Assignment 2:

Application of Neural Networks

NAME AND SURNAME	STUDENT NUMBER
ZENAN SHANG	SHNZEN001
FRANCOIS DU TOIT	DTTFRA013
HANNAH KVASSHEIM	KVSHAN001

Table of Contents

Problem Formulation

Machine learning problem

This neural network-based AI system aims to predict the job title of an individual working in the Data Science field based on the following factors: work years, experience level, employment type, salary, employee residence, remote ratio, company location and size.

Difficulty

The task of predicting job titles based on the provided factors can be difficult given the 93 possible outputs of the model. The challenge lies in identifying the relationship and patterns between these diverse factors and corresponding job titles. The neural network needs to be carefully designed by tuning the hyperparameters and trained to effectively set the parameters that capture and learn these complex associations.

Usefulness

Building an AI system that accurately predicts job titles can be highly valuable for several parties in the workplace environment. Job seekers can leverage this system to identify potential job titles that align with their qualifications, experience, desired salary and place of residence. Recruiters can benefit from this system by automating the process of matching candidates to appropriate job titles, streamlining the recruitment process and improving efficiency. Although the model is initially adapted to predict data science jobs, it could potentially be redefined or extended to include job predictions from other sectors.

Ethics

This model could also unintentionally influence certain groups of people in a negative way. For example, graduates may undervalue certain jobs and the experience they can get out of them because of the low ranking in our prediction model or having low salary expectations based on historical data. Therefore, this model should mainly be viewed for educational purposes rather than as guidance for job seeking.

The model exhibits a tendency to favor certain job titles based on different types of features, it may generate heightened interest among job seekers towards companies offering these positions. This could lead to an unfair advantage for these companies, creating an imbalance in competitiveness.

Therefore, this model should mainly be viewed for educational purposes rather than as guidance for job seeking.

The Baseline

We use a Linear Regression model to predict the linear relationship between the job title and the group of variables, including work years, experience level, salary in USD, employment type, employee residence, remote ratio, company location and size. We chose linear progression because of 5 factors. The data split for the baseline was 60/20/20 for training, validation and testing in that order.

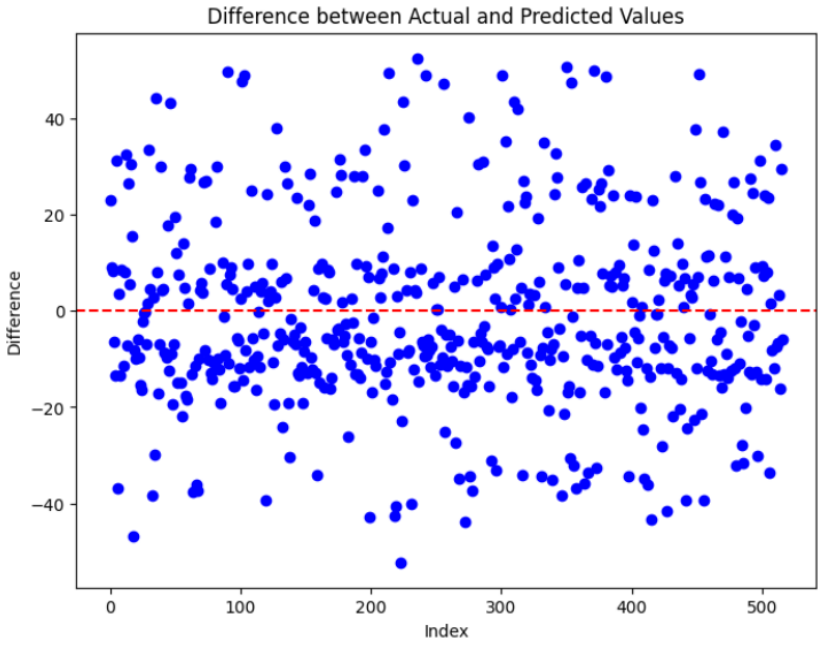
Simplicity: Linear regression is a simple and interpretable model that assumes a linear relationship between the input features and the output variable. It provides a straightforward way to understand the impact of each feature on the predicted job title.

Interpretable coefficients: In linear regression, the coefficients associated with each input feature indicate the strength and direction of their influence on the predicted job title. This can provide insights into which factors are most important in determining job titles.

Baseline performance: Linear regression serves as a baseline model to establish a benchmark performance. It allows us to evaluate the performance of our more complex model against this baseline to assess whether the additional complexity is justified.

Efficiency: Linear regression has low computational complexity. It is computationally efficient, making it suitable for large datasets or situations where real-time predictions are required.

Feature importance: By analyzing the coefficients of the linear regression model, you can identify the most influential features for predicting job titles. This can help you prioritize and focus on the most relevant factors when developing more sophisticated models.



Baseline Data Accuracy:

MSE: 0.8408657606947736

RMSE: 0.9169873285355549

MAE: 0.7207115323652393

R2 Score: 0.04883874546552758

Model Design

Model architecture

The model architecture for this prediction task is a feedforward neural network with one hidden layers, including a reLU activation function and a softmax output. The chosen design is appropriate for several reasons:

- The feedforward neural network introduces non-linearity through the activation functions, allowing the model to capture complex relationships and patterns between the input features and job titles.
- The extra hidden layer gives the network greater representational capacity due to a deeper architecture. It allows the model to learn hierarchical features and abstractions that are beneficial to handle complex patterns in our data.
- The softmax activation function in the output layer is suitable for multi-class classification tasks like job prediction. It assigns probabilities to each possible job title, allowing the model to produce probability distributions over every output. This

makes it easier to interpret the model's predictions and understand the relative likelihood of different job titles for a given set of features.

Input representation

The features included for the job prediction task includes information about the individual and the job context. The following table provides the input representation for the neural network that is based on the Data Science Job Salaries Dataset:

Feature	**Values**
Work experience	Entry-, Mid-, Senior-, and Executive-level
Employment type	Part-time, Full-time, Contract, Freelance
Salary in USD	total gross amount in USD
Employee residence	Country of residence
remote ratio	0,50,100
Company location	Country of main office
Company size (number of employees)	S (<50), M (50-250), L (>250)

The salary and salary currency factors provided in the dataset are not included, since they would create redundant information. The salary in USD already provides a standardized representation, eliminating the need for separate salary and currency features.

Output representation

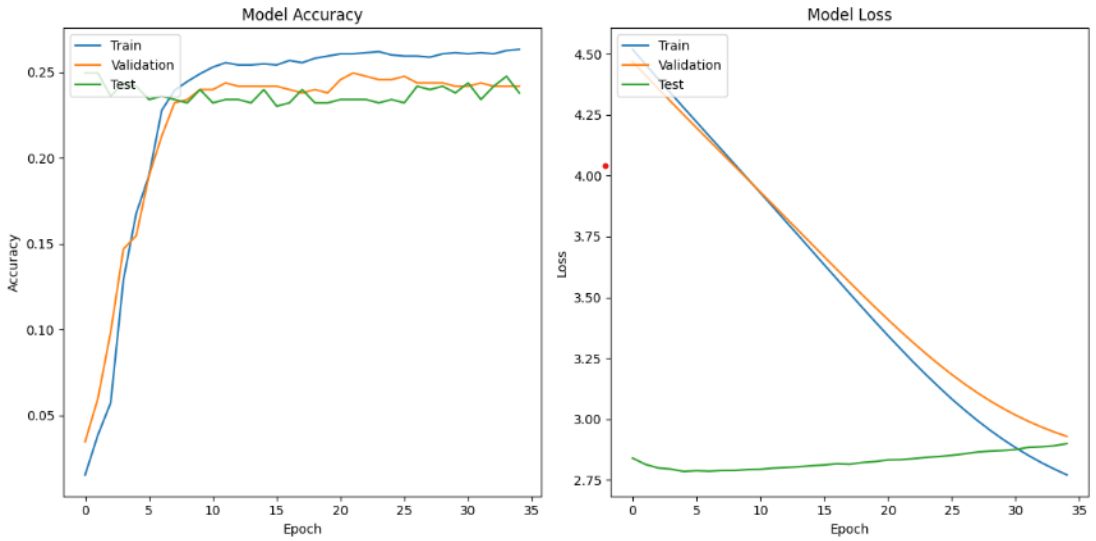
The output will include the predicted job title as a categorical variable in a range of 93 possible categories. The model will output a probability distribution over all possible job title classes. The predicted job title would then correspond to the class with the highest probability.

Model Validation

Hyperparameters

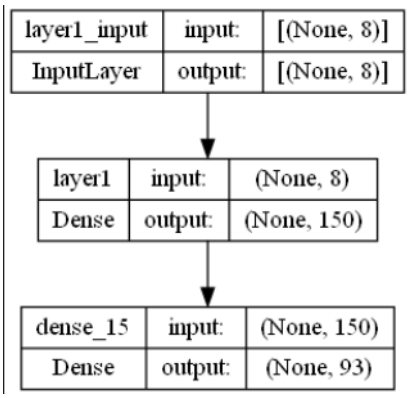
The way we set our epoch, batch size, number of hidden layers, and number of hidden neurons is by manually tuning the different hyperparameters to find the optimal set.

- ReLU: This is a linear function that outputs the input if it is positive; otherwise, it outputs zero. This simple nature makes the neural network easier and faster to train, reducing the exploding gradient problem. Thus, we use ReLU as our activation function for the hidden layer.
- Softmax: We use the Softmax function in our output layer of a classifier. It transforms the inputs to be between 0 and 1, the total sum of the outputs is also equal to one. The predicted value will be the highest value in the array of outputs. This makes it suitable for probability interpretation.
- Number of neurons / Hidden layer: This is a crucial factor in determining the complexity and capacity of our model. They can take one or more inputs and apply a transmission to them to produce an output.
- Batch size / Epochs: Batch size is the number of training examples used in one iteration for updating the model's parameter. Epoch refers to one complete pass through the entire training dataset during training.



This is the result of 1 hidden layer with 150 neurons, 35 epochs and 1000 batch size. The model accuracy and model loss for training, validation and testing are all around the values meaning that there are no signs of overfitting/underfitting.

The diagram below is a visualization of our multi-layered model.



Evaluation

Training/Validation/Test

A 60/20/20 data split is often considered good practice for dividing the dataset into training, validation, and test sets. Allocating 60% of data to training provides a significant portion of data for the model to learn, making it easier to capture patterns and relationships in the data. Leading to better generalization and performance. Allocating 20% for the validation test set ensures a substantial portion for evaluation purposes (hyperparameter tuning) while ensuring a sufficient number of samples in the test set. This results in a potentially reliable estimate of the model's performance.

Evaluation metrics

The selected evaluation metrics for the neutral network are commonly used for multi-class classification tasks. We used the following metrics to provide a standard and effective way to assess the model's performance in predicting job titles across multiple classes:

- Accuracy: measures the percentage of correctly predicted job titles out of the total predictions.

- Precision: measures the proportion of true positive predictions (correctly predicted job titles) out of all positive predictions (true positives and false positives). It indicates the model's ability to identify the positive instance correctly by considering a class as positive and the rest as negative and averaging it out for all classes.
- Recall: measures the proportion of true positive predictions out of all actual positive instances. It indicates the model's ability to identify the positive instances correctly.
- F1-score: is the harmonic mean of precision and recall, providing a balanced measure of both metrics. It considers both precision and recall to evaluate the model's overall performance.

By considering multiple evaluation metrics, we can gain a comprehensive understanding of the model's performance.

Analysis of Model Performance

Referring back to model validation, we can see that we have tested batch size, epoch, number of neurons and number of hidden layers for our model. That is why we have the following parameters:

- Batch size: 1000
- Epoch: 35
- Hidden Layer: 1
- Number of neurons: 150 for the hidden layer

We use the ReLU activation function for our hidden layer and Softmax for our output layer. The reason why we did not use Sigmoid and Tanh in our model is that they can suffer from vanishing gradients due to the large dataset with fairly complicated models. Therefore, we stick to the most popular choice of ReLU for the hidden layer and softmax for the output layer.

The following metrics show the analysis of the final model's performance.

Test loss: 2.939941883087158
Test accuracy: 0.24177949130535126
Precision: 0.12
Recall: 0.21
F1: 0.15

Despite the careful considerations concerning the architecture of this neural network, the low scores obtained in the metrics suggest that the model lacks the ability to accurately predict job tasks. Also, this suggests that the initial assumption regarding the challenging nature of the prediction task was indeed valid.

Shortcomings

Neural network design and approaches often have potential shortcomings that must be considered. In this particular case, our hyperparameters were tuned using a manual search approach involving trial and error on the validation set. While this method allows for iterative adjustment of hyperparameters based on observed performance, it can be time-consuming and may not thoroughly explore the entire hyperparameter space. In hindsight, alternative techniques such as grid search, random search, or automated hyperparameter optimization libraries could have been employed to ensure a more comprehensive search and a more accurate result. It is worth noting that even small changes in hyperparameter values can significantly impact the training process and the final performance of the model. However, it is important to balance the time required for hyperparameter tuning and the available computational resources.