Predecir cuantos juegos puede ganar en una temporada un equipo específico de béisbol de las Ligas Mayores de Béisbol

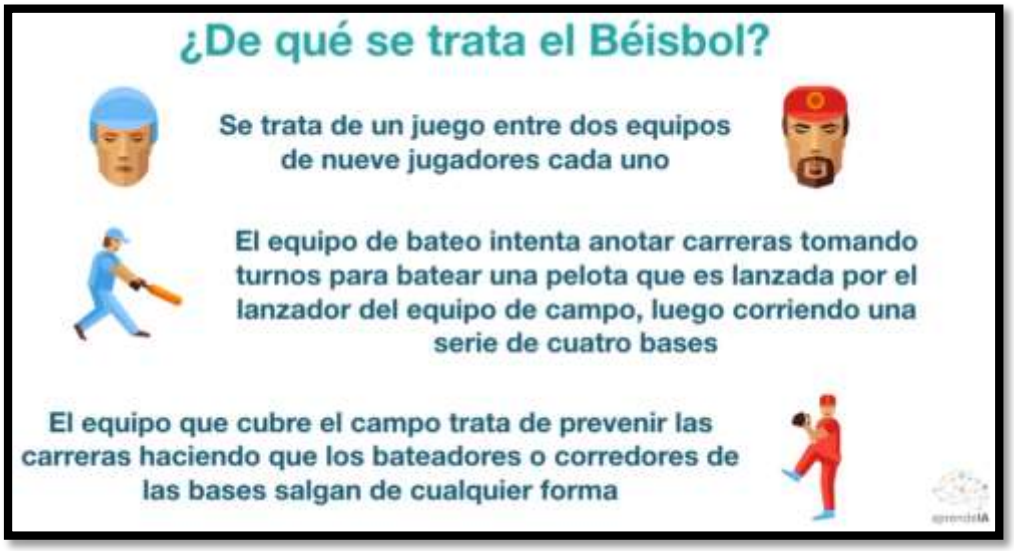
Enlace github: https://github.com/dvgr78/PROYECTOS-MODELOS-MACHINE-LEARNING/tree/main/MODELO%20PARTIDOS%20BASEBALL

email: dvgr78@hotmail.com

Contenido

TÍTULO

Predecir cuantos juegos puede ganar en una temporada un equipo específico de béisbol de las Ligas Mayores de Béisbol



Importamos librerías

```
# Importamos librerías
import pandas as pd
import matplotlib.pyplot as plt
```

Análisis de los datos

```
# Importamos los datos contenidos en Team.csv
data = pd.read_csv('Teams.csv')
print(data)
```

	Agno	Liga	Equipo	Franquicia	...	Factor_Pitcher	Id_RW	Id_BRW	Id_R
0	1871	NaN	BS1	BNA	...	98	BOS	BS1	BS1
1	1871	NaN	CH1	CNA	...	102	CHI	CH1	CH1
2	1871	NaN	CL1	CFC	...	100	CLE	CL1	CL1
3	1871	NaN	FW1	KEK	...	107	KEK	FW1	FW1
4	1871	NaN	NY2	NNA	...	88	NYU	NY2	NY2
...	...	...	...	...	...	...	...	...	...
2920	2019	NL	SLN	STL	...	97	STL	SLN	SLN
2921	2019	AL	TBA	TBD	...	96	TBR	TBA	TBA
2922	2019	AL	TEX	TEX	...	112	TEX	TEX	TEX
2923	2019	AL	TOR	TOR	...	98	TOR	TOR	TOR
2924	2019	NL	WAS	WSN	...	104	WSN	MON	WAS

