Kaggle Competition – Independent Study Class

CS 406 – PROFESSOR ROB HESS

CONNOR BENTZLEY – BENTZLEC@OREGONSTATE.EDU

Introduction

Kaggle is an influential online learning platform that hosts machine learning and data science competitions. Founded in 2010, the website hosts a variety of competitions that allow programming enthusiasts to put their analytical and predictive skills to the test against real-world problems. These competitions can cover various domains, from healthcare to computer vision, and be hosted by entities ranging from corporations to international research firms. Each competition supplies the necessary data sets, instructions, and resources to build a machine learning model on a local machine or a Jupyter notebook. All competitions have leaderboards, and some even have cash prizes. By providing access to valuable resources and exciting ways to learn Data Science, Kaggle has become a go-to platform for aspiring machine learning practitioners seeking to advance their skills.

Methods

I began each competition by thoroughly reading the details of the competition to gain a firm understanding of the purpose of the competition, the evaluation criteria, the available data, and any specific requirements or constraints outlined by the competition organizers. My next step was to look at the data. I would open the dataset in the Kaggle browser to see the type of inputs and number of parameters. Then, I would create my Jupyter/Kaggle Notebook and use Pandas to print the shape (if not stated on the competition page), count nulls/invalid values, and do a pre-scoping of the data. Next, I would go onto the discussions tab and read the different approaches that people took. This would help me foresee any potential issues, as many take to the discussion board when they are debugging. I also found a helpful strategy was to control + f and search for "%," because those are posts done by people who have completed a competition submission. All

of these are steps I would take before beginning writing the code. The following sections are my experiences entering the Kaggle competitions.

Competition 1 – Natural Language Processing with Disaster Tweets

Background

One of the ongoing competitions I entered on Kaggle is centered around the analysis of Twitter data in times of emergencies. While primarily a social media platform, Twitter has emerged as a crucial communication channel for individuals to report and receive news in real-time. Because of this, there is growing interest among various organizations, including disaster relief agencies and news outlets, in programmatically monitoring Twitter to identify and respond to potential disasters. However, issues have been encountered with distinguishing disaster-related tweets from those that use such language in a metaphorical context. The example given on the competition page is a tweet with a photo of a sunset with the caption, "On plus side LOOK AT THE SKY LAST NIGHT IT WAS ABLAZE," aside a photograph of a beautiful sunset. To address this issue, Kaggle has supplied data for users to build machine learning models to correctly determine if a tweet is genuinely disaster-related or not.

Code

I first began by using the same libraries that I used in a past assignment of a Machine Learning course I took at Oregon State University. The code I wrote is heavily inspired by a discussion post on Kaggle titled Implementation of Naïve Bayes from scratch, as well as the aforementioned skeleton code from the ML course. Following the imports, I read in the data and

begin cleansing the data. I pre-processed the data by removing noise such as casing, links, weird punctation, new line characters, and such. I attempted tokenizing the strings using multiple different techniques. I started with using whitespaces to split, but I ultimately used a recommendation from the forums to use a simple regular expression. The result looked something like this:

```
0    [our, deeds, are, the, reason, of, this, earth...
1        [forest, fire, near, la, ronge, sask, canada]
2    [all, residents, asked, to, shelter, in, place...
3    [people, receive, wildfires, evacuation, order...
4    [just, got, sent, this, photo, from, ruby, ala...
Name: text, dtype: object
```

Afterwords, I removed stop words, which are words in the string that have little semantic meaning. These words are noise and provide little context. Without these words, the input ended up looking like this:

	id	keyword	location	text	target
0	1	NaN	NaN	[deeds, reason, earthquake, may, allah, forgiv...	1
1	4	NaN	NaN	[forest, fire, near, la, ronge, sask, canada]	1
2	5	NaN	NaN	[residents, asked, shelter, place, notified, o...	1
3	6	NaN	NaN	[people, receive, wildfires, evacuation, order...	1
4	7	NaN	NaN	[got, sent, photo, ruby, alaska, smoke, wildfi...	1

Result & Conclusion

I achieved a prediction accuracy of 0.79528 and a leaderboard rank of 545. I looked at the top of the leaderboard and I saw some users had achieved an accuracy of 1.0. Some of these users had public discussions posts and public code relating to the competition, so I decided to review one of the top spot's strategies. I decided to review the #1 ranked person because they had a lot of

publicly-viewable resources. A notable thing they did different from me was they did more pre-processing of the text data, such as removing emojis, iOS flags, and hashtag marks. In the end, they ranked each tuple in the dataset as either relevant or irrelevant without using a library like XGBoost.

