```
In [132…
from nltk.tokenize import sent_tokenize
import numpy as np
import pandas as pd
from pandas import read_csv
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestRegressor
import seaborn as sns
import math
from sklearn.cluster import KMeans
import statistics as stat
from sklearn.neural_network import MLPRegressor
from sklearn.decomposition import PCA
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.compose import make_column_transformer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
import string
```

EJERCICIO 1

- Coje un conjunto de datos y aplica un PipeLine y gridsearch sobre un Random Forest

Utilizaremos un conjunto de datos usado por varias universidades para el estudio de las estrellas mediante el diagrama de Hertzsprung-Russell. El Dataset consta de unas 240 estrellas

```
In [33]:
df= pd.read_csv("stars.csv" )

df.head()
```

Out[33]:

	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

Miramos las dimensiones y columnas

```
In [34]:
df.shape
```

```
Out[34]:
(240, 7)
```

Las columnas de del Dataset son:

Temperatura absoluta(en K): Es la temperatura absoluta de la superficie de la estrella

Luminosidad relativa(L/Lo): Es el brillo relativo de la estrella, donde Lo es el brillo del objeto más brillante del cielo, el Sol

Radio relativo(R/Ro) es el radio relativo de la estrella

Magnitud absoluta(Mv) es el brillo que se vería a 10 Parsecs de distancias

Color de la estrella (Blanco, Rojo, Azul, amarillo, Amarillo-naranja, etc), que depende de la temperatura de la superficie

Clase espectral(O,B,A,F,G,K,,M)

Tipo de estrella(Red Dwarf, Brown Dwarf, White Dwarf, Main Sequence , SuperGiants, HyperGiants)

- **Magnitud aparente y absoluta (brillo)**

Considere dos fuentes de luz: una linterna en la mano y una farola distante a cierto punto de distancia. Superficialmente, uno llegaría a la conclusión de que la linterna que tenemos en la mano es más brillante que la farola lejana. En esencia, este problema se traduce en brillo estelar. Una mirada casual a las estrellas no ayuda a revelar si la estrella es una brasa brillante cercana o un gran faro distante. Para distinguir entre qué tan brillante se ve una estrella y qué tan brillante es realmente, los astrónomos utilizan la medida de magnitud aparente frente a absoluta.

Como se entiende por el nombre, el brillo aparente es el brillo relativo de una estrella percibido por un observador en la Tierra, mientras que el brillo absoluto es la percepción del mismo observador si todas las estrellas estuvieran colocadas mágicamente a la misma distancia estándar, 10 Parsecs.

Una forma de medir el brillo aparente es en términos de unidades de fotos por segundo. Una forma más conveniente es expresar esto como una proporción de una estrella brillante prominente con la estrella real. Sirius, la estrella más brillante, tiene una magnitud aparente de -1,46, mientras que las estrellas más débiles visibles a simple vista tienen magnitudes de alrededor de 6.

El brillo aparente de una estrella, en su mayor parte, depende de dos factores:

1. Brillo real

2. Distancia desde la Tierra

Por otro lado, la magnitud absoluta refleja la verdadera cantidad de luz emitida por la estrella. Por lo tanto, expresamos este valor como la luminosidad de la estrella.

- **Clase espectral y temperaturas superficiales**

Henry Draper fue el primero en fotografiar espectros estelares en el año 1872. Las clases espectrales fueron clasificadad por la temperatura, y les pusieron designaciones de letras en una secuencia que iba de caliente a fría en el orden: O, B, A, F, G, K y M. Las estrellas de tipo O tienen las temperaturas superficiales más altas que pueden existir. tan calientes como 30.000 Kelvins y las estrellas Tipo M pueden ser tan frías como 3.000 Kelvins.

Diagrama de Hertzsprung-Russell

De la Ley de Stefan-Boltzmann, sabemos que:

formula

Al igual que los cuerpos negros, las estrellas también emiten radiaciones. Por lo tanto, podemos hacer uso de las tres propiedades anteriores (luminosidad, radio y temperatura) para comparar y categorizar estrellas. Ejnar Hertzsprung y Henry Russells trazaron el ahora famoso Diagrama HR que se convirtió en una herramienta fundamental de la astronomía moderna. Trazando la clasificación espectral contra la magnitud absoluta, encontraron que la mayoría de las estrellas se encuentran en ciertas regiones del diagrama.

diagrama

diagrama

Las estrellas se situan en cuatro zonas: Enanas Blancas, secuencia principa, Gigantes y supergigantes.