[2925 rows x 48 columns]

```
# Analizamos los datos
print(data.shape)
```

```
(2925, 48)
```

```
# Formato de los datos
print(data.dtypes)
```

```
Agno                    int64
Liga                   object
Equipo                 object
Franquicia             object
Division               object
Clasificacion           int64
P_Jugados               int64
Local                 float64
Ganados                 int64
Perdidos                int64
Gan_Division           object
Gan_WC                 object
Gan_Liga               object
Gan_WS                 object
Carreras_Ganadas        int64
Bateos                  int64
Golpes_Bateador         int64
Dobles                  int64
Triples                 int64
Home_Runs               int64
Carreras              float64
```

```
Strike_Fuera                 float64
Bases_Robadas                float64
Eliminados                   float64
Eliminados_Pitcher           float64
Sacrificados                 float64
Carreras_Oponentes             int64
Carreras_Perdidas              int64
Avg_CP                       float64
Juegos                         int64
Cierres                        int64
Salvados                       int64
Strikex3                       int64
Bateos_Recibidos               int64
Home_Runs_Recibidos            int64
Carreras_Recibidas             int64
Eliminados_Pitcher_Recibidos   int64
Errores                        int64
Doble_Juego                    int64
Porcentaje_Campo             float64
Nombre_Equipo                 object
Nombre_Estadio                object
```

```
Asistencia_Estadio    float64
Factor_Bateadores       int64
Factor_Pitcher          int64
Id_RW                  object
Id_BRW                 object
Id_R                   object
dtype: object
```

```
# Buscamos datos nulos o faltantes
print(data.isnull().sum())
```

```
Agno                    0
Liga                   50
Equipo                  0
Franquicia              0
Division             1517
Clasificacion           0
P_Jugados               0
Local                 399
Ganados                 0
Perdidos                0
Gan_Division         1545
Gan_WC               2181
Gan_Liga               28
Gan_WS                357
Carreras_Ganadas        0
Bateos                  0
Golpes_Bateador         0
Dobles                  0
Triples                 0
Home_Runs               0
Carreras                1
```

```
Strike_Fuera                   16
Bases_Robadas                 126
Eliminados                    832
Eliminados_Pitcher           1158
Sacrificados                 1541
Carreras_Oponentes              0
Carreras_Perdidas               0
Avg_CP                          0
Juegos                          0
Cierres                         0
Salvados                        0
Strikex3                        0
Bateos_Recibidos                0
Home_Runs_Recibidos             0
Carreras_Recibidas              0
Eliminados_Pitcher_Recibidos    0
Errores                         0
Doble_Juego                     0
Porcentaje_Campo                0
Nombre_Equipo                   0
Nombre_Estadio                 34
```

```
Asistencia_Estadio    279
Factor_Bateadores       0
Factor_Pitcher          0
Id_RW                   0
Id_BRW                  0
Id_R                    0
dtype: int64
```

Procesamiento de los datos

```
# Eliminación de columnas innecesarias
borrar_columnas = ['Liga','Franquicia','Clasificacion','Local',
                   'Gan_Division','Gan_WC','Gan_Liga','Gan_WS',
                   'Sacrificados','Nombre_Equipo','Nombre_Estadio',
                   'Asistencia_Estadio','Factor_Bateadores',
                   'Factor_Pitcher','Id_R','Id_BRW','Id_RW']
data = data.drop(borrar_columnas, axis=1)
data = data.drop(['Eliminados','Eliminados_Pitcher','Division'], axis=1)
# Completamos los datos nulos con la media de cada uno
data['Carreras']        = data['Carreras'].fillna(data['Carreras'].median())
data['Strike_Fuera']    =
data['Strike_Fuera'].fillna(data['Strike_Fuera'].median())
data['Bases_Robadas']   =
data['Bases_Robadas'].fillna(data['Bases_Robadas'].median())
print(data.shape)
print(data.dtypes)
print(data.isnull().sum())
```

```
(2925, 28)
```

Agno	int64
Equipo	object
P_Jugados	int64
Ganados	int64
Perdidos	int64
Carreras_Ganadas	int64
Bateos	int64
Golpes_Bateador	int64
Dobles	int64
Triples	int64
Home_Runs	int64
Carreras	float64
Strike_Fuera	float64
Bases_Robadas	float64

Carreras_Oponentes	int64
Carreras_Perdidas	int64
Avg_CP	float64
Juegos	int64
Cierres	int64
Salvados	int64
Strikex3	int64
Bateos_Recibidos	int64
Home_Runs_Recibidos	int64
Carreras_Recibidas	int64
Eliminados_Pitcher_Recibidos	int64
Errores	int64
Doble_Juego	int64
Porcentaje_Campo	float64

Agno	0
Equipo	0
P_Jugados	0
Ganados	0
Perdidos	0
Carreras_Ganadas	0
Bateos	0
Golpes_Bateador	0
Dobles	0
Triples	0
Home_Runs	0
Carreras	0
Strike_Fuera	0
Bases_Robadas	0

Carreras_Oponentes	0
Carreras_Perdidas	0
Avg_CP	0
Juegos	0
Cierres	0
Salvados	0
Strikex3	0
Bateos_Recibidos	0
Home_Runs_Recibidos	0
Carreras_Recibidas	0
Eliminados_Pitcher_Recibidos	0
Errores	0
Doble_Juego	0
Porcentaje_Campo	0

Segmentación de los datos

Por eras o etapas:

Se dividirá en 8 eras en total:

Menor a 1920 = Era 1
Entre 1920 y 1941 = Era 2
Entre 1942 y 1945 = Era 3
Entre 1946 y 1962 = Era 4
Entre 1963 y 1976 = Era 5
Entre 1977 y 1992 = Era 6
Entre 1993 y 2009 = Era 7
Mayor a 2010 = Era 8

```
# Separamos por eras o etapas
i =1
for year in data['Agno']:
    if year < 1920:
        data.loc[i,"era"]=1
    elif year >= 1920 and year <= 1941:
        data.loc[i, "era"] = 2
    elif year >= 1942 and year <= 1945:
        data.loc[i, "era"] = 3
    elif year >= 1946 and year <= 1962:
        data.loc[i, "era"] = 4
    elif year >= 1963 and year <= 1976:
        data.loc[i, "era"] = 5
    elif year >= 1977 and year <= 1992:
        data.loc[i, "era"] = 6
    elif year >= 1993 and year <= 2009:
        data.loc[i, "era"] = 7
    elif year >= 2010:
        data.loc[i, "era"] = 8
    i += 1
```

Por décadas:

Se dividirá en décadas:

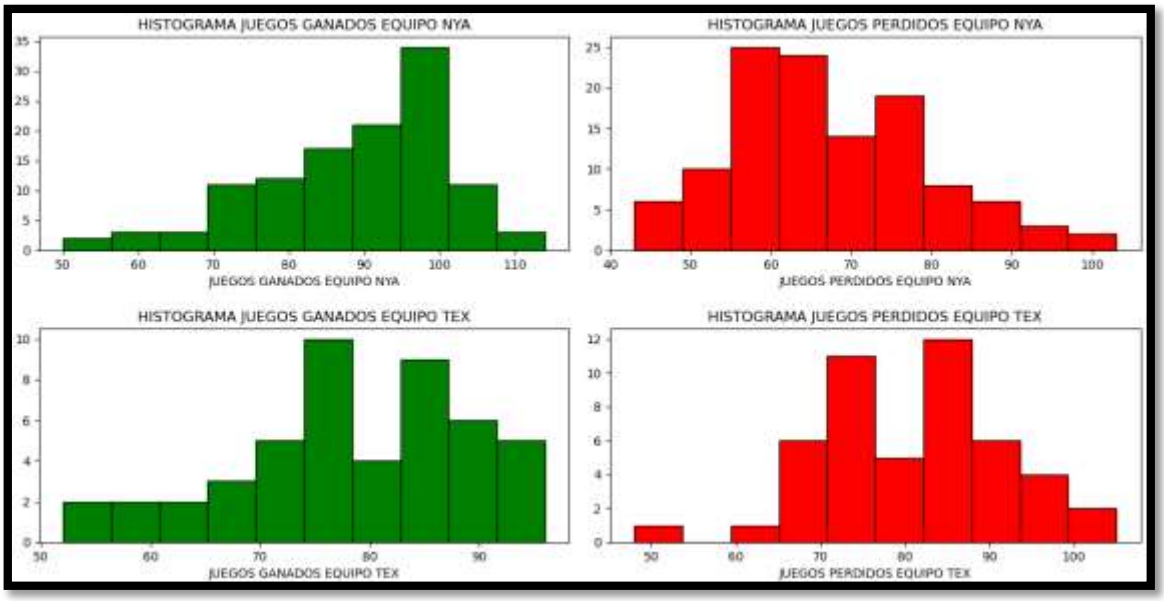
Menor a 1920 = 1910
Entre 1920 y 1929 = 1920
Entre 1930 y 1939 = 1930
Entre 1940 y 1949 = 1940
Entre 1950 y 1959 = 1950
Entre 1960 y 1969 = 1960
Entre 1970 y 1979 = 1970
Entre 1980 y 1989 = 1980
Entre 1990 y 1999 = 1990
Entre 2000 y 2009 = 2000
Mayor a 2010 = 2010

```
# Separamos por décadas
j =1
for year in data['Agno']:
    if year     <   1920:
        data.loc[j,"decada"]    =   1910

    elif year   >=  1920 and year   <=  1929:
        data.loc[j, "decada"]   =   1920

    elif year   >=  1930 and year   <=  1939:
        data.loc[j, "decada"]   =   1930

    elif year   >=  1940 and year   <=  1949:
        data.loc[j, "decada"]   =   1940

    elif year   >=  1950 and year   <=  1959:
        data.loc[j, "decada"]   =   1950

    elif year   >=  1960 and year   <=  1969:
        data.loc[j, "decada"]   =   1960

    elif year   >=  1970 and year   <=  1979:
        data.loc[j, "decada"]   =   1970

    elif year   >=  1980 and year   <=  1989:
        data.loc[j, "decada"]   =   1980

    elif year   >=  1990 and year   <=  1999:
        data.loc[j, "decada"]   =   1990

    elif year   >=  2000 and year   <=  2009:
        data.loc[j, "decada"]   =   2000

    elif year   >=  2010:
        data.loc[j, "decada"]   =   2010

    j += 1
```

