PROYECTO RECOMENDACIÓN DE ANIMES

INTEGRANTES:

EMIRO MORENO SOTO

JENNIFER DUQUE ORTIZ

DANIELA JIMÉNEZ GÓMEZ

CURSO:

INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL

UNIVERSIDAD DE ANTIOQUIA

FACULTAD DE INGENIERÍA

25 DE MAYO DEL 2023

INTRODUCCIÓN

De acuerdo a la necesidad de una empresa de streaming especializada en anime en saber las características de los más detectados y los mejores puntuados por los usuarios, el siguiente informe predice predice aquellos factores que hacen relevante que dicho anime tenga éxito.

Usando el dataset de kaggle encontrado en: https://www.kaggle.com/datasets/hernan4444/anime-recommendation-database-202, dónde el conjunto de datos contiene información sobre 17.562 anime y la preferencia de 325.772 usuarios diferentes. En particular, este conjunto de datos contiene, calificaciones dadas por los usuarios a los animes que han visto por completo entre otros datos.

Como métrica de Machine Learning se utilizará el método de regresión Gretl, el cual se podrá usar con algunas librerías importadas para esto, como hipótesis se emplearon 9 variables para saber que tan acogido será el anime y en base a esto elegir si lanzarlo o no. Como métrica de negocio se podría verificar el incremento en usuario, si durante un mes se ve un aumento del 8% se dejará el modelo y se le harán mejoras.
Si los anime que han sido seleccionado con la ayuda de este modelo predictivo no tienen gran acogida después de un semestre se dejarán de usar.

DESARROLLO

Para este avance de proyecto sobre “Recomendaciones de animes” donde se va a predecir factores que hacen relevante que dicho anime tenga éxito, se toma como punto de inicio el conjunto de datos que contiene la información oportuna para implementar el adecuado desarrollo del mismo.

Esta información contiene la preferencia de diferentes usuarios que alimentan la base de datos a utilizar; los cuales muestran los listados por usuario, con su estado de descartado, planeado para ver, los que ven actualmente y aquellos que tienen en espera, de esta forma también contiene las características como género, estudio, estadísticas y calificaciones que dan los usuarios de 1 a 10 y si lo han visto completamente. Estas bases ayudan a separar las particularidades de cada anime como una información HTML, en estos archivos se encuentra además reseñas, sinopsis, información sobre el personal, sus estadísticas de anime entre otras características.

Las siguientes imágenes evidencian el paso a paso de lo realizado al momento:

1. Se importan las librerías necesarias para que se ejecuten las líneas correctamente.
2. Se da el permiso al programa para que acceda al archivo alojado en el drive.
3. Se carga el dataset seleccionado (contiene 17562 datos).
4. Se define las columnas que se desean mostrar, debido a que es un conjunto de datos amplio, por lo tanto, es necesario limitarlo
5. Se define que sólo se desea mostrar 35 columnas del data set.
6. Se verifican los encabezados antes mencionados.
7. Se realiza una limpieza de los datos para evitar errores en la ejecución.
8. Se filtran las variables que se requieren para el análisis.

PROYECTO DEL CURSO INTELIGENCIA ARTIFICIAL

Haz doble clic (o pulsa Intro) para editar

```
[ ] #Importamos las librerias que necesitamos
    import math
    import matplotlib.pyplot as plt
    import numpy as np
    import pandas as pd
    import seaborn as sns
    import sklearn

    from sklearn.preprocessing import scale
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import mean_squared_error
    from sklearn.metrics import r2_score
```

```
from google.colab import drive
drive.mount('/content/drive')

#Damos permiso para usar archivos de nuestro Drive
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[ ] df=pd.read_csv('/content/drive/MyDrive/anime.csv')
    df.shape
    #Cargamos el data set a usar y vemos sus dimensiones
```

```
(17562, 35)
```

```
[ ] pd.set_option('display.max_columns',35)
    #definimos las columnas para mostra por pantalla
```

```
[ ] df.head()
    #vemos los encabezados antes definidos
```

