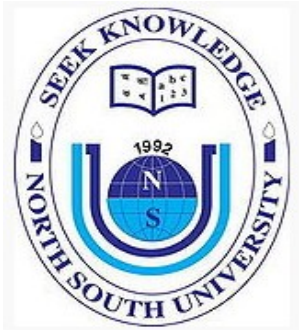
Project Report

CSE465.2 - Deep Learning

Project Title: Concrete Strength Prediction



Submitted By
A. S. M. Sabiqul Hassan
NSU ID - 1812442042
sabiqul.hassan@northsouth.edu

Submitted To
Dr. Mohammad Ashrafuzzaman Khan (Azk)
Assistant Professor

ELECTRICAL AND COMPUTER ENGINEERING
NORTH SOUTH UNIVERSITY
Spring 2021

Table of Contents

Abstract

People need building for every sector like living, trade, education, etc. We have to make sure our building is made with maintaining proper quality otherwise there can be accidents. To reduce the rate of accidents in building to tried to make a deep learning model which will help us to predict the strength of concrete using different features of that concrete region.

Keywords

Concrete, Strength, Prediction, Deep Learning, Train, Test and Dev Dataset

Introduction

The construction section plays an important role in every country. Most of the time we face accidents like earthquakes, fractures on buildings, etc. due to the low quality of concrete strength. The project is desired to check the strength of concrete on different stages by deep learning model which performs well on a large dataset.

Background

This dataset was collected from Kaggle and it was designed from a random source, there was not much information about previous work on the concrete strength section.

Design Methodology

This report utilized a methodology to predict the strength of concrete of a construction site. After collecting the dataset, we have pre-processed it and prepared to implement deep learning models (ANN). We used the PyTorch framework and some libraries like pandas, matplotlib, sklearn, etc. for the coding part.

Project Workflow

The workflow below describes the total process to make our desired machine learning model.

Dataset collection

↓

Preprocess the dataset

- features selection
- categorical data handling
- missing numeric value handling

↓

Dataset split into train and test

↓

Created ANN model

↓

Applied the model on the Train dataset

↓

Applied the model on Test (Dev) dataset

↓

Observed the performance of the model

Dataset

The dataset has been collected from Kaggle[1]. It was created from an anonymous source to make the students familiar with the deep learning project. The dataset was in a single CSV file with about 1k+ instances.

Data Cleaning and Preprocessing

Features Selection

There were a total of 8 features and 1 target before preprocessing our dataset. As we know we there is no need to do feature selection for a deep learning project and it was an automatic process.

The table below represents the features including the label of our dataset that we have used:

Feature Name	Type
cement	float
blast furnace slag	float
fly ash	float
water	float
superplasticizer	float
coarse aggregate	float
fine aggregate	float
age	int
strength	float

`

Categorical Data Handling

In our dataset, there was no categorical data. But for our requirement to draw a pair plot later, we converted the strength column into a boolean field when the strength amount is more than 50% it would be strong (1 – true) otherwise it would not be strong (0 – false).

Missing and Null IntegerData Handling

In our dataset, there were no missing or null values. The code is attached with the Github link.

Dataset Split Part
As we know for the deep learning project, we divide our dataset into 3 parts: train, dev, and test. But our training dataset was comparatively smaller (only 1k+ instances), we used the same dataset for both train and test (dev actually) purposes. The train part contains 80% and the test part contains 20% of the total dataset.
As our dataset was an np array of numbers we applied ANN with the PyTorch framework.

Artificial Neural Network (ANN) [2]

Artificial Neural Network (ANN) simply called Neural Networks (NN) is a computing system inspired by biological neural networks. An ANN is divided into 3 parts mainly: input layer, hidden layers, output layer.

Each layer is connected with the next layer. The hidden layers are not fixed; they can be different according to project requirements. Each layer contains one or multiple neurons which are units of a hidden layer.

The hidden layers are used to do automatic feature engineering on a dataset. They convert the features into tensors then a transition function is applied to them to convert their behavior from linear to non-linear.

Let, $y = w1x1 + w2x2 + \ldots\ldots\ldots + wnxn$
Where, the set of weights, $W = \{w1, w2, \ldots.., wn\}$
And the set of input features, $X = \{x1, x2, \ldots\ldots\ldots, xn\}$

The connections are called edges which have a weight (w). The weights (w) and bias (b) work as controlling noob for the transition function of our deep learning models. Finally, the last hidden layer connects with the output layer and provides a predicted result.

There are some other terms like hyperparameters, parameters, cost function, optimizer, etc which help the model to give better performance in prediction.

Coding Part

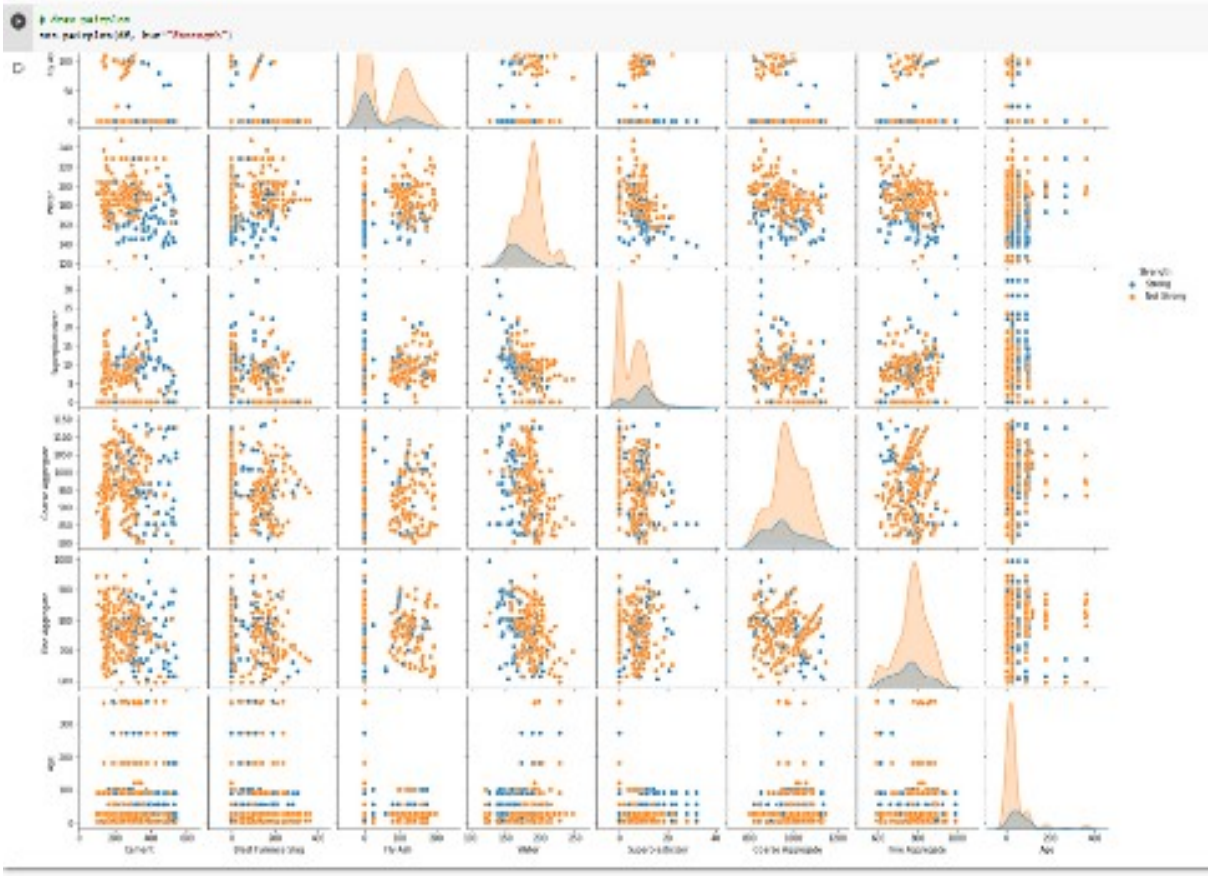
We followed the following steps for the coding part of the project.

- loaded the dataset with pandas and kept in a data frame
- checked the categorical data, missing or null value exists in the dataset
- plotted the pair plot of the dataset to check the relevance of each feature with label
- divided the dataset into train and test parts (dev)
- converted np arrays intro tensors using PyTorch
- created our ANN model with 8 input features, 2 hidden layers with 20 neurons and 2 output features,
- defined our loss function (cross-entropy function)
- defined our optimizer (adam optimizer with learning rate = 0.01)

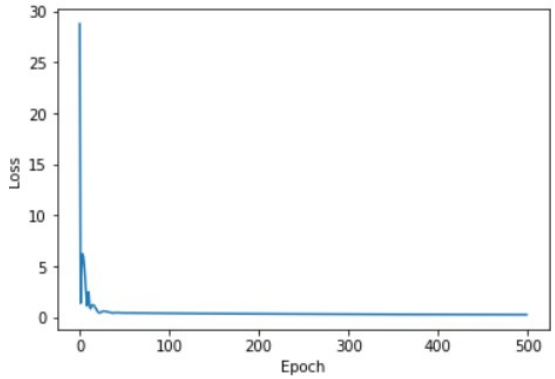
- took random 500 epochs to train our models with train and test data and calculated accuracy, loss, etc
- used confusion matrix to get the accuracy on train and test data
- calculated the loss and accuracy amount

Results & Discussion

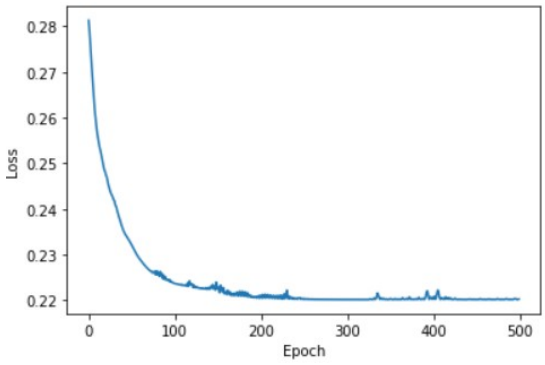
From the pair plot, we can see how each feature of the dataset was related to the label.



Graph of Loss function on Train dataset:



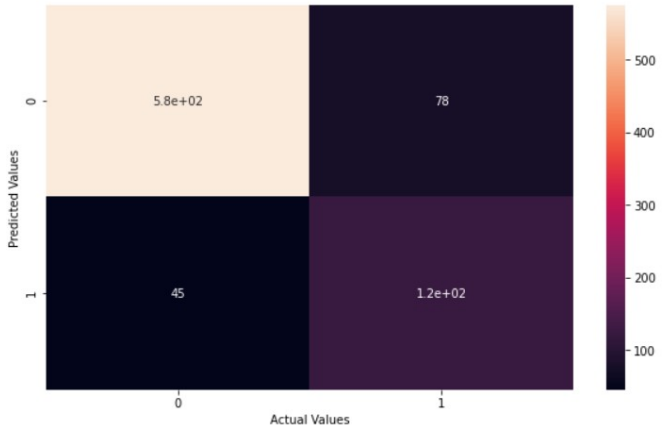
Graph of Loss function on Test (Dev) dataset:



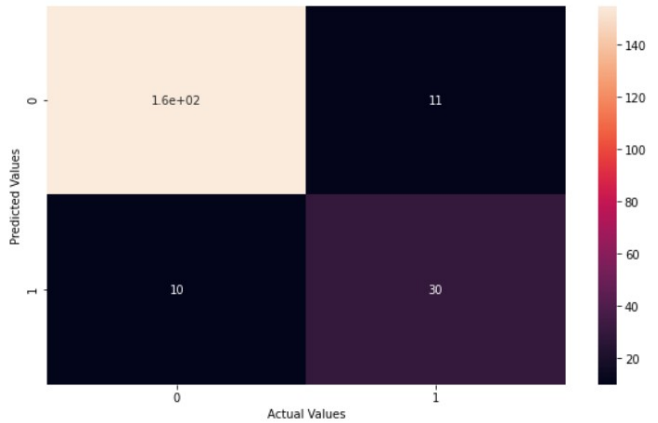
We calculated loss and accuracy on the train and test (dev) dataset from our model after training.

	Train Dataset (%)	Test (Dev) Dataset (%)
Accuracy	85.07	89.81
Loss	5.16	3.52

Graph of Confusion Matrix on Train dataset:



Graph of Confusion Matrix on Test (Dev) dataset:



We calculated the confusion matrix on the train and test (dev) dataset from our model after training.

	Train Dataset	Test (Dev) Dataset
True Positive	1.2e+02	30
True Negative	45	10

False Positive	5.8e+02	1.6e+02
False Negative	78	11

Conclusion & Future Work

The report proposes a methodology to predict the strength of concrete. We can say that accuracy on the train and test (dev) dataset is almost closer and accepted. As we know deep learning model performs better on a large dataset, our data will give better results on more datasets.
Still, now we created a basic ANN model for the given dataset. We can make our model better by using some techniques like tuning hidden layers and neuron numbers, epoch numbers, learning rate, etc. We will try to apply these steps in the future.

Acknowledgment

I would like to give special thanks to two YouTube channel owners, Standford University (for the deep learning course of Andrew Ng) and Krish Naik (for PyTorch coding examples). I used these two channels' content as the helping resource to complete my project.

GitHub Code Link

https://github.com/SabiqulHassan13/CSE465.2-SP2021-PROJECT/blob/main/concrete_strength_prediction_v2.ipynb

References

1. https://www.kaggle.com/prathamtripathi/regression-with-neural-networking
2. https://en.wikipedia.org/wiki/Artificial_neural_network