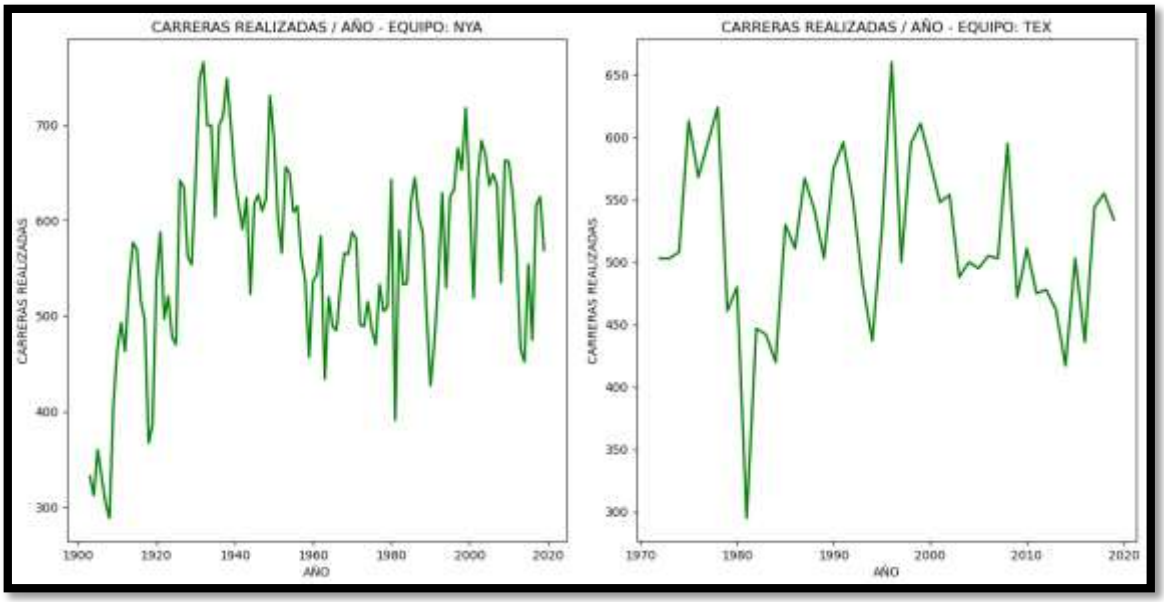
Visualización gráfica

```
# Graficamos datos
fig = plt.figure(figsize=(12,6))
ax1 = fig.add_subplot(2,2,1)
ax2 = fig.add_subplot(2,2,2)
ax3 = fig.add_subplot(2,2,3)
ax4 = fig.add_subplot(2,2,4)

ax1.hist(data_equipo['Ganados'],bins=10, color='green',linewidth=1,
edgecolor="black")
ax1.set_xlabel("JUEGOS GANADOS EQUIPO {}".format(equipo))
ax1.set_title("HISTOGRAMA JUEGOS GANADOS EQUIPO {}".format(equipo))

ax2.hist(data_equipo['Perdidos'],bins=10, color='red', linewidth=1,
edgecolor="black")
ax2.set_xlabel("JUEGOS PERDIDOS EQUIPO {}".format(equipo))
ax2.set_title("HISTOGRAMA JUEGOS PERDIDOS EQUIPO {}".format(equipo))

ax3.hist(data_equipo1['Ganados'],bins=10, color='green', linewidth=1,
edgecolor="black")
ax3.set_xlabel("JUEGOS GANADOS EQUIPO {}".format(equipo1))
ax3.set_title("HISTOGRAMA JUEGOS GANADOS EQUIPO {}".format(equipo1))
```