	MAL_ID	Name	Score	Genres	English name	Japanese name	Type	Episodes	Aired	Premiered	Producers	Licensors	Studios	Source	Duration	
0	1	Cowboy Bebop	8.78	Action, Adventure, Comedy, Drama, Sci-Fi, Space	Cowboy Bebop	カウボーイビバップ	TV	26	Apr 3, 1998 to Apr 24, 1999	Spring 1998	Bandai Visual	Funimation, Bandai Entertainment	Sunrise	Original	24 min. per ep.	(\ pr
1	5	Cowboy Bebop: Tengoku no Tobira	8.39	Action, Drama, Mystery, Sci-Fi, Space	Cowboy Bebop:The Movie	カウボーイビバップ 天国の扉	Movie	1	Sep 1, 2001	Unknown	Sunrise, Bandai Visual	Sony Pictures Entertainment	Bones	Original	1 hr. 55 min.	(\ pr
2	6	Trigun	8.24	Action, Sci-Fi, Adventure, Comedy, Drama, Shounen	Trigun	トライガン	TV	26	Apr 1, 1998 to Sep 30, 1998	Spring 1998	Victor Entertainment	Funimation, Geneon Entertainment USA	Madhouse	Manga	24 min. per ep.	I Ti
3	7	Witch Hunter	7.27	Action, Mystery, Police	Witch Hunter	Witch Hunter ROBIN	TV	26	Jul 2, 2002 to	Summer	TV Tokyo, Bandai Visual,	Funimation,	Sunrise	Original	25 min.	I Ti

Activar Windows
Ve a Configuración para activar Windows.

```
df = df.replace('Unknown', float(0))
#hacemos una limpieza de datos reemplazndo string por numeor flotante cero para evitar error con el algoritmo
```

```
df.head(2)
```

	MAL_ID	Name	Score	Genres	English name	Japanese name	Type	Episodes	Aired	Premiered	Producers	Licensors	Studios	Source	Duration	Rating
0	1	Cowboy Bebop	8.78	Action, Adventure, Comedy, Drama, Sci-Fi, Space	Cowboy Bebop	カウボーイビバップ	TV	26	Apr 3, 1998 to Apr 24, 1999	Spring 1998	Bandai Visual	Funimation, Bandai Entertainment	Sunrise	Original	24 min. per ep.	R - 17+ (violence & profanity)
1	5	Cowboy Bebop: Tengoku no Tobira	8.39	Action, Drama, Mystery, Sci-Fi, Space	Cowboy Bebop:The Movie	カウボーイビバップ 天国の扉	Movie	1	Sep 1, 2001	0.0	Sunrise, Bandai Visual	Sony Pictures Entertainment	Bones	Original	1 hr. 55 min.	R - 17+ (violence & profanity)

Activar Windows
Ve a Configuración para activar Windows.

```
[ ] df=df.drop(['MAL_ID', 'Name', 'Genres', 'English name', 'Japanese name',
        'Type', 'Aired', 'Premiered', 'Producers', 'Licensors',
        'Studios', 'Source', 'Duration', 'Rating', 'Ranked', 'Score-10', 'Score-9', 'Score-8', 'Score-7', 'Score-6',
        'Score-5', 'Score-4', 'Score-3', 'Score-2', 'Score-1'],axis=1)

    #filtramos solo las variables a usar
```

```
[ ] df.head()
```

	Score	Episodes	Popularity	Members	Favorites	Watching	Completed	On-Hold	Dropped	Plan to Watch
0	8.78	26	39	1251960	61971	105808	718161	71513	26678	329800
1	8.39	1	518	273145	1174	4143	208333	1935	770	57964
2	8.24	26	201	558913	12944	29113	343492	25465	13925	146918
3	7.27	26	1467	94683	587	4300	46165	5121	5378	33719
4	6.98	52	4369	13224	18	642	7314	766	1108	3394