Secuencia principal La estrecha banda de estrellas que va desde la esquina superior izquierda hasta la esquina inferior izquierda forma la secuencia principal. Estas estrellas reflejan las reacciones de fusión dentro del núcleo del Sol. Las estrellas más grandes de esta secuencia (alta luminosidad, alta temperatura) experimentan reacciones de fusión rápidas. Por lo tanto, cuanto mayor sea la posición de la estrella en esta secuencia, menor será su vida. Y más baja la posición, más larga la vida.

Gigantes El cúmulo de estrellas ubicado sobre la Secuencia Principal en la esquina superior derecha se denomina Gigantes. Estas estrellas aparecen en su mayoría de color rojo debido a su gran radio y bajas temperaturas superficiales. Habiendo quemado todo el hidrógeno en helio, estas estrellas han entrado en la etapa final de su vida.

supergigantes Con radios mayores que los de las Gigantes Rojas, las Supergigantes se encuentran dispersas sobre la Secuencia Principal y las estrellas Gigantes. El proceso de fusión de helio-carbono ocurre mucho más rápido en estas estrellas debido a sus masas y, por lo tanto, nunca se vuelven Gigantes. Los supergigantes pueden fusionar carbono en oxígeno, oxígeno en neón, neón en magnesio, magnesio en silicio y, finalmente, silicio en hierro. Con un núcleo de hierro, el supergigante finalmente morirá en una explosión de supernova. Dependiendo de la masa original de la estrella, el resultado sería una estrella de neutrones o un agujero negro.

Enanas Blancas Las pocas estrellas que se encuentran en la esquina inferior izquierda de las estrellas de la Secuencia Principal. La reacción de fusión de estas estrellas se detiene con el carbono y el núcleo finalmente comienza a encogerse. La capa exterior se disipa y todo lo que queda es el pequeño núcleo frío de la estrella. Las enanas blancas tienen temperaturas que dan como resultado colores de blanco amarillento a blanco azulado.

Enanas marrones y Rojas pertenecen a la secuencia principal.

en la misma información de donde saco el archivo, podemos ver la siguiente información :

Brown Dwarf -> Star Type = 0

Red Dwarf -> Star Type = 1

White Dwarf-> Star Type = 2

Main Sequence -> Star Type = 3

Supergiant -> Star Type = 4

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 x 10^26 Watts Ro = 6.9551 x 10^8 m

Vamos a intentar predecir el tipo de estrella a través del data set mediante una Random Forest Classifier.

Vamos a mirar la cantidad de valores nulos la varianza explicada, y algunos datos más

```
In [36]:
df.isna().sum()
```

```
Out[36]:
Temperature (K)           0
Luminosity(L/Lo)          0
Radius(R/Ro)              0
Absolute magnitude(Mv)    0
Star type                 0
Star color                0
Spectral Class            0
dtype: int64
```

```
In [39]:
# Miramos las clases que nos encontramos en cada variable categórica
df["Star type"].value_counts()
```

```
Out[39]:
0    40
1    40
2    40
3    40
4    40
5    40
Name: Star type, dtype: int64
```

```
In [40]:
df["Star color"].value_counts()
```

```
Out[40]:
Red                   112
Blue                   55
Blue-white             26
Blue White             10
yellow-white            8
White                   7
Blue white              3
Yellowish White         3
white                   3
Whitish                 2
Orange                  2
yellowish               2
Pale yellow orange      1
White-Yellow            1
Blue                    1
Yellowish               1
Orange-Red              1
Blue white              1
Blue-White              1
Name: Star color, dtype: int64
```

```
In [41]:
df["Spectral Class"].value_counts()
```

```
Out[41]:
M    111
B     46
O     40
A     19
F     17
K      6
G      1
Name: Spectral Class, dtype: int64
```

podemos ver que Spectral class y Star Color están completamente desbalanceados

RandomForestClassifier admite tanto variables categoricas como numércias, lo que nos viene bien, y sin necesidad de Dummies o OneHotEncoder. Tenemos muy poco datos, solo 240. Así que voy a interntar suplir la baja cantidad de datos con muchos estimadores

-Vamos a ver el VIF

```
In [42]:
df.columns
```

```
Out[42]:
Index(['Temperature (K)', 'Luminosity(L/Lo)', 'Radius(R/Ro)',
       'Absolute magnitude(Mv)', 'Star type', 'Star color', 'Spectral Class'],
      dtype='object')
```

```
In [44]:
variables =['Temperature (K)', 'Luminosity(L/Lo)', 'Radius(R/Ro)',
       'Absolute magnitude(Mv)']
def vif(X):
    vifDF = pd.DataFrame()
    vifDF["variables"] = X.columns
    vifDF["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
    return vifDF
Xvif=df.drop(["Star type", 'Star color', 'Spectral Class'], axis=1)
round(vif(Xvif),2)
```