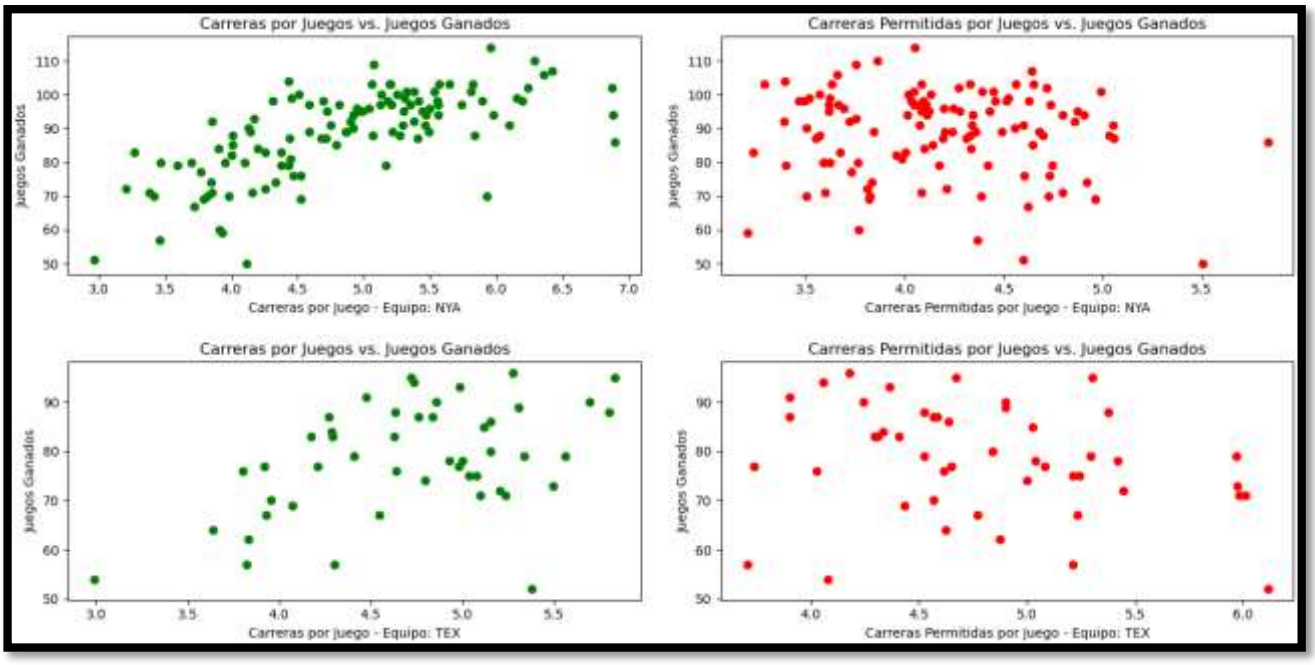
```
ax4.hist(data_equipo1['Perdidos'],bins=10, color='red',linewidth=1,
edgecolor="black")
ax4.set_xlabel("JUEGOS PERDIDOS EQUIPO {}".format(equipo1))
ax4.set_title("HISTOGRAMA JUEGOS PERDIDOS EQUIPO {}".format(equipo1))
plt.tight_layout()
plt.show()
```




```
# Graficamos las carreras realizadas
fig = plt.figure(figsize=(12,6))
ax1 = fig.add_subplot(1,2,1)
ax2 = fig.add_subplot(1,2,2)
ax1.plot(data_equipo['Agno'],data_equipo['Carreras'],linewidth=2.0,color="green")
ax1.set_title("CARRERAS REALIZADAS / AÑO - EQUIPO: {}".format(equipo))
ax1.set_ylabel("CARRERAS REALIZADAS")
ax1.set_xlabel("AÑO")
ax2.plot(data_equipo1['Agno'],data_equipo1['Carreras'],linewidth=2.0,color="green")
ax2.set_title("CARRERAS REALIZADAS / AÑO - EQUIPO: {}".format(equipo1))
ax2.set_xlabel("AÑO")
ax2.set_ylabel("CARRERAS REALIZADAS")
plt.tight_layout()
plt.show()
```



```
# Gráficas adicionales
CarrerasG_x_Partido  = data_equipo['Carreras_Ganadas'] / data_equipo['P_Jugados']
CarrerasG_x_Partido1 = data_equipo1['Carreras_Ganadas'] / data_equipo1['P_Jugados']
CarrerasP_x_Partido = data_equipo['Carreras_Oponentes'] / data_equipo['P_Jugados']
CarrerasP_x_Partido1 = data_equipo1['Carreras_Oponentes'] /
data_equipo1['P_Jugados']

fig = plt.figure(figsize=(12,6))
ax1 = fig.add_subplot(2,2,1)
ax2 = fig.add_subplot(2,2,2)
ax3 = fig.add_subplot(2,2,3)
ax4 = fig.add_subplot(2,2,4)
ax1.scatter(CarrerasG_x_Partido,data_equipo['Ganados'], c="green")
ax1.set_title("Carreras por Juegos vs. Juegos Ganados")
ax1.set_ylabel("Juegos Ganados")
ax1.set_xlabel("Carreras por Juego - Equipo: {}".format(equipo))
ax2.scatter(CarrerasP_x_Partido,data_equipo['Ganados'], c="red")
ax2.set_title("Carreras Permitidas por Juegos vs. Juegos Ganados")
ax2.set_ylabel("Juegos Ganados")
ax2.set_xlabel("Carreras Permitidas por Juego - Equipo: {}".format(equipo))
ax3.scatter(CarrerasG_x_Partido1,data_equipo1['Ganados'], c="green")
ax3.set_title("Carreras por Juegos vs. Juegos Ganados")
ax3.set_ylabel("Juegos Ganados")
ax3.set_xlabel("Carreras por Juego - Equipo: {}".format(equipo1))
ax4.scatter(CarrerasP_x_Partido1,data_equipo1['Ganados'], c="red")
ax4.set_title("Carreras Permitidas por Juegos vs. Juegos Ganados")
ax4.set_ylabel("Juegos Ganados")
ax4.set_xlabel("Carreras Permitidas por Juego - Equipo: {}".format(equipo1))
plt.tight_layout()
plt.show()
```