Activar
Ve a Confi

Continuando con el informe del proyecto sobre “Recomendaciones de animes” y después de concretar los datos a utilizar para su prueba, se convierten todos los valores a tipo flotantes para que el modelo pueda procesar más efectivamente, de este modo al dimensionar el proyecto resulta más fácil para limpiar y filtrar los data set de información con las respectivas variables de análisis y respuesta.

```
#convertir todos los valores a tipo flotante para que el modelo lo pueda procesar
df=df.astype(float)
```

```
[ ] #ver dimensiones del data set final con el que vamos a trabajar
    df.shape

    (17562, 10)
```

```
#filtro para las variable score la cual es nuestra variable de respuesta
df = df[df.Score > 0.0]
```

```
[ ] df.isnull().sum()

    Score          0
    Episodes       0
    Popularity     0
    Members        0
```

De este modo las variables serían:

```
#las  siguiente asignamos valores de respuesta Y y las variables pedictoras.
df.columns
```

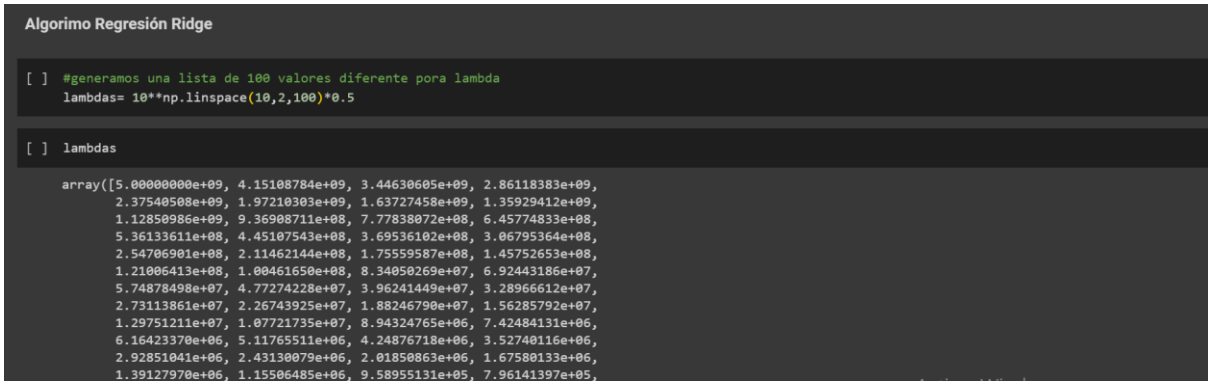
```
Index(['Score', 'Episodes', 'Popularity', 'Members', 'Favorites', 'Watching',
       'Completed', 'On-Hold', 'Dropped', 'Plan to Watch'],
      dtype='object')
```

```
[ ] Y=df.Score
```

```
[ ] X=df.drop(['Score'],axis=1)
```

```
[ ] Y.head(2)

    0    8.78
    1    8.39
    Name: Score, dtype: float64
```

