Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Puebla

TC3006C. Inteligencia artficial avancada para la ciencia de datos
(Grupo 103)

Módulo 2 Análisis y Reporte sobre el desempeño del modelo.

Alejandro López Hernández A01733984

8 de Septiembre 2022

Introducción: Dataset empleado

El dataset designado para esta implementación refiere a anime; considerando series, películas y OVAs. La información incluye algunos datos como el título, la duración (para películas), si se encuentra en emisión, el año de inicio, el año de finalización, la descripción, el estudio de animación, cuántos usuarios han visto el anime, cuántos usuarios se encuentran viendo el anime, cuántos usuarios quieren ver el anime, cuántos usuarios dejaron el anime inconcluso, el rating, los votos que ha recibido, entre otros.

Para efectos del modelo, se determinó emplear sólo 2 variables con tal de poder realizar una regresión lineal simple con una variable dependiente y una variable independiente. La primera columna es la cantidad de votos que ha recibido el anime en cuestión; y la segunda variable consta del rating o calificación que tiene el anime en una calificación de 0 a 5

La tendencia esperada dentro del modelo es lógica y sencilla: a mayor cantidad de votos, mayor será la calificación del ánime. A continuación se muestra un resumen del dataset con algunas de las columnas relevantes.

	title	mediaType	eps	duration	ongoing	startYr	...	watched	watching	wantWatch	dropped	rating	votes
1	Fullmetal Alchemist: Brotherhood	TV	64	NaN	False	2009	...	103707.0	14351	25810	2656	4.702	86547
2	your name.	Movie	1	107	False	2016	...	58831.0	1453	21733	124	4.663	43960
3	A Silent Voice	Movie	1	130	False	2016	...	45892.0	946	17148	132	4.661	33752
4	Haikyuu!! Karasuno High School vs Shiratorizaw...	TV	10	NaN	False	2016	...	25134.0	2183	8082	167	4.660	17422
5	Attack on Titan 3rd Season: Part II	TV	10	NaN	False	2019	...	21308.0	3217	7864	174	4.650	15789
...	...	...	..	...	...	...	...	...	...	...	...	...	...
12706	Bonobono: Uchuu kara Kita Tomodachi	Movie	1	22	False	2017	...	17.0	2	34	1	2.473	10
12718	Sore Ike! Anpanman: Kirameke! Ice no Kuni no V...	Movie	1	NaN	False	2019	...	22.0	1	29	1	2.807	10
12833	Hulaing Babies Petit	TV	12	5	False	2020	...	13.0	10	77	2	2.090	10
13081	Marco & The Galaxy Dragon	OVA	1	NaN	False	2020	...	17.0	0	65	0	2.543	10
13374	Ultra B: Black Hole kara no Dokusaisha BB!!	Movie	1	20	False	1988	...	15.0	1	19	1	2.925	10

Link del dataset: https://www.kaggle.com/datasets/alancmathew/anime-dataset

Herramientas utilizadas

Pandas: Librería de python open source dedicada al análisis de datos de manera intuitiva y flexible. Pandas puede abarcar la gran mayoría de ETL, sobre todo siendo fundamental para la extracción, limpieza y transformación de datos a emplear.

Numpy: Librería de python dedicada a ciencia de datos desde la perspectiva de vectorización, indexado, funciones matemáticas, funciones algebraicas, permutación de matrices, cálculo vectorial, etcétera.

Matplotlib: Librería de python dedicada a la graficación de datos, fundamental para poder comprender el comportamiento de los datos, por ejemplo, a partir de su ordenamiento o introducción a un modelo determinado.

Sci-kit Learn: Librería de python dedicada al análisis de datos y aprendizaje máquina que va desde modelos simples como regresiones lineales o logísticas hasta redes neuronales complejas.