Entrenamiento de los modelos

```
# Eliminamos últimas columnas que no son necesarias
data_equipo = data_equipo.drop(['Agno','Equipo'], axis = 1)
data_equipo1 = data_equipo1.drop(['Agno','Equipo'], axis = 1)

# Importamos nuevas librerías para entrenar nuestro modelo
from sklearn.model_selection    import train_test_split
from sklearn.linear_model       import LinearRegression
```

```
from sklearn.metrics            import r2_score
from sklearn.metrics            import mean_squared_error

# Definimos las variables independientes y la dependiente
X = data_equipo.drop("Ganados",axis = 1)
y = data_equipo["Ganados"]
X1 = data_equipo1.drop("Ganados",axis = 1)
y1 = data_equipo1["Ganados"]

# Separamos entre entrenamiento y prueba
X_train, X_test, y_train, y_test        = train_test_split(X, y, test_size=0.20,
random_state=1)
X_train1, X_test1, y_train1, y_test1    = train_test_split(X1, y1, test_size=0.20,
random_state=1)

# Elegimos el modelo "Regresión Lineal"
algoritmo = LinearRegression()
algoritmo1 = LinearRegression()

# Entrenamos el algoritmo
algoritmo.fit(X_train, y_train)
algoritmo1.fit(X_train1, y_train1)

# Realizamos una predicción
y_test_pred = algoritmo.predict(X_test)
y_test_pred1 = algoritmo1.predict(X_test1)

# Calculamos la precisión del modelo
# Error promedio al cuadrado
# Calculo de R2
mse =mean_squared_error(y_test,y_test_pred)
rmse_rf =(mean_squared_error(y_test,y_test_pred))**(1/2)
r2 = r2_score(y_test,y_test_pred)
mse1 =mean_squared_error(y_test1,y_test_pred1)
rmse_rf1 =(mean_squared_error(y_test1,y_test_pred1))**(1/2)
r21 = r2_score(y_test1,y_test_pred1)
print("************************************************************")
print("**********       MSE {}:                      {}".format(equipo, mse))
print("**********       ERROR CUADRÁTICO MEDIO {}: {}".format(equipo, rmse_rf))
print("**********       R2 {}:                       {}".format(equipo, r2))
print("**********       MSE {}:                      {}".format(equipo1, mse1))
print("**********       ERROR CUADRÁTICO MEDIO {}: {}".format(equipo1, rmse_rf1))
print("**********       R2 {}:                       {}".format(equipo1, r21))
print("************************************************************")
```

```
************************************************************
**********       MSE NYA:                      0.4376403884470463
**********       ERROR CUADRÁTICO MEDIO NYA: 0.6615439429448706
**********       R2 NYA:                       0.9955653918840072
**********       MSE TEX:                      0.28840036545715286
**********       ERROR CUADRÁTICO MEDIO TEX: 0.5370292035421843
**********       R2 TEX:                       0.995453966496577
************************************************************
```