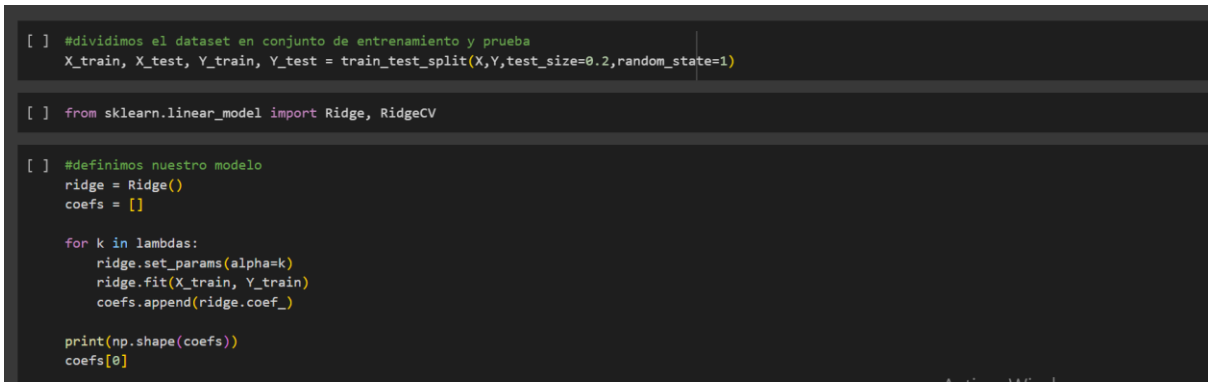
Generando la lista de 100 valores lamdas procedemos a dividir los dataset en conjuntos de entrenamiento y prueba donde definiremos el modelo para generar una observación más clara de nuestro proyecto sobre “Recomendaciones de animes”.



Algorimo Regresión Ridge

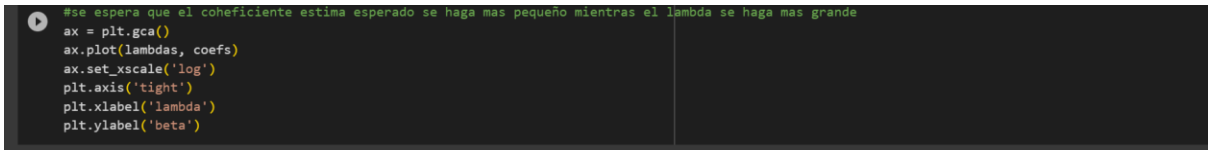
```
[ ] #generamos una lista de 100 valores diferente pora lambda
    lambdas= 10**np.linspace(10,2,100)*0.5

[ ] lambdas

    array([5.00000000e+09, 4.15108784e+09, 3.44630605e+09, 2.86118383e+09,
           2.37540508e+09, 1.97210303e+09, 1.63727458e+09, 1.35929412e+09,
           1.12850986e+09, 9.36908711e+08, 7.77838072e+08, 6.45774833e+08,
           5.36133611e+08, 4.45107543e+08, 3.69536102e+08, 3.06795364e+08,
           2.54706901e+08, 2.11462144e+08, 1.75559587e+08, 1.45752653e+08,
           1.21006413e+08, 1.00461650e+08, 8.34050269e+07, 6.92443186e+07,
           5.74878498e+07, 4.77274228e+07, 3.96241449e+07, 3.28966612e+07,
           2.73113861e+07, 2.26743925e+07, 1.88246790e+07, 1.56285792e+07,
           1.29751211e+07, 1.07721735e+07, 8.94324765e+06, 7.42484131e+06,
           6.16423370e+06, 5.11765511e+06, 4.24876718e+06, 3.52740116e+06,
           2.92851041e+06, 2.43130079e+06, 2.01850863e+06, 1.67580133e+06,
           1.39127970e+06, 1.15506485e+06, 9.58955131e+05, 7.96141397e+05,
```

Se divide el dataset para separar los datos de entrenamiento de los de prueba, a su vez, se define el modelo a usar.