My main takeaway was learning Natural Language Processing and concepts such as TF-IDF (Term Frequency-Inverse Document Frequency). TF-IDF is a common technique in NLP that makes text documents into numerical vectors. The importance of each word is represented by a weight which is assigned by frequency and rarity. I also became more familiar with the XGBoost library and how to score the vectors using it.

Competition 2 – Digit Recognizer

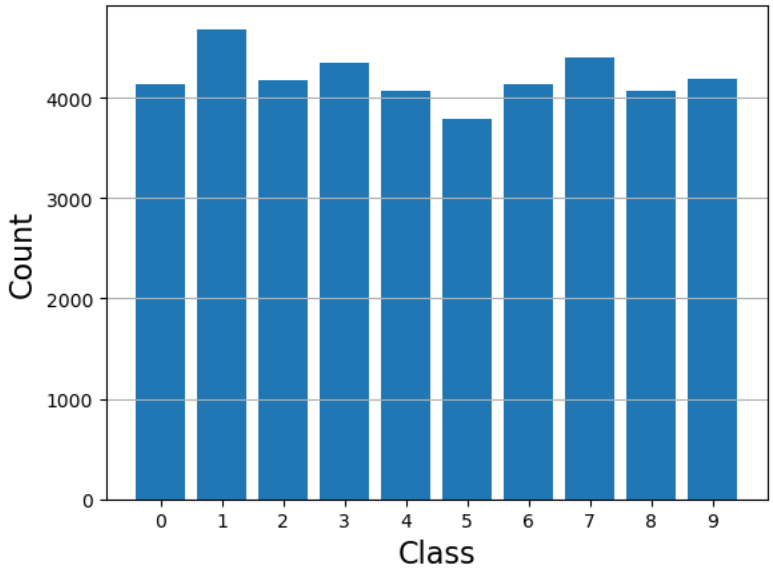
Background

The main task of this competition is to predict the actual digit when given an image of a grayscale handwritten number (from 0 to 9). This competition uses a famous dataset called MNIST, which is a famous institute that provides resources for computer vision projects. It has served as a benchmark for computer vision challenges since 1999. The ultimate goal is to correctly identify the digits in a massive dataset and use those predictions to submit against a hidden dataset on the leaderboard. I plan on using Support Vector Machines (SVM) and Convolution Networks (CNN), as recommended in the competition's description. SVM is an algorithm that relies on finding the best separating line between different classes. It considers the support vectors, which are the closest examples, and builds a decision boundary to distinguish each class. CNN uses interconnected layers of "neurons" to perform non-linear operations to

capture the critical features in the images. CNN is capable of interpreting complex patterns, so I will use that primarily.

Code

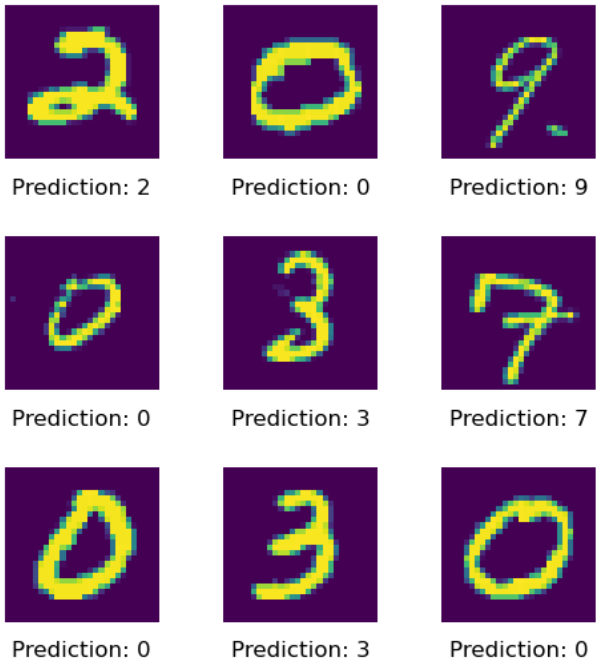
My code is heavily based on a similar project I had in my CS 434 ML class. The project was also based on the handwritten numbers from the MNIST dataset, but instead of using PyTorch, I used Torch. I also used a lot of the matplotlib code from that project to visualize predictions. Starting from the top of the file, I import the aforementioned libraries and load the test and training data. Next, I define the CNN model for 2D image data and use ReLU as my linear activation function. I also set some conditions as per recommended on the discussion boards to set a dropout layer to prevent overfitting. I then define the forward pass. Using the functions, I train the model. I also made a graph to show the predictions on the training data (drawn from CS434 code).



Conclusion & Result

I achieved a prediction accuracy of 0.98925 on the hidden leaderboard dataset and placed 582nd on the leaderboard. This accuracy is lower than the 0.99996 that I was able to achieve on the ML

class assignment I did based on the same training and test data. I speculate that either the dataset is either a different portion of the MNIST is different, or that my usage of Torch may have interfered. The most likely possibility is that since I ran the ML class's assignment locally on my own hardware, the professor must have shrunk the dataset, so I may have had a smaller training and test dataset for that project.



Competition 3 – Advanced Regression Techniques on House Prices

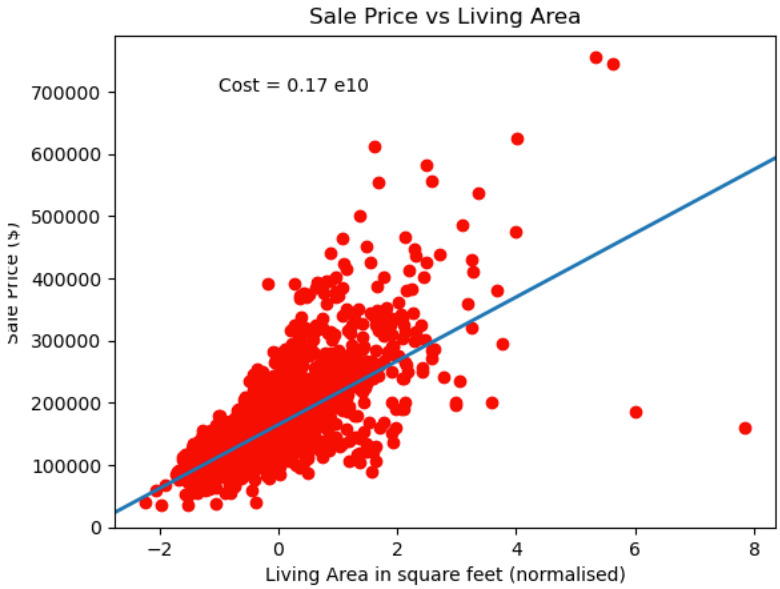
Background

The purpose of this competition is to predict the price of houses based on 79 descriptive variables of the homes. The variables pertain to details such as square footage, number of floors, bathrooms, architectural style, and more. These variables are typically represented by integers or short strings. The dataset is based on real data on houses in Ames, Iowa. The competition description recommends using regression techniques and gradient descent. The approach I plan

on using is linear regression, which involves finding a line that best represents the relationship between the variables and house prices. I will combine this with gradient descent, which is an optimization algorithm that iteratively improves the fit of the line by minimizing error. Gradient descent uses the slope/derivate of a function, and “steps” in the direction of steepest descent to determine the minimum of the function.

Code

First, I import the necessary libraries. I used numpy, pandas, and matplotlib. Using pandas, I read in the data. I then set the parameters for gradient descent, such as the initial numbers, the step size, and number of steps. I opted for a step size of 0.01 because this is what step size I have typically used in the past. Going higher than 0.01 gave me lower accuracy numbers. After doing gradient descent for 2000 iterations, I find my gradient descent number and use it to fit the line representing the relationship between Sale Price and Living Area, as living area seemed to be the best influencer of sales price. I used cool animation code from a previous project, and this is how the fitted line turned out:



Result & Conclusion

I ultimately achieved a score of 0.17312 and a leaderboard rank of 3635. While this score and leaderboard rank sounds very bad, the score works in reverse. The score is representative of the sale predictions gotten wrong, so 0.0 is the ideal score. My low leaderboard rank can likely be explained by this competition being one of the longest on Kaggle. However, I think I could improve significantly on my model. Firstly, just by observing the graph, we can see that linear regression may not be the best way to fit a prediction line. Based on the distribution of the points in relation to the line, a nonlinear model would likely provide a better fit. Another possible way of improving my model would be to do some pre-processing of the data, and ranking the 23 variables by their relevance to the house price.