Código completo:

```
# Importamos librerías
import pandas as pd
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Importamos los datos contenidos en Team.csv
data = pd.read_csv('Teams.csv')
print(data)

# ************************************
# *****   ANÁLISIS DE LOS DATOS    *****
# ************************************
# Analizamos los datos
print(data.shape)
# Formato de los datos
print(data.dtypes)
# Buscamos datos nulos o faltantes
print(data.isnull().sum())

# ************************************
# ***** PROCESAMIENTO DE LOS DATOS *****
# ************************************

# Eliminación de columnas innecesarias
borrar_columnas = ['Liga','Franquicia','Clasificacion','Local',
                   'Gan_Division','Gan_WC','Gan_Liga','Gan_WS',
                   'Sacrificados','Nombre_Equipo','Nombre_Estadio',
                   'Asistencia_Estadio','Factor_Bateadores',
                   'Factor_Pitcher','Id_R','Id_BRW','Id_RW']
data = data.drop(borrar_columnas, axis=1)
data = data.drop(['Eliminados','Eliminados_Pitcher','Division'], axis=1)

# Completamos los datos nulos con la media de cada uno
data['Carreras']        = data['Carreras'].fillna(data['Carreras'].median())
data['Strike_Fuera']    = data['Strike_Fuera'].fillna(data['Strike_Fuera'].median())
data['Bases_Robadas']   = data['Bases_Robadas'].fillna(data['Bases_Robadas'].median())
print(data.shape)
print(data.dtypes)
print(data.isnull().sum())

# Separamos por eras o etapas
i =0
for year in data['Agno']:
    if year < 1920:
        data.loc[i,"era"]=1
    elif year >= 1920 and year <= 1941:
        data.loc[i, "era"] = 2
    elif year >= 1942 and year <= 1945:
        data.loc[i, "era"] = 3
    elif year >= 1946 and year <= 1962:
        data.loc[i, "era"] = 4
    elif year >= 1963 and year <= 1976:
        data.loc[i, "era"] = 5
    elif year >= 1977 and year <= 1992:
        data.loc[i, "era"] = 6
    elif year >= 1993 and year <= 2009:
        data.loc[i, "era"] = 7
    elif year >= 2010:
        data.loc[i, "era"] = 8
```

```
    i += 1

# Separamos por décadas
j =0
for year in data['Agno']:
    if year     <   1920:
        data.loc[j,"decada"]    =   1910

    elif year   >=  1920 and year   <=  1929:
        data.loc[j, "decada"]   =   1920

    elif year   >=  1930 and year   <=  1939:
        data.loc[j, "decada"]   =   1930

    elif year   >=  1940 and year   <=  1949:
        data.loc[j, "decada"]   =   1940

    elif year   >=  1950 and year   <=  1959:
        data.loc[j, "decada"]   =   1950

    elif year   >=  1960 and year   <=  1969:
        data.loc[j, "decada"]   =   1960

    elif year   >=  1970 and year   <=  1979:
        data.loc[j, "decada"]   =   1970

    elif year   >=  1980 and year   <=  1989:
        data.loc[j, "decada"]   =   1980

    elif year   >=  1990 and year   <=  1999:
        data.loc[j, "decada"]   =   1990

    elif year   >=  2000 and year   <=  2009:
        data.loc[j, "decada"]   =   2000

    elif year   >=  2010:
        data.loc[j, "decada"]   =   2010

    j += 1
print(data.Equipo)

# Seleccionamos un equipo para evaluar
equipo = 'NYA'
data_equipo = data.loc[data.Equipo == equipo]
equipo1 = 'TEX'
data_equipo1 = data.loc[data.Equipo == equipo1]

# ************************************
# *****   VISUALIZACIÓN GRÁFICA    *****
# ************************************
# Graficamos datos
fig = plt.figure(figsize=(12,6))
ax1 = fig.add_subplot(2,2,1)
ax2 = fig.add_subplot(2,2,2)
ax3 = fig.add_subplot(2,2,3)
ax4 = fig.add_subplot(2,2,4)

ax1.hist(data_equipo['Ganados'],bins=10, color='green',linewidth=1, edgecolor="black")
ax1.set_xlabel("JUEGOS GANADOS EQUIPO {}".format(equipo))
ax1.set_title("HISTOGRAMA JUEGOS GANADOS EQUIPO {}".format(equipo))

ax2.hist(data_equipo['Perdidos'],bins=10, color='red', linewidth=1, edgecolor="black")
```

```
ax2.set_xlabel("JUEGOS PERDIDOS EQUIPO {}".format(equipo))
ax2.set_title("HISTOGRAMA JUEGOS PERDIDOS EQUIPO {}".format(equipo))

ax3.hist(data_equipo1['Ganados'],bins=10, color='green', linewidth=1, edgecolor="black")
ax3.set_xlabel("JUEGOS GANADOS EQUIPO {}".format(equipo1))
ax3.set_title("HISTOGRAMA JUEGOS GANADOS EQUIPO {}".format(equipo1))

ax4.hist(data_equipo1['Perdidos'],bins=10, color='red',linewidth=1, edgecolor="black")
ax4.set_xlabel("JUEGOS PERDIDOS EQUIPO {}".format(equipo1))