Out[44]:

	variables	VIF
0	Temperature (K)	1.76
1	Luminosity(L/Lo)	2.56
2	Radius(R/Ro)	1.70
3	Absolute magnitude(Mv)	1.31

Podemos ver unos resultado de VIF muy buenos, con variables bastante independientes Vamos a empezar a trabajar con los diferentes parámetros.

```
In [75]:
scaler= StandardScaler()# el eslcalado de variables numéricas
rfc = RandomForestClassifier()# el Bosque aleatorio
column_transform = make_column_transformer((scaler,['Temperature (K)', 'Luminosity(L/Lo)', 'Radius(R/Ro)',
       'Absolute magnitude(Mv)'] ))
pasos = [ ("transformar ",column_transform), ("RFC", rfc) ]# los pasos para el PipeLine
pip= Pipeline(pasos)
paramametros = {
    'RFC__n_estimators': [400, 800, 1200],
    'RFC__max_features': ['auto', 'sqrt', 'log2'],
    'RFC__max_depth' : [4,6,8,10,12],
    'RFC__criterion' :['gini', 'entropy']}# los parámetros del GridSearchCV, básicamente del Bosque Aleatorio
```

```
In [82]:
grid = GridSearchCV(estimator = pip,
                    param_grid =paramametros ,
                    cv = 5, scoring="accuracy")# El Grid
```

Antes de todo vamos a separar el pequeño DataSet en una parte para entrenar y otra para testear.

```
In [83]:
X= df.drop(["Star type"], axis=1 )
y= df["Star type"]
Xtrain, Xtest, ytrain, ytest = train_test_split( X, y, test_size=0.30, random_state=42)
```

```
In [84]:
grid.fit(Xtrain, ytrain)
```

```
Out[84]:
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('transformar ',
                                        ColumnTransformer(transformers=[('standardscaler',
                                                                         StandardScaler(),
                                                                         ['Temperature '
                                                                          '(K)',
                                                                          'Luminosity(L/Lo)',
                                                                          'Radius(R/Ro)',
                                                                          'Absolute '
                                                                          'magnitude(Mv)'])])),
                                       ('RFC', RandomForestClassifier())]),
             param_grid={'RFC__criterion': ['gini', 'entropy'],
                         'RFC__max_depth': [4, 6, 8, 10, 12],
                         'RFC__max_features': ['auto', 'sqrt', 'log2'],
                         'RFC__n_estimators': [400, 800, 1200]},
             scoring='accuracy')
```

Pasamos a mirar los mejores parámetros y a estimar el test con el mejor parámetro

```
In [86]:
grid.best_params_
```

```
Out[86]:
{'RFC__criterion': 'gini',
 'RFC__max_depth': 12,
 'RFC__max_features': 'sqrt',
 'RFC__n_estimators': 400}
```

```
In [87]:
rfcgrid= grid.best_estimator_
fx= rfcgrid.predict(Xtest)
```

```
In [94]:
from sklearn.metrics import confusion_matrix
cmrfc= confusion_matrix(ytest, fx)

cmrfc
```

```
Out[94]:
array([[14,  0,  0,  0,  0,  0],
       [ 0, 12,  0,  0,  0,  0],
       [ 0,  0, 10,  0,  0,  0],
       [ 0,  0,  0, 10,  0,  0],
       [ 0,  0,  0,  0, 11,  0],
       [ 0,  0,  0,  0,  0, 15]], dtype=int64)
```

Podemos ver que los ha predicho todos correctamente.

```
In [128…
import nltk
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from nltk.corpus import stopwords
nltk.download("punkt")
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\walte\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\walte\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
Out[128…
True
```

Ejercicio 2. Coge un texto en inglés y calcula la frecuencia de palabras.

```
In [137…
from io import open
import re
type(string.punctuation)
```

```
Out[137…
str
```

```
In [141…
fichero = open ( "Manifesto.txt", "r",  encoding="utf8")# importamos el archivo

texto= fichero.read()

lib= re.sub(r'[^\w\s]','',texto) # quitamos los signos de puntuación
libro = lib.lower()# pasamos todas las palabras a minúsculas para evita que palabras sin valor como The, In, Ar
```

```
In [142…
tokens = word_tokenize(libro, language='english', preserve_line=False) # creamos el split de palabras
palabras_vacias=set(stopwords.words("english")) # quitamos las palabras sin contenido
filtro=[]
for w in tokens:
    if w not in palabras_vacias:
        filtro.append(w)

frecuencia= FreqDist(filtro)
```

```
In [144…
frecuencia.plot(30,cumulative=False)
plt.show()
```