0.692619	0.676845
R	0.064216	0.526516	1	-0.608728	0.660975
A_M	-0.420261	-0.692619	-0.608728	1	-0.955276
Type	0.411129	0.676845	0.660975	-0.955276	1

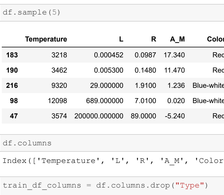
In [72]:
```
import seaborn as sns
sns.heatmap(df.corr(), annot=True)
```

Out[72]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x23462061048>
```



In [73]:
```
temp_cols_list = ["A_M", "Type"]
temp_df_1 = df[temp_cols_list]
```

In [74]:
```
plt.scatter(temp_df_1["A_M"], temp_df_1["Type"], color = "#ff0000")
plt.xlabel("A_M")
plt.ylabel("Type")
plt.title("Scatter plot")
plt.show()
```



In [75]:
```
df.sample(5)
```

Out[75]:

	Temperature	L	R	A_M	Color	Spectral_Class	Type
183	3218	0.000452	0.0987	17.340	Red	M	0
190	3462	0.005300	0.1480	11.470	Red	M	1
216	9320	29.000000	1.9100	1.236	Blue-white	A	3
98	12098	689.000000	7.0100	0.020	Blue-white	A	3
47	3574	200000.000000	89.0000	-5.240	Red	M	4

In [76]:
```
df.columns
```

Out[76]:
```
Index(['Temperature', 'L', 'R', 'A_M', 'Color', 'Spectral_Class', 'Type'], dtype='object')
```

In [77]:
```
train_df_columns = df.columns.drop("Type")

print(train_df_columns)
```

```
Index(['Temperature', 'L', 'R', 'A_M', 'Color', 'Spectral_Class'], dtype='object')
```

In [78]:
```
train_df = df[train_df_columns]
test_df = df["Type"]
```

In [79]:
```
def stringToInteger(list_of_vals) -> dict :
    d = {}
    idx = 0
    for i in list_of_vals:
        if i not in d:
            d[i] = idx
            idx += 1
    return d
```

In [80]:
```
colour_dict = stringToInteger(train_df['Color'])
train_df['colour_modified'] = train_df['Color'].apply(lambda x : colour_dict[x])

spectral_class_dict = stringToInteger(train_df['Spectral_Class'])
train_df['spectral_class_modified'] = train_df['Spectral_Class'].apply(lambda x: spectral_class_dict[x])
train_df.drop(["Color","Spectral_Class"],axis=1,inplace=True)
train_df.sample(5)
```

Out[80]:

	Temperature	L	R	A_M	colour_modified	spectral_class_modified
5	2840	0.00065	0.11000	16.98	0	0
140	13420	0.00059	0.00981	13.67	1	1
142	18290	0.00130	0.00934	12.78	6	1
7	2600	0.00040	0.09600	17.40	0	0
48	3625	184000.00000	84.00000	-6.74	0	0

In [81]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(train_df, test_df, test_size = 0.20, random_state=42)
```

In [82]:
```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(random_state=42, max_iter= 2000, verbose=False).fit(X_train,Y_train)
accr = clf.score(X_test,Y_test)
print(f"we have an accuracy of {accr} \n")
```

```
we have an accuracy of 0.8958333333333334
```

In [83]:
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
```

In [84]:
```
k=5
kf =KFold(n_splits=k, random_state=None)
result = cross_val_score(clf, X_train, Y_train, cv =kf)
print("Avg accuracy : {}".format(result.mean()))
```

```
Avg accuracy : 0.8901484480431849
```