ax4.set_title("HISTOGRAMA JUEGOS PERDIDOS EQUIPO {}".format(equipo1))
plt.tight_layout()
plt.show()

# Graficamos las carreras realizadas
fig = plt.figure(figsize=(12,6))
ax1 = fig.add_subplot(1,2,1)
ax2 = fig.add_subplot(1,2,2)
ax1.plot(data_equipo['Agno'],data_equipo['Carreras'],linewidth=2.0,color="green")
ax1.set_title("CARRERAS REALIZADAS / AÑO - EQUIPO: {}".format(equipo))
ax1.set_ylabel("CARRERAS REALIZADAS")
ax1.set_xlabel("AÑO")
ax2.plot(data_equipo1['Agno'],data_equipo1['Carreras'],linewidth=2.0,color="green")
ax2.set_title("CARRERAS REALIZADAS / AÑO - EQUIPO: {}".format(equipo1))
ax2.set_xlabel("AÑO")
ax2.set_ylabel("CARRERAS REALIZADAS")
plt.tight_layout()
plt.show()

# Gráficas adicionales
CarrerasG_x_Partido  = data_equipo['Carreras_Ganadas'] / data_equipo['P_Jugados']
CarrerasG_x_Partido1 = data_equipo1['Carreras_Ganadas'] / data_equipo1['P_Jugados']
CarrerasP_x_Partido = data_equipo['Carreras_Oponentes'] / data_equipo['P_Jugados']
CarrerasP_x_Partido1 = data_equipo1['Carreras_Oponentes'] / data_equipo1['P_Jugados']

fig = plt.figure(figsize=(12,6))
ax1 = fig.add_subplot(2,2,1)
ax2 = fig.add_subplot(2,2,2)
ax3 = fig.add_subplot(2,2,3)
ax4 = fig.add_subplot(2,2,4)
ax1.scatter(CarrerasG_x_Partido,data_equipo['Ganados'], c="green")
ax1.set_title("Carreras por Juegos vs. Juegos Ganados")
ax1.set_ylabel("Juegos Ganados")
ax1.set_xlabel("Carreras por Juego - Equipo: {}".format(equipo))
ax2.scatter(CarrerasP_x_Partido,data_equipo['Ganados'], c="red")
ax2.set_title("Carreras Permitidas por Juegos vs. Juegos Ganados")
ax2.set_ylabel("Juegos Ganados")
ax2.set_xlabel("Carreras Permitidas por Juego - Equipo: {}".format(equipo))
ax3.scatter(CarrerasG_x_Partido1,data_equipo1['Ganados'], c="green")
ax3.set_title("Carreras por Juegos vs. Juegos Ganados")
ax3.set_ylabel("Juegos Ganados")
ax3.set_xlabel("Carreras por Juego - Equipo: {}".format(equipo1))
ax4.scatter(CarrerasP_x_Partido1,data_equipo1['Ganados'], c="red")
ax4.set_title("Carreras Permitidas por Juegos vs. Juegos Ganados")
ax4.set_ylabel("Juegos Ganados")
ax4.set_xlabel("Carreras Permitidas por Juego - Equipo: {}".format(equipo1))
plt.tight_layout()
plt.show()

# Eliminamos últimas columnas que no son necesarias
data_equipo = data_equipo.drop(['Agno','Equipo'], axis = 1)
data_equipo1 = data_equipo1.drop(['Agno','Equipo'], axis = 1)
```

```
# Importamos nuevas librerías para entrenar nuestro modelo
from sklearn.model_selection    import train_test_split
from sklearn.linear_model       import LinearRegression
from sklearn.metrics            import r2_score
from sklearn.metrics            import mean_squared_error

# Definimos las variables independientes y la dependiente
X = data_equipo.drop("Ganados",axis = 1)
y = data_equipo["Ganados"]
X1 = data_equipo1.drop("Ganados",axis = 1)
y1 = data_equipo1["Ganados"]

# Separamos entre entrenamiento y prueba
X_train, X_test, y_train, y_test        = train_test_split(X, y, test_size=0.20,
random_state=1)
X_train1, X_test1, y_train1, y_test1    = train_test_split(X1, y1, test_size=0.20,
random_state=1)

# Elegimos el modelo "Regresión Lineal"
algoritmo = LinearRegression()
algoritmo1 = LinearRegression()

# Entrenamos el algoritmo
algoritmo.fit(X_train, y_train)
algoritmo1.fit(X_train1, y_train1)

# Realizamos una predicción
y_test_pred = algoritmo.predict(X_test)
y_test_pred1 = algoritmo1.predict(X_test1)

# Calculamos la precisión del modelo
# Error promedio al cuadrado
# Calculo de R2
mse =mean_squared_error(y_test,y_test_pred)
rmse_rf =(mean_squared_error(y_test,y_test_pred))**(1/2)
r2 = r2_score(y_test,y_test_pred)
mse1 =mean_squared_error(y_test1,y_test_pred1)
rmse_rf1 =(mean_squared_error(y_test1,y_test_pred1))**(1/2)
r21 = r2_score(y_test1,y_test_pred1)
print("************************************************************")
print("**********       MSE {}:                    {}".format(equipo, mse))
print("**********       ERROR CUADRÁTICO MEDIO {}: {}".format(equipo, rmse_rf))
print("**********       R2 {}:                     {}".format(equipo, r2))
print("**********       MSE {}:                    {}".format(equipo1, mse1))
print("**********       ERROR CUADRÁTICO MEDIO {}: {}".format(equipo1, rmse_rf1))
print("**********       R2 {}:                     {}".format(equipo1, r21))
print("************************************************************")
```