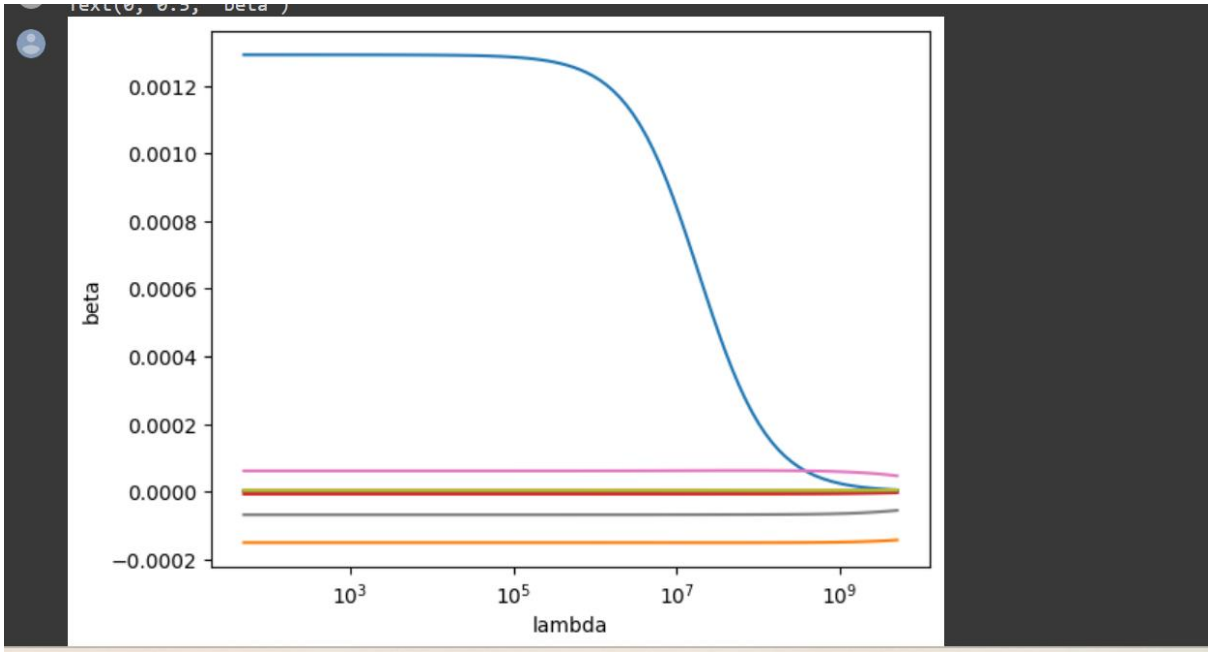


```
[ ] #dividimos el dataset en conjunto de entrenamiento y prueba
    X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2,random_state=1)

[ ] from sklearn.linear_model import Ridge, RidgeCV

[ ] #definimos nuestro modelo
    ridge = Ridge()
    coefs = []

    for k in lambdas:
        ridge.set_params(alpha=k)
        ridge.fit(X_train, Y_train)
        coefs.append(ridge.coef_)

    print(np.shape(coefs))
    coefs[0]
```

Cuando se ejecuta el modelo, se espera que el coeficiente se haga pequeño, mientras lambda crece, como se muestra en la imagen.



```
#se espera que el coheficiente estima esperado se haga mas pequeño mientras el lambda se haga mas grande
ax = plt.gca()
ax.plot(lambdas, coefs)
ax.set_xscale('log')
plt.axis('tight')
plt.xlabel('lambda')
plt.ylabel('beta')
```



Al aplicar la regresión Ridge con alpha=4, arroja los datos para evaluar lo que se está buscando.

