Final Project White Paper

Name: Victor Chimaobi Offordile
RNo: R11897978

Introduction

This is typically a regression problem. We are getting our dataset from Kaggle - Salary of Data Professionals. Here is a link to the dataset
https://www.kaggle.com/datasets/krishujeniya/salary-prediction-of-data-professions

Below is a description of its columns
FIRST NAME: First name of the data professional (String)
LAST NAME: Last name of the data professional (String)
SEX: Gender of the data professional (String: 'F' for Female, 'M' for Male)
DOJ (Date of Joining): The date when the data professional joined the company (Date in MM/DD/YYYY format)
CURRENT DATE: The current date or the snapshot date of the data (Date in MM/DD/YYYY format)
DESIGNATION: The job role or designation of the data professional (String: e.g., Analyst, Senior Analyst, Manager)
AGE: Age of the data professional (Integer)
SALARY: Annual salary of the data professional (Float)
UNIT: Business unit or department the data professional works in (String: e.g., IT, Finance, Marketing)
LEAVES USED: Number of leaves used by the data professional (Integer)
LEAVES REMAINING: Number of leaves remaining for the data professional (Integer)
RATINGS: Performance ratings of the data professional (Float)
PAST EXP: Past work experience in years before joining the current company (Float)

The goal is to predict the salary of data professionals with the numerical features of the above. Below are the features we are considering;

Feature number	Description
0	Age
1	Unit
2	Leaves Used
3	Leaves Remaining
4	Ratings
5	Past Experience

The target value is the salary of data professionals per annum in dollars.

First, we loaded the csv file and filtered out the rows which has at least one empty value. This is helpful so we don't run into the issue of working with values of type *nan.* We then converted it to a numpy array. The dataset is partitioned into the training set and the testing set. After filtering the data, we have a total of 2631 rows, so we partitioned it into two with the training set having 1315 and the test set having the remaining half.

Analyzing the data, we notice that the scaling is one of the issues we are having. Notice that a change in the *rating* feature will have a greater impact when compared to a change in the *age* feature. Recall from 12.3 of our note, we can apply a Normalization layer to the model's input. This same issue occurs in the target value which is the salary of these data professionals. The typical target values are within the range of 40000-400000. We will scale this by dividing those by 10000.

After the Normalization layer, we now add a densely connected neural network with 4 hidden layers, having 128, 64, 32, 8 nodes, each using the LeakyReLU activation function. Now, the reason for the 4 hidden layers with increased number of nodes in each is to reduce the loss that we will get as the epoch grows in number. The LeakyReLU activation function that was chosen is to account for negative and positive values as well just as we can see from its graph from section 10.5 in our note. We now add a last layer with just one node. This is because we want our target value to be in one dimension.

We note that while compiling the model, we choose the loss function to be the mean square error function. This is because this is a regression problem and the best loss function is the mean squared error function.

We then train the model while reserving the x_train which comprises of the 2 dimensional array with the features that we need to predict the salary just as seen in the table above. We reserve the x_train for validation. This is because we want to see how the model is performing on some data on which it has been trained.

The choice of epoch that we should use is another problem. Guessing the number of epoch can be misleading. The best solution that we can come up with is using a callback function on the *model.fit* function. This callback function will be called after each epoch. This will have access to several important values that describe the performance at the end of each epoch which includes the "loss" and "val_loss". We now create a a callback function that monitors the loss on the validation set and terminates training if there is no improvement for 10 consecutive epochs. The parameter that helps us to set the maximum epoch before termination is known as the patience parameter.

After the fitting, we evaluate the model on the test set. That is, we check how well it is doing on the test set. We then plot two graphs, the first one is the loss on training & validation sets by epoch while the second is the prediction vs actual values on the test set.

Note that there is no need to plot the accuracy graph because this is a regression problem and we don't expect any of the predictions to be exactly correct. Our major goal here is to know how accurate the final model is on the test set which is a decrease in the loss

From the scattered diagram, notice that the most data professionals are earning between $40,000 - $100,000 based on the actual and predicted data. Just few of the data professions are earning between $101,000 - $400,000.

Conclusion: Our major goal is to predict the salary of the data professionals per annum with some of the features that were mentioned via the table above. Now, with the model that we just had, we were able to plot a scatter diagram which has a positive correlation between the predicted and the actual salary. Notice that in the first diagram, our loss is around 1.4 which is good enough as this is the best I could attain by working with so many models.

Reference

Python ML Lecture Note by *Dr. Chris Monico* . Section 12.3 - California housing prices: a regression example.