PHASE 4: ADVANCED MODELING AND FEATURE ENGINEERING

1. FEED FORWARD NEURAL NETWORK WITH BOW FEATURES

1.1. Dataset

In advance modeling, first I started with the final BoW based features. The features which we got after applying forward feature selection.

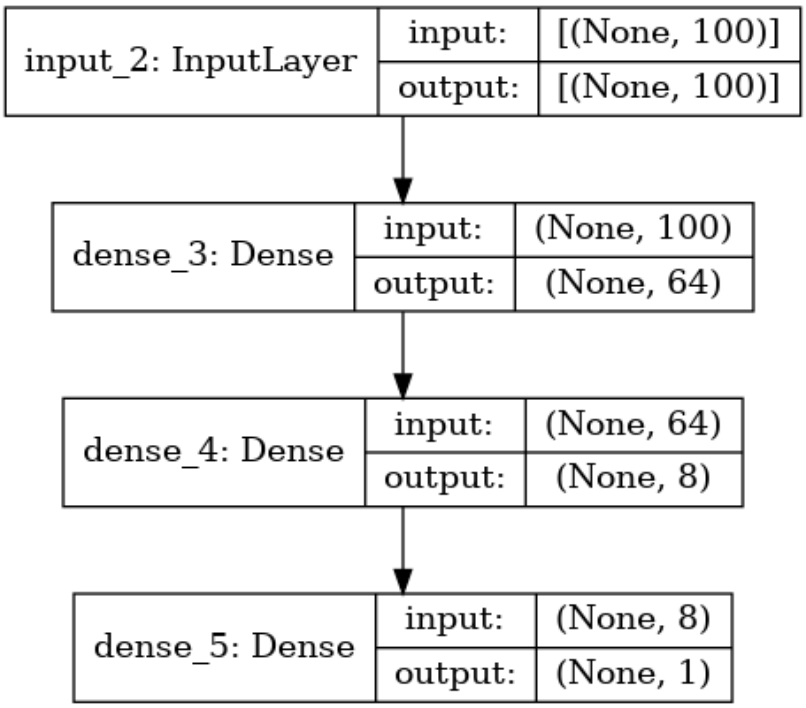
It only has a feature dimension of size 100. That means it only have 100 words from the whole corpus. Which is kind of better than NER (named entity recognition) because we have used the data and model to get these important words.

I have divided the data into three parts: train, cross validation (cv) and test.

- We have a total of 90 data points.
- We have 100 features.
- We have 53 data points for training.
- We have 10 data points for cross validation.
- We have 27 data points for testing.

1.2. Model Architecture

Then I created simple feed forward neural network as below,



After a lot of trials I settled with this architecture. It has a total of 6993 trainable parameters. And all three activation functions have ReLU activation function.

In regression tasks, we generally use linear activation functions. But since this is a match percentage prediction task, where there are no negative values. So it makes sense to use ReLU instead of linear activation function.

1.3. Training and Evaluation

During training I found that the model is overfitting a lot and the MSEs are almost constant for train and cv data. I even tried with a very small learning rate but without any luck.

- Train mse (mean squared error) is 1904.343
- Train mae (mean absolute error) is 40.7598
- CV mse (mean squared error) is 1219.7331
- CV mae (mean absolute error) is 30.2521
- Test mse (mean squared error) is 1800.1959
- Test mae (mean absolute error) is 39.575

From these performance metrics we can conclude that this model is very poor.

2. FEED FORWARD NEURAL NETWORK WITH BOW AND AVERAGE W2V FEATURES

2.1. Dataset

After the poor performance of the previous model. I thought of training a feed forward neural network with final BoW and final Average Word2Vec features.

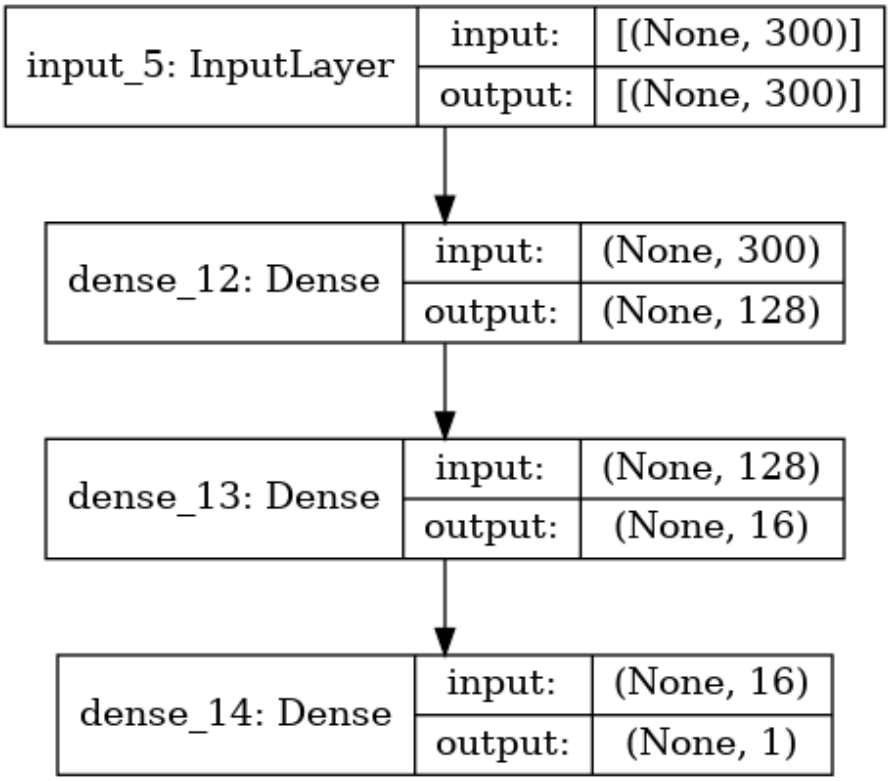
As I mentioned earlier we have 100 features for the final BoW. And for the final average word2vec we had 200 features. Which brings our total feature dimension to 300.

I have divided the data into three parts: train, cross validation (cv) and test.

- We have a total of 90 data points.
- We have 300 features.
- We have 53 data points for training.
- We have 10 data points for cross validation.
- We have 27 data points for testing.

2.2. Model Architecture

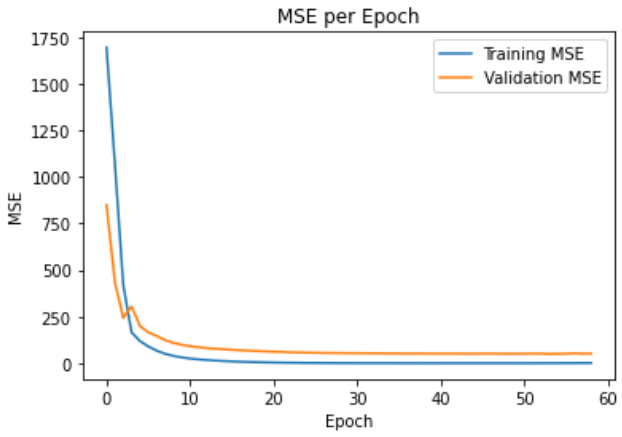
Since now we have more features I decided to use a few more activation functions then before. The model architecture is shown below,



It has a total of 40,609 trainable parameters. And all three layers have ReLU activation functions as before. I had to do quite a bit of tweaking to get the best architecture.

2.3. Training and Evaluation

At first, the model ran for 59 epochs and the performance was way better than the previous model. But it seems that the model is overfitting a lot.



We can see that the error is decreasing very fast but it becomes almost constant after a few epochs. Also the mse and mae of train and cv data is pointing that the model is overfitting.

So I decided to execute it only for 20 epochs. And we got a better result.

- Train mse (mean squared error) is 3.0928
- Train mae (mean absolute error) is 1.3764
- CV mse (mean squared error) is 72.9421
- CV mae (mean absolute error) is 7.0519
- Test mse (mean squared error) is 39.6974
- Test mae (mean absolute error) is 5.3252

This is very impressive. As we can see this model is very comparable to the stacking ensemble model, where the base models or level-0 models are Support Vector Regression (Linear Kernel) model based on BoW features and Linear Regression (L2 Regularized) model based on Word2Vec features and the meta model or level-1 model is a KNN Regressor.

This makes sense because both models are based on the same features.

3. BERT WITH FEED FORWARD NEURAL NETWORK

Till now we have used previous features with simple feed forward neural networks. But BERT (Bidirectional Encoder Representations from Transformers) is a SOTA (state of the art) technique for NLP tasks. It is based on transformer which usage encoder only structure. And as the name indicates it is bi-directional. BERT base can take a maximum of 512 worded input and it outputs 768 dimensional representation for each token.

BERT is trained on huge dataset of google books, wikipedia and other internet crawled datasets. It is trained with Masked Language Modeling task and Next Sentence Prediction task. So naturally it assumes the sequential nature of the text. But here we do not have a sequential data because we have very few sentences in the resumes. So even though it is sota for nlp tasks it should not work.

BERT combined multiple research papers ideas such as,

I) Semi supervised sequence learning -
It does semi-supervised sequence learning with Masked Language Modeling and Next Sentence Prediction tasks.

II) Contextualised word embeddings -
It got the idea of contextualised word embeddings from ELMO paper. Where each word embedding is generated based on the whole sentence.

III) Transfer Learning -
It got the idea of transfer learning from ULM-FiT. We get the pre-trained BERT model and then we can do fine tuning as per our need.

IV) Encoder Only Transformer -
It got this idea from OpenAI transformer which is a decoder only transformer.

Here I have used DistilBERT, which is a simpler and faster version of BERT with a very minimal decrease in performance. I have loaded the distilbert-base-uncased that means the case does not matter.

3.1. Dataset

I have loaded the PDFs and then done encoding with DistilBERT based tokenizer with max length = 300 and zero padding if required. Zero padding is done so that we can perform batch training.

Then using the pretrained model I have extracted the output corresponding to CLS token which will be of 768 dimension. I have done this for both job description and resume.

I have divided the data into three parts: train, cross validation (cv) and test.

- We have a total of 90 data points.
- We have 1536 features. (768+768)
- We have 62 data points for training.
- We have 9 data points for cross validation.
- We have 19 data points for testing.

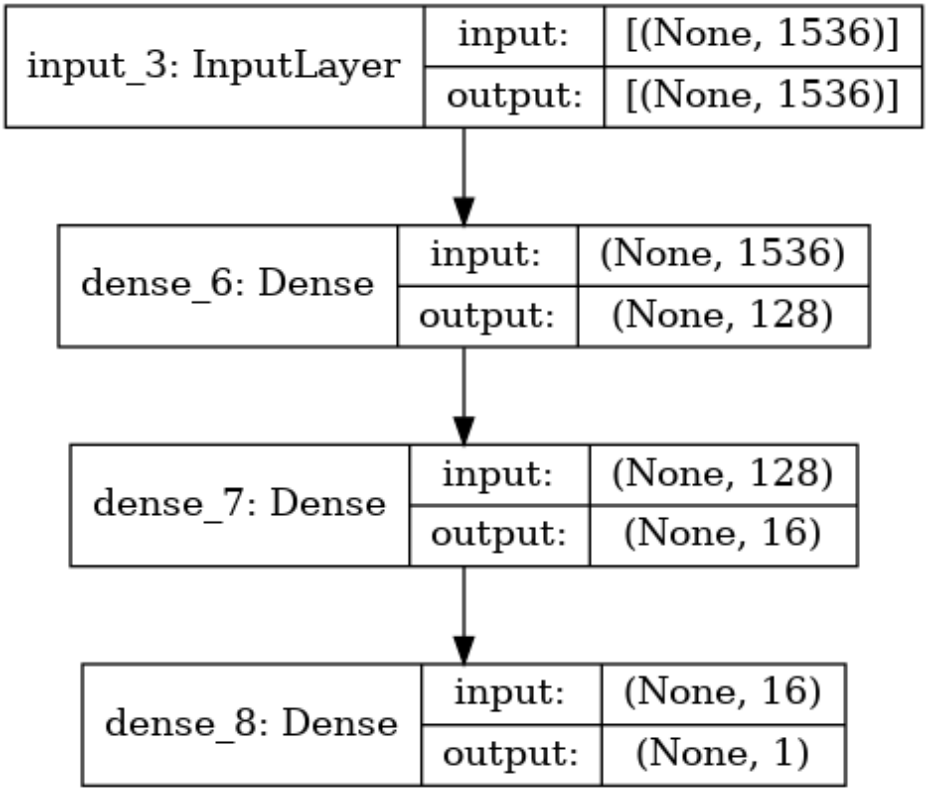
Distribution of output for all three data are almost overlapping. That means the data split is good.

3.2. Model Architecture

Just like before the model has three dense layer and all of them have ReLU activation unit. The total trainable params are 198,817.

In all the models I have used Adam optimiser. Because it is the best