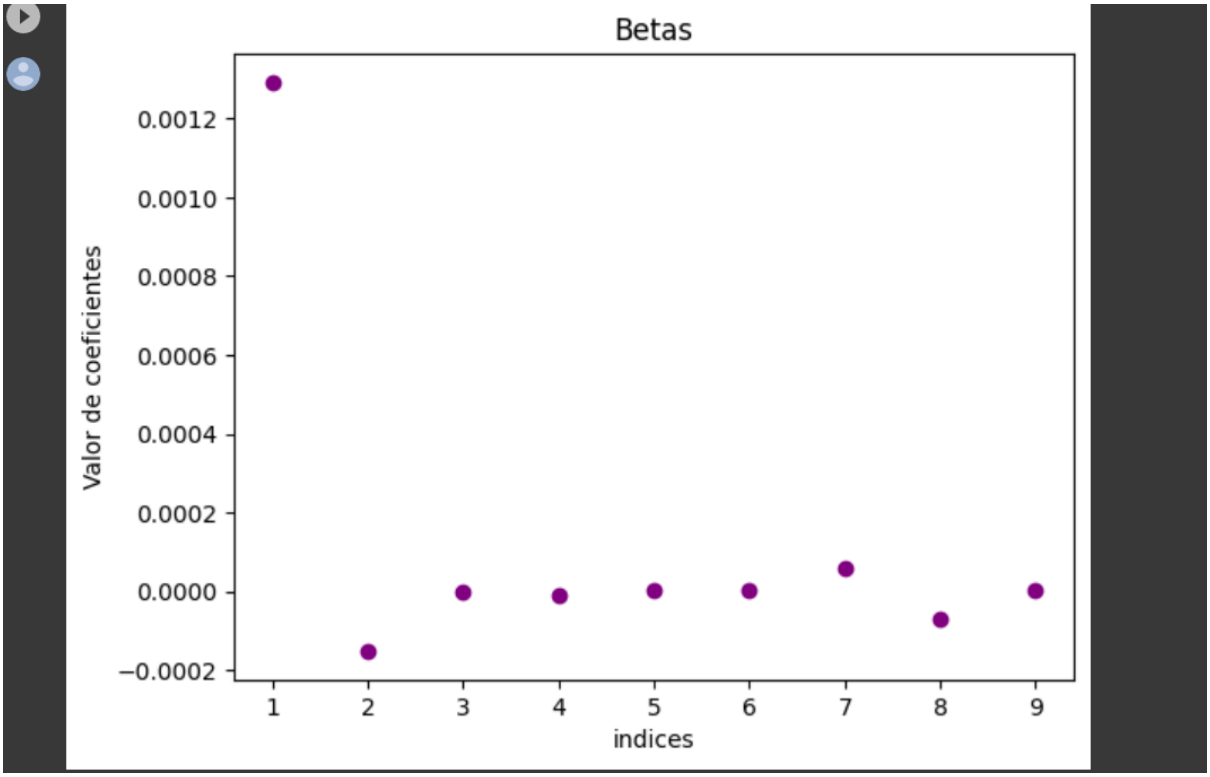
Regresión Ridge con λ=4

+ Código + Texto

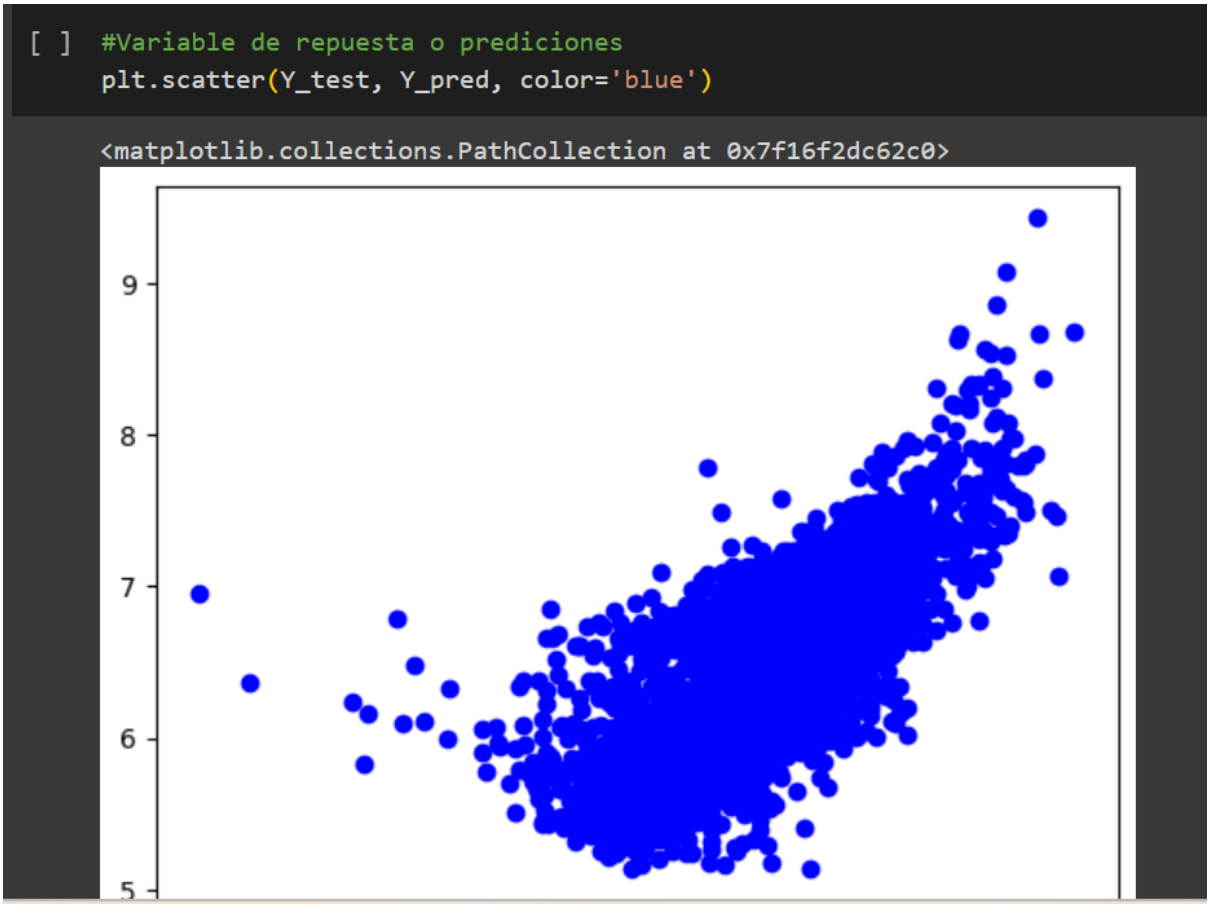
```
[ ]  #score de nuestras variables alpha=4
     mod_ridge4 = Ridge(alpha =4)
     mod_ridge4.fit(X_train, Y_train)
     print(pd.Series(mod_ridge4.coef_ , index=X.columns))
```

```
Episodes         1.291738e-03
Popularity      -1.511535e-04
Members         -8.258051e-07
Favorites       -7.692250e-06
Watching         2.094438e-06
Completed        1.729744e-06
On-Hold          6.097291e-05
Dropped         -6.892429e-05
Plan to Watch    3.414316e-06
dtype: float64
```

```
[ ]  #valor de las variables o categorias selecionada del data set para nuestro entrenamiento
     eje_X = range(1,len(mod_ridge4.coef_)+1,1)
     plt.scatter(eje_X, mod_ridge4.coef_, color='purple')
     plt.title('Betas')
     plt.xlabel('indices')
     plt.ylabel('Valor de coeficientes')
```