Limpieza de datos

Como primer paso, se lee el archivo csv descargado de Kaggle y alojado en la carpeta del proyecto. Empleando pandas, se genera un dataframe a partir del archivo; dicho dataframe vendrá con las columnas ya etiquetadas con sus respectivos nombres previamente alojados en una lista.

```
#Obtención de los datos a partir de un csv con pandas
columns =
["title","mediaType","eps","duration","ongoing","startYr","finishYr","s
znOfRelease","description","studios","tags","contentWarn","watched","wa
tching","wantWatch","dropped","rating","votes"]
df = pd.read_csv('mod2_machinelearning/anime.csv', names = columns)
```

Se hace limpieza del data frame, eliminando aquellos registros que tengan valores NaN en ambas columnas a utilizar.

```
#Limpieza de datos, considerando que no puede haber datos vacíos en
dichas columnas
df = df.dropna(subset=['rating', 'votes'])
```

Posteriormente, se extraen las dos columnas principales para convertirlas al tipo de dato float. Al estar inicialmente en string, ambas columnas no podrían emplearse para el modelo de regresión ni para los siguientes pasos de limpieza de datos.

```
#conversion de columnas principales a tipo float
df['votes']= df['votes'][1:].astype(int)
df['rating'] = df['rating'][1:].astype(float)
```

Por último, se hace una última limpieza, descartando los casos extremos en los que algún anime no tenga votos; puesto que es fundamental para el modelo a implementar que no haya valores en 0 para la variable independiente.

```
#Los animes deben tener al menos 1 voto
df = df[df['votes'] > 0]

print(df)
```

Modelo de regresión lineal

El modelo a emplear es una regresión lineal simple; que consta de una variable dependiente y de una variable independiente. La base principal de la regresión lineal

simple recae en la fórmula y = mx + b, de donde se desglosa la pendiente (m) y el intercepto (b).

Como primer paso, se desglosa el data frame en dos arreglos: el arreglo para el eje X con los valores de la columna de votos en logaritmo natural; y el arreglo para el eje Y con los valores de la columna de rating. Se optó por guardar el logaritmo natural de los votos debido a que se buscó una forma de poder agrupar un poco más los datos sin tener que afectar realmente su valor. Con datos con valores más pequeños en base de logaritmo natural, el modelo podrá trabajar de una manera más precisa en términos de regresión lineal y tendrá coeficientes razonables.

Posteriormente se hace un llamado a la función de numpy reshape, que redimensiona los arreglos en matrices con tal de que la función de regresión lineal pueda interpretarlos.

```
#determinación de la variable correspondiente al eje X como los votos del anime
X = np.array(np.log(df['votes'])).reshape(-1,1)
#determinación de la variable Y como la calificación del anime
Y = np.array(df['rating']).reshape(-1,1)
```

Después de haber obtenido X e Y, se hace la división de los datos en train y test con la función train_test_split. Dicha función retorna 4 variables de listas divididas en el X e Y tanto para el set de testing como para el set de training. La división quedó de la siguiente manera: 75% de los datos fueron asignados a set de training y 25% al set del testing.

```
#division del data set en train y test
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25)
```

Posteriormente, se hace el llamado a la función de scikit learn correspondiente a la regresión lineal en su configuración predeterminada.

```
#llamada a la regresión lineal de scikit learn
regr = LinearRegression()
regr.fit(X_train, Y_train)
#coeficiente e intercepto
print("Coefficient: ", regr.coef_)
print("Intercept: ", regr.intercept_)
#predicciones
print("Predicitons:")
custom_pred = [[1234], [2345], [6349], [42368], [923], [14]]

for i in custom_pred:
    print(f"Votos: {i} Rating aproximado: {regr.predict(np.log([i]))}")
```

