IML Report

Life_expectancy_task:

Introduction – Briefly describe the dataset and problem.

The problem is to identify the life expectancy of an entry given certain details about that country. Making life expectancy the target feature. The features are along the lines of “years”, “BMI”, “GDP” etc. They are very varied in nature and also have a country attached to them.

The idea is to create a regression model to predict the life expectancy of a new entry, given this data set as training data.

Methodology – Algorithms and preprocessing techniques used.

First, the data was studied by looking at the relations (pearson) of all the features to the target feature and amongst themselves.

This helped identify more important features and possibly features that needed to be combined.

Data transformations:

- Remove rows that do not have “Life expectancy” (it is NaN)
- Status: Convert Developed to +0.5 and developing to -025
    - These decisions were made based on some initial testing and to be honest, eye balling given my very baseline level of research, but they seem to work. How exactly, I do not know, but these seem to work best.
- One hot encode the Country feature
- Take care of missing values by taking the median of the entire column. Other methods were considered, but we know too little about the data set to use those
- Normalize data (expect target) to -1 to 1 range (inclusive), used formula:
    - x_scaled = 2 * (x - min) / (max - min) – 1

Feature selection based on correlation:

- Any features with correlation with target variable less than 0.05 were dropped. These create unnseceary complexity and are not worth considering even.
- Certain pairwise products were taken:
    - For example: 'GDP' * 'Schooling
    - These were taken based on the Pearson relation between these variables if it exceeded a certain amount, it was considered
    - This allows for the capture of more complex, even non-linear relationships

Experimentation – Different model variations and hyperparameters tested.

The first model tested was a simple linear model after the data engineering.

Its metrics:

MSE: 13.923475

MAE: 2.741469

R^2: 0.841091

These are quite good and indicate that my data enginerring was quite helpful.

Hyper parameters used:

Learning rates: [1e-4, 3e-4, 1e-3, 3e-3, 1e-2, 3e-2]

The process of testing these was automated in the script and we found 0.003 to work the best.

EPOCHS = 3000:

We tested 1000 and 4000 and 5000, but those gave very bad R values, and found these to work best, at 5000, the model took too long to run and honestly out of fear I switched back to 3000

PATIENCE = 200

We tested for 100 as well as 250, they worked pretty similarly slight difference, but only somewhat significant.

Hyper parameters for final model (using similar process) decided on:

Learning rate = 0.01

EPOCHS = 4000

PATIENCE = 300

Results – Metrics, tables, and graphs.

Can all be found in the respective results folder

After the success of our first model, we had another look at the data and tried to use a polynomial model (degree = 2):

Metrics:

MSE: 11.034147

MAE: 2.422431

R^2: 0.874067

As you can see, this performed slightly better than the previous one, so we took it as our final.

All the exact same hyper parameter testing was used, but the hyper paramaters we found to work best were different:

EPOCHS = 4000

PATIENCE = 300

Also, we realized that maybe using a polynomial model, the expansion would be too large and we had too many features, so we also added these hyper parameters:
MAX_FEATURES_FOR_EXPANSION = 150

TOP_BASE_FEATURES_FOR_POLY = 80

Explanation – Insights / Explanation of the model in terms of relation between predictor variables and the predicted variable. This will be a function of regularization.

The relationship seems to be fairly accurately represented by a polynomial of degree 2, with rather low error and rather high R^2. From what I looked online, these are rather high for something like life expectancy, as that has several factors that go into it.

We chose not to implement regularization as we felt it unnecessary. We have a pretty good R^2 and decent MSE, even on the training and testing data; indicating that the model is not overfitting. Thus, we felt it make the code unneseairly complicated. From what I have been told, model simplicity is also something to be considered and thus we did not implement regularization.

Challenges – Difficulties faced and how you addressed them.

The main difficulties were in implementation of the code. Although we figured our way through mainly by ramming our heads against the code (I mean going bit by bit and developing the code slowly).

A major difficulty was that I did not know (nor was I told even after asking) what some of the features meant. Some were self-explanatory. But others were not, especially in terms of what they are measured in.

Ultimately, I just had to make educated guesses and some testing to see what worked best.

Laptop_price_task:

Introduction – Briefly describe the dataset and problem.

The target feature is “Laptop Price”. The dataset describes the specifications of laptops and the price I assume they were either sold for or listed as.

Based on the given previous data about specifications and price, we need to make predictions about the price of other laptops, given their specifications.

To do this, our task is to create a regression model of some kind.

Methodology – Algorithms and preprocessing techniques used.

First, the data was studied by looking at the relations (pearson) of all the features to the target feature and amongst themselves.

This helped identify more important features and possibly features that needed to be combined.

Data transformations and engineering:

- Rows with a missing 'Price' are removed
- Missing numerical values are filled using the median of their respective columns, could have take mean but the range and skewness of the data is too high for these columns
- All numerical features are normalized to a range of -1 to 1.

As the data set is very complex to convert into one that is usable by regression, here are a bulk of the transformations and enginerrings used:

- Extracts the X and Y resolution (e.g., 1920 and 1080) into new numerical columns ('x_res', 'y_res')

- Extracting any descriptive text (like "IPS Panel" or "Touchscreen") into a new categorical column, 'ScreenText'
- For “Cpu”:
    - Extracts the CPU frequency in GHz into a new numerical column, 'cpu_ghz'
    - The remaining text (e.g., "Intel Core i5") is kept as a categorical feature, 'cpu_text'. Could have one hot encoded this, but that would lead to an explosion in number of features.
- RAM is converted into a numerical feature representing its size in terabytes
- 'Memory':
    - SSD capacity in terabytes and stores it in a new numerical column, 'ssd_tb'
    - The types of memory (e.g., "SSD", "HDD") are extracted into a new categorical feature, 'MemoryText'
    - 'Weight' Parsing: The weight is extracted in kilograms into a new numerical column, 'weight_kg'

Everything that I had to one hot encode (I tried but there is no way around not one hot encoding these):

- 'Company'
- 'TypeName'
- 'ScreenText'
- 'cpu_text'
- 'MemoryText'
- 'Gpu'
- 'OpSys'

More notes:

Features with an absolute correlation to 'Price' below a threshold of 0.03 are removed.

Creating Interaction Features: New features are created from the pairwise products of the top eight most correlated features.

I am going to be honest, this number was selected at random, if I did any more, the number of features just got too high.

Experimentation – Different model variations and hyperparameters tested.

As a baseline, a linear regression model was implemented:

Metrics:

MSE: 283236244.015153

MAE: 11339.324582

R^2: 0.736251

Notice that the error is rather high, even though the R^2 value is not a bad starting point at all.

We realized maybe because this was due to the price being disproportionately higher than the rest of the variables, especially as a lot were one hot encoded and a lot were categorical.

Thus in the final model, a log was applied to the price

The exact hyper parameter testing was done on this task as well (refer above), the final hyper parameters decided on for this model was (and the ones tested):

Learning rate: 0.003 tested: [1e-4, 3e-4, 1e-3, 3e-3, 1e-2]

EPOCHS = 3000 (tested: in range of 2000-5000)

PATIENCE = 200 (tested in range of 100-250)

Hyper parameters of final model decided:

EPOCHS = 4000 (tested in range of 2000-5000)

PATIENCE = 300 (tested in range of 100-500)

LR_CANDIDATES = 0.0003 tested: [3e-4, 1e-3, 3e-3, 1e-2]

WEIGHT_DECAY = 0.0

POLY_DEGREE = 2

TOP_BASE_FEATURES_FOR_POLY = 80

Metrics of final model:

MSE: 0.105542

MAE: 0.232447

R2: 0.722797

As you can see, the accuracy has gone down very slightly, but the MSE and MAE have increased. So the taking the log was a good idea.

Results – Metrics, tables, and graphs.

They are in the respective results folder

Explanation – Insights / Explanation of the model in terms of relation between predictor variables and the predicted variable. This will be a function of regularization.

In all honesty regualisatiotion was tried, but it did not work. The model was not computing.

Challenges – Difficulties faced and how you addressed them.

The major difficulty was the data engineering. Making the parsing algorithms and implementing the, coming up with them and the sheer amount of one hot encoding was very hard to deal with.