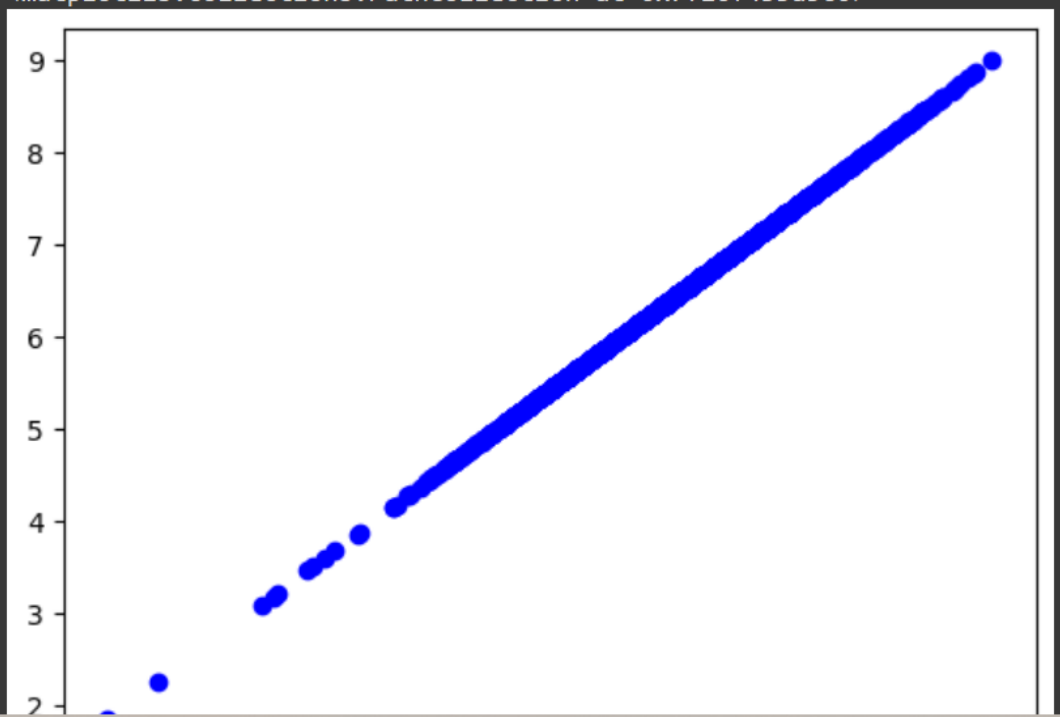
La siguiente imagen muestra la variable respuesta a las predicciones realizadas.



La variables estimada se aleja de lo evidenciado anteriormente, sin embargo, es una muestra de que no se comporta como se esperaba.

```
#Variable estimada
plt.scatter(Y_test, Y_test, color='blue')
```

```
<matplotlib.collections.PathCollection at 0x7f16f433a5c0>
```



CONCLUSIONES

1. Importación de bibliotecas: El código comienza importando varias bibliotecas populares para el análisis de datos, como math, matplotlib, numpy, pandas, seaborn y sklearn. Estas bibliotecas proporcionan funciones y herramientas para el manejo y análisis de datos.
2. Lectura y manipulación de datos: El código carga un archivo CSV llamado "anime.csv" utilizando pandas y realiza algunas operaciones de limpieza y preparación de datos, como eliminar columnas no deseadas y reemplazar valores desconocidos por ceros.
3. División de datos: Los datos se dividen en variables de entrada (X) y variable de salida (Y), donde se pretende predecir la puntuación de los animes. Además, se realiza una división adicional en conjuntos de entrenamiento y prueba utilizando la función train_test_split de sklearn.

4. Modelo de regresión Ridge: Se utiliza el algoritmo de regresión Ridge para ajustar un modelo a los datos de entrenamiento. Se prueba una serie de valores de lambda (alfa) para encontrar el mejor ajuste del modelo. Se registran los coeficientes correspondientes a cada valor de lambda.
5. Visualización de resultados: Se traza un gráfico para mostrar cómo los coeficientes del modelo varían con respecto a los diferentes valores de lambda. También se muestra una representación gráfica de los coeficientes finales del modelo Ridge.
6. Predicción y evaluación: Se realizan predicciones sobre los datos de prueba utilizando el modelo Ridge ajustado y se muestra un gráfico de dispersión para comparar las predicciones con los valores reales.

En resumen, el código muestra un ejemplo de cómo implementar un modelo de regresión Ridge para predecir la puntuación de los animes. Se aprendió sobre la importancia de la preparación de datos, la selección de características y la evaluación del modelo en el contexto del análisis de datos. Además, se ilustra el uso de diversas bibliotecas de Python para realizar estas tareas de manera eficiente.