Para probar el modelo, se hace la predicción de personas que habrían visto un anime con determinado rating de 0 a 5; considerando puntos decimales.

```
#predicciones
print("Predicitons:")
custom_pred = [[3.7], [4.5], [2.9], [1.3], [0.5]]

for i in custom_pred:
    print(f"Rating del anime: {i} Usuarios que lo han visto: {regr.predict([i])}")
```

```
Predicitons:
Rating del anime: [3.7] Usuarios que lo han visto: [[4486.08958618]]
Rating del anime: [4.5] Usuarios que lo han visto: [[5456.05490211]]
Rating del anime: [2.9] Usuarios que lo han visto: [[3516.12427025]]
Rating del anime: [1.3] Usuarios que lo han visto: [[1576.19363839]]
Rating del anime: [0.5] Usuarios que lo han visto: [[606.22832246]]
```

Antes de la graficación, se determina de manera rápida el error en las predicciones tanto de train como de test. Para determinar el error se realiza una resta entre la predicción para Y y los datos originales en Y en ambos sectores de datos.

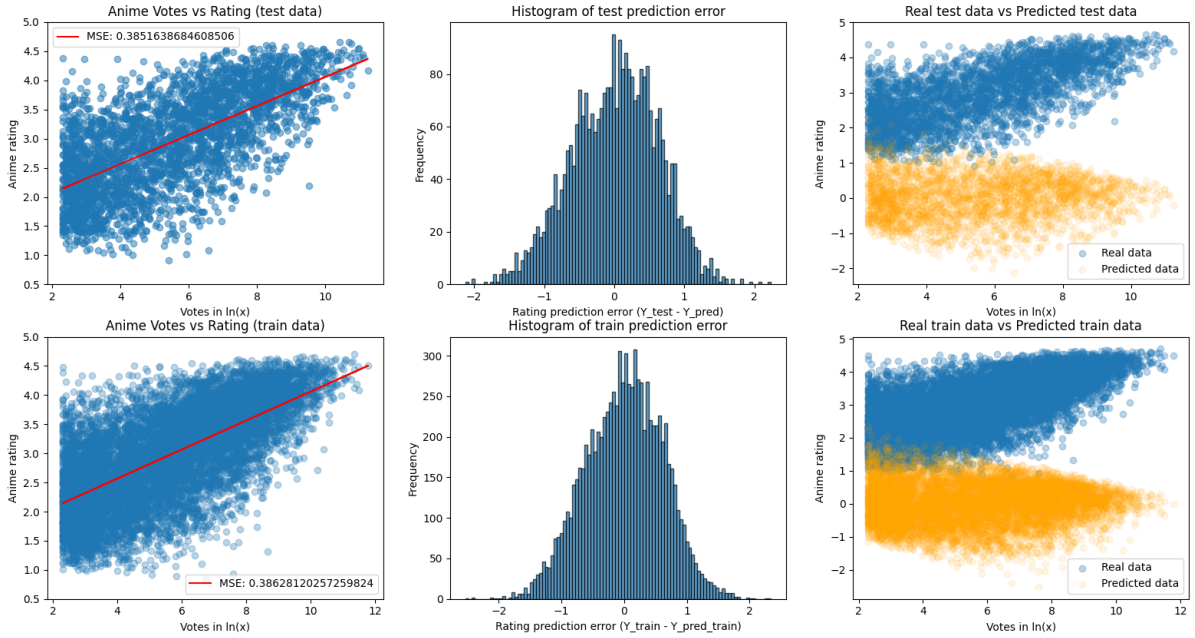
```
#obtencion del error de prediccion en test y train
Y_pred = regr.predict(X_test)
Pred_error_test = Y_pred - Y_test
Y_pred_train = regr.predict(X_train)
Pred_error_train = Y_pred_train - Y_train
```

Finalmente, se hace una graficación de la regresión lineal tanto para train como data. Del mismo modo, se grafica un histograma con los resultados de los errores en la predicción.

