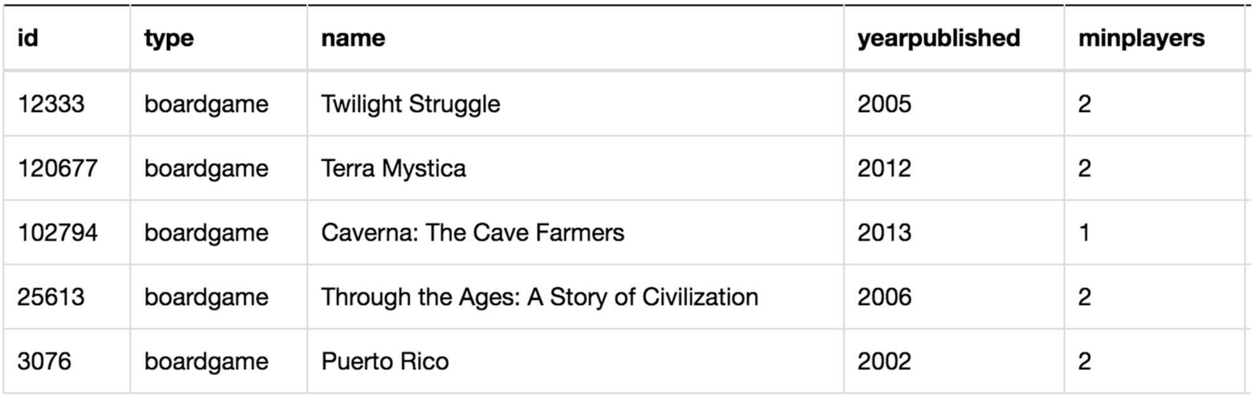
IE 7374 ST. Fall-2020

Regression Analysis:

Board games have been making a comeback lately, and deeper, more strategic boardgames, like Settlers of Catan have become hugely popular. In this use case, you'll be working with a dataset that contains 8000 board games and their associated review scores. The data was scraped from BoardGameGeek and stored in board_games.csv

Here's a preview of the first 5 rows and columns:



id	type	name	yearpublished	minplayers
12333	boardgame	Twilight Struggle	2005	2
120677	boardgame	Terra Mystica	2012	2
102794	boardgame	Caverna: The Cave Farmers	2013	1
25613	boardgame	Through the Ages: A Story of Civilization	2006	2
3076	boardgame	Puerto Rico	2002	2

Each row represents a single board game, and has descriptive statistics about the board game, as well as review information. Here are some of the interesting columns:

name -- name of the board game.
playing_time -- the playing time (given by the manufacturer).
min_playtime -- the minimum playing time (given by the manufacturer).
max_playtime -- the maximum playing time (given by the manufacturer).
min_age -- the minimum recommended age to play.
users_rated -- the number of users who rated the game.
average_rating -- the average rating given to the game by users. (0-10)
total_weights -- Number of weights given by users. Weight is a subjective measure that is made up by BoardGameGeek. It's how "deep" or involved a game is. Here's a full explanation.
average_weight -- the average of all the subjective weights (0-5).

One interesting machine learning task might be to predict average_rating using the other columns. The dataset contains quite a few missing values, and rows where there are no reviews, where the score is 0. You'll need to remove these as you explore the data to make prediction easier.

- Read board_games.csv into a Dataframe called board_games using the Pandas library.
- Print out the first few rows of board_games and look closely at the data.
- Use the dropna Dataframe method with the axis argument set to 0 to remove any rows that contain missing values.
- Remove any rows in board games where users rated equals 0. This will remove any rows that have no reviews

You want to predict the average_rating column using the other columns, but you'll need to do some data exploration before you're ready to do so. The exploration will help you understand the distribution of average_rating better, as well as select an error metric that you'll use to evaluate the performance of your machine learning model.

- Create a histogram of the average_rating column using the hist function.
- Calculate the standard deviation of the average_rating column and print it out.
- Calculate the mean of the average_rating column and print it out.
- Think about what error metric might make sense for this data and write a markdown cell with your thoughts.
- List out the reason of picking this error and why?

Now that you're done some data exploration, you can figure out which columns correlate well with average_rating. This will enable you to remove columns that don't add much predictive power to the model. Columns that are uncorrelated with the target won't help a linear regression model, which is what you'll be using. It will also enable you to remove columns that are derived from the target, or otherwise cause overfitting.

- Use the corr method on numeric_columns to compute correlations between columns. Assign the result to correlations.
- Print out the average_rating column of correlations. This shows how much each column in numeric_columns are correlated with average_rating.
- Do any of the correlations surprise you? Write up your thoughts in a markdown cell.
- Figure out which columns, if any, you want to remove.
- Make Insights through correlation plots, what do you observe from data?

Now that you're done exploring the data, you're ready to create a linear regression model and make predictions for newly created board games. Split the data into training and testing sets, train the algorithm on the training set, and test its performance on the test set.

You'll fit a linear regression model to board_games, using the columns you think should be predictors, and average_rating as the target. You'll then generate predictions using the same predictors you used in the fitting process.

- Initialize a Linear Regression model, and assign it to the variable reg.
- Use the fit method on reg to fit the model using the columns of board_games that you think should be used as predictors, and the average_rating column of board_games as the target.
    - Make sure not to include average_rating in the predictors.
- Use the predict method to make predictions using the columns of board_games that you think should be used as predictors.
    - The predictors you pass into predict should be the same predictors you passed into fit.
    - Assign the result to predictions.
    - Calculate the error metric you chose.
    - Write up what the error value is?

Hyper Parameters: You can try different hyper parameters like

- splitting the data to train (70%) and test (30%) using sklearn.model_selection.**train_test_split()**
- perform 5- cross validation to analyze the model performance using sklearn.model_selection.**KFold**(). Try predicting model using k-folds and without k-folds and see which performs better for model training.
- Different modelling techniques can be Ordinary least squares method using statsmodels.api.**OLS()** and gradient descent method using sklearn.linear_model.**SGDRegressor()** of linear regression.
- Try different learning rates ranging from 0.001 to 0.1, tolerance values of 0.5,0.1,0.01, Maximum iterations should not exceed 50000 and random_state = 02120
- Use LassoCV() for l1 regularization with default parameters and fit the model. Note rmse.
- Use RidgeCV() from linear_model in order to perform l2 regularization. Note and compare rmse. Which one do you prefer?

Bonus: You can try Random Forest Regressor using RandomForestRegressor from sklearn.ensembles and Neural Networks MLPRegressor from sklearn.neural_networks and compare both the results. Which one is best? justify