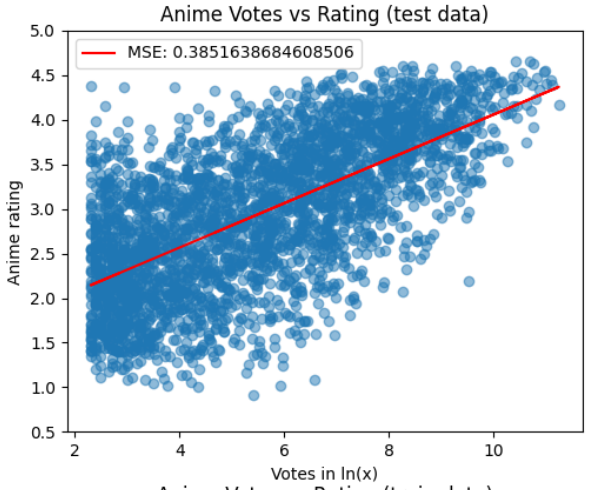
```
#plot
figure, axis = plt.subplots(2,3)
#TEST
#regresion
axis[0,0].scatter(X_test, Y_test, alpha = 0.5)
axis[0,0].plot(X_test, Y_pred, color='red')
axis[0,0].set_title("Anime Votes vs Rating (test data)")
axis[0,0].set(xlabel = 'Votes in ln(x)', ylabel = 'Anime rating')
axis[0,0].set_ylim([0.5,5])

#histograma(bias)
axis[0,1].hist(Pred_error_test, alpha = 0.7, edgecolor = 'black', bins = 100)
axis[0,1].set_title('Histogram of test prediction error')
axis[0,1].set(xlabel = 'Watched prediction error (Y_test - Y_pred)', ylabel = 'Frequency')

#varianza(?)
axis[0,2].scatter(X_test, Y_test, alpha = 0.3, label = 'Real data')
axis[0,2].scatter(X_test, Pred_error_test, color='orange',alpha = 0.1, label = 'Predicted
data')
axis[0,2].set_title("Real test data vs Predicted test data")
axis[0,2].set(xlabel = 'Votes in ln(x)', ylabel = 'Anime rating')
axis[0,2].legend()
```

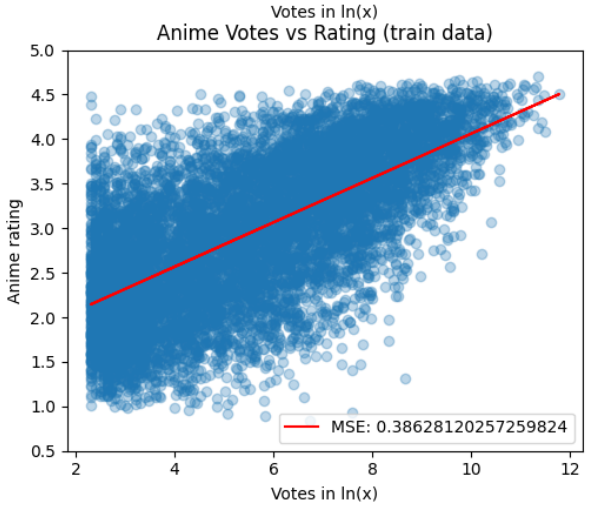
```
#TRAIN
#regresion
axis[1,0].scatter(X_train, Y_train, alpha = 0.3)
axis[1,0].plot(X_train, Y_pred_train, color='red')
axis[1,0].set_title("Anime Votes vs Rating (train data)")
axis[1,0].set(xlabel = 'Votes in ln(x)', ylabel = 'Anime rating')
axis[1,0].set_ylim([0.5,5])

#histograma(bias)
axis[1,1].hist(Pred_error_train, alpha = 0.7,edgecolor = 'black', bins = 100)
axis[1,1].set_title('Histogram of train prediction error')
axis[1,1].set(xlabel = 'Watched prediction error (Y_train - Y_pred_train)', ylabel =
'Frequency')

#varianza(?)
axis[1,2].scatter(X_train, Y_train, alpha = 0.3, label = 'Real data')
axis[1,2].scatter(X_train, Pred_error_train, color='orange',alpha = 0.1, label = 'Predicted
data')
axis[1,2].set_title("Real train data vs Predicted train data")
axis[1,2].set(xlabel = 'Votes in ln(x)', ylabel = 'Anime rating')
axis[1,2].legend()
plt.show()
```

El resultado de la graficación es el siguiente.



Asimismo, las gráficas correspondientes a la regresión lineal quedaron de la siguiente manera; con los votos en proporción a su logaritmo natural como variable independiente y el rating del anime como la dependiente. Asimismo, se denota la línea de regresión correspondiente a la predicción elaborada por el modelo.

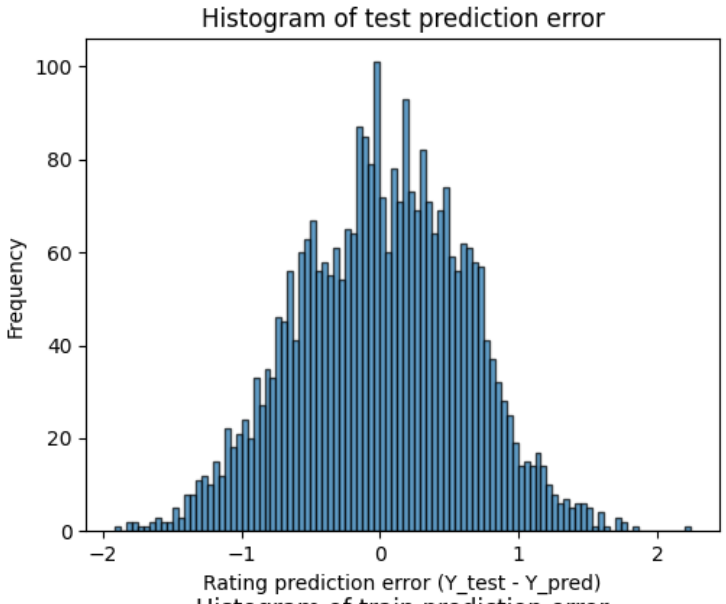


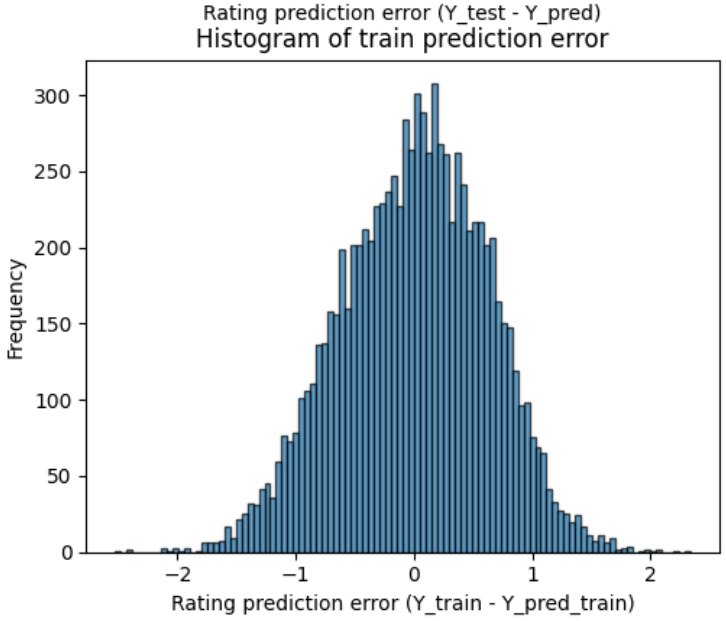


Por último, se hicieron predicciones para probar si la implementación del modelo estaba correcta en términos de lo esperado por la regresión lineal simple. Asimismo, de determinaron los coeficientes del mean squared error (MSE) y el score de confianza del modelo basado en R^2.

```
Coefficient:  [[0.24867868]]
Intercept:  [1.57232275]
Predicitons:
Votos: [1234] Rating aproximado: [[3.34242161]]
Votos: [2345] Rating aproximado: [[3.50207941]]
Votos: [6349] Rating aproximado: [[3.74976634]]
Votos: [42368] Rating aproximado: [[4.22178235]]
Votos: [923] Rating aproximado: [[3.27020856]]
Votos: [14] Rating aproximado: [[2.22860004]]
MSE test:  0.3851638684608506
Model score test:  0.44368764635577607
MSE train:  0.38628120257259824
Model score train:  0.43398127083236715
```

Sesgo



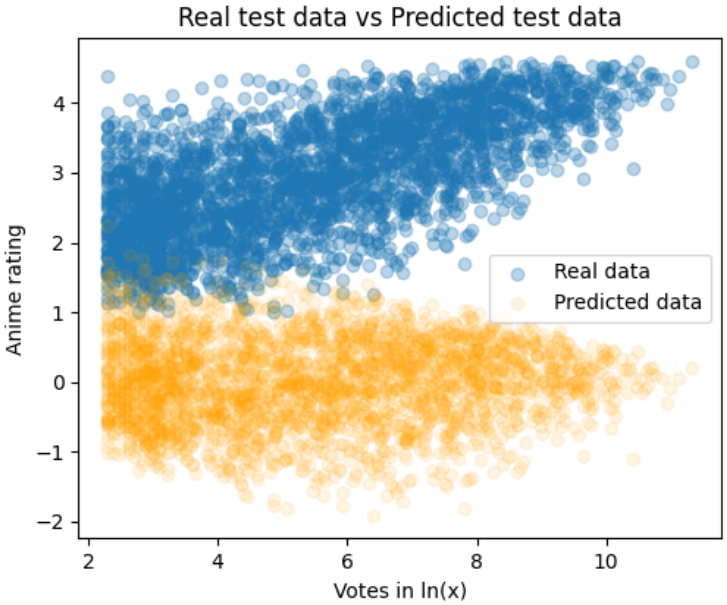


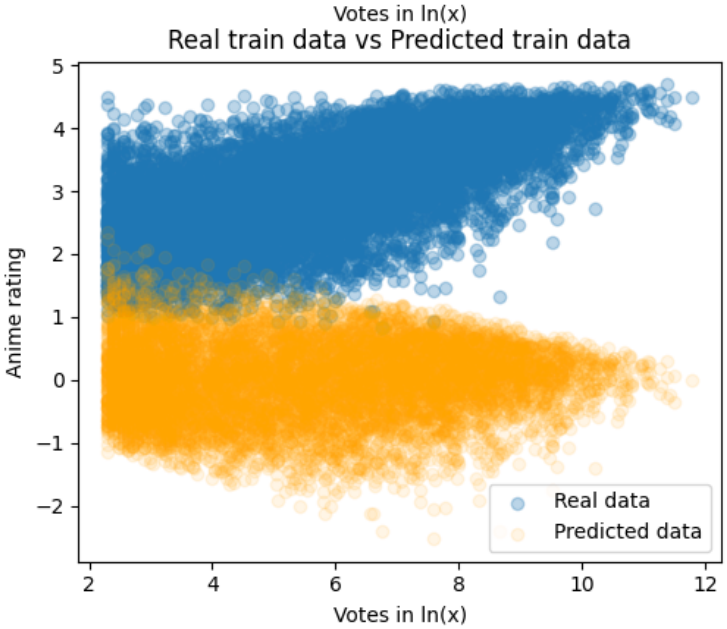
Para la obtención del sesgo de ambos datasets se consideró la obtención del error de la predicción del modelo. Dicho error consta de la diferencia entre el arreglo correspondiente a los datos reales y el arreglo de la predicción; quedando en **EP = Y_train - Y_pred_train** para el training, y **EP = Y_tests - Y_pred** para el testing.

Observando los histogramas resultantes, una ligera distribución descentralizada para los datos del testing con ligero desorden entre valores. Por otro lado, el training muestra un poco más de consistencia entre los datos; pero sigue mostrando una ligera tendencia respecto a su lado derecho.

Por las anteriores razones, el modelo implementado cuenta con un sesgo significativamente alto; por lo que se puede considerar como un modelo con **high bias**.

Varianza





Para la graficación mostrada, se decidió hacer una comparación entre los datos originales, tanto de train como de test, respecto a los datos generados con el error de la predicción empleada para cada uno de los datasets. Si bien esta gráfica emplea datos bastante similares a los del histograma, la representación de los mismos da una idea más clara acerca de la consistencia en el comportamiento de los datos sin importar si se trata de train o de test.

Para ambos datasets, los datos mantienen una tendencia prácticamente similar en términos de extensión y patrones. Inclusive esto da un complemento al hecho de que el modelo cuenta con un sesgo elevado; puesto que los datos reales y los resultantes del error de predicción muestran una notable separación entre sí.

En adición a ello, se buscaron mejoras dentro del modelo empleando el modelo Lasso (Least Absolute Shrinkage and Selection Operator) con scikit learn. Lasso sería empleado para reducir tanto MSE y el score del modelo para cuando se presenta una alta varianza.

```
model_lasso = Lasso(alpha = 0.01)
model_lasso.fit(X_train, Y_train)
pred_train_lasso = model_lasso.predict(X_train)

print("MSE in Lasso train: ", mean_squared_error(Y_train, pred_train_lasso))
print("Lasso score train: ", r2_score(Y_train, pred_train_lasso))

pred_test_lasso = model_lasso.predict(X_test)
print("MSE in Lasso test: ", mean_squared_error(Y_test, pred_test_lasso))
print("Lasso score test: ", r2_score(Y_test, pred_test_lasso))
```

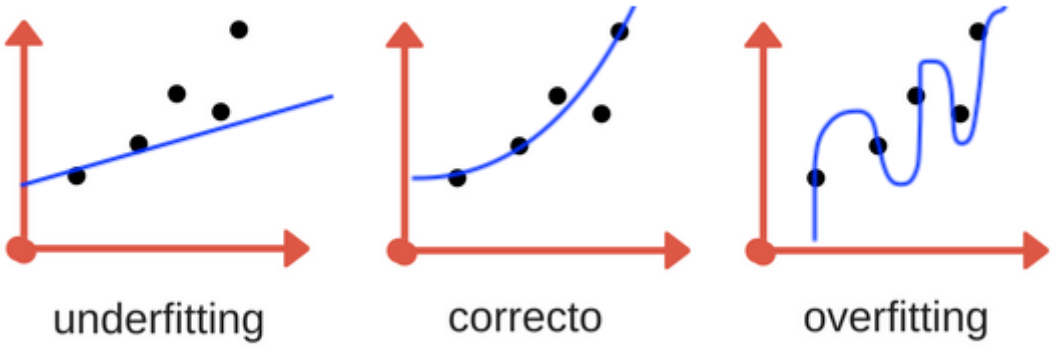
En comparación con el modelo de regresión lineal original, los coeficientes de MSE y score con Lasso no mejoraron; incluso en algunos casos puede llegar a ser menor tanto en test como en train.

```
MSE test:  0.4006293401815206
Model score test:  0.41004751930846495
MSE train:  0.38112697597933376
Model score train:  0.4451208429373934
MSE in Lasso train:  0.3811477455429248
Lasso score train:  0.44509060472633954
MSE in Lasso test:  0.40051335061002863
Lasso score test:  0.4102183214154824
```

En resumen, al repetirse patrones en las gráficas de train y data; y la nula mejoría con la implementación del modelo Lasso comprueban que se tiene una varianza baja o **low variance**.

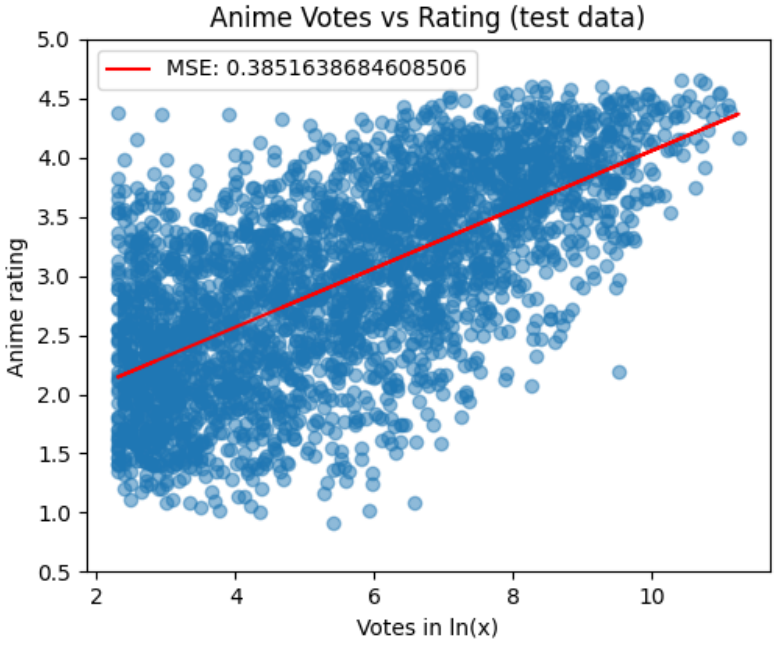
Ajuste del modelo

Al ser un modelo con **high bias** y **low variance**, se concluye que el modelo de regresión lineal presentado para el data set en cuestión se encuentra en una situación de **underfitting**; lo que significa que el modelo no es lo suficientemente sofisticado como para ser confiable ante la cantidad de datos empleado; lo que desemboca en un entrenamiento deficiente. Por naturaleza, el modelo de regresión lineal corresponde a un modelo de underfitting; que se denota más deficiente conforme aumenta la cantidad de datos distribuidos de manera poco óptima para la línea de regresión. Un ejemplo corto de underfitting, fitting y overfitting se muestra a continuación.



Áreas de mejora

En términos del modelo, definitivamente el modelo de regresión lineal no parece ser una solución idónea entre las dos variables seleccionadas. Si bien el MSE presenta valores bajos y aceptables; del mismo modo lo presenta el porcentaje de confianza del modelo con un 40%-45%.



Debido a la densidad de datos empleados del dataset, una regresión lineal siempre tendrá underfitting con regresión lineal simple. Por lo que valdría la pena buscar más opciones de modelos que se ajusten de mejor manera a la proporción entre votos y calificación de un anime; probablemente mediante la inclusión de más variables dependientes para buscar una regresión múltiple o buscar un enfoque hacia una regresión logística.

Conclusión

El dataset me pareció sumamente interesante, en adición al tema que abarca, en términos de la gran variedad de datos que almacena: datos numéricos, fechas, booleanos, descripciones, tiempo, etc. Esta variedad abre la posibilidad de explorar más modelos de machine learning a partir de dicho dataset.

Las librerías empleadas en Python para la extracción, limpieza y modelación de los datos tuvieron un muy buen desempeño, la sintaxis y su uso hace que sea más fácil comprender los puntos importantes de machine learning. En la actividad previa, se hizo la implementación de un modelo de machine learning basado en regresión lineal pero sin el uso de librerías de ciencia de datos como scikit learn; por esta razón, se pudo realizar esta actividad con una comprensión más profunda acerca del funcionamiento de las librerías.

Reconozco que el modelo, al ser muy sencillo, pudo no haber cumplido en su desempeño respecto a los datos; pero lo importante es reconocer las causas principales del underfitting presentado por medio de algunos de los conceptos vistos en clase y así, eventualmente, poder aplicar modelos más acertados en algún futuro.

Como conclusión personal, el desarrollo de esta actividad fue de muchísima utilidad para poder comprender lo visto en clase. Sin duda me llevo una muy buena experiencia de introducción al machine learning y estoy muy emocionado por aprender más.

Referencias

- https://pandas.pydata.org/docs/index.html
- https://numpy.org/
- https://matplotlib.org/
- https://scikit-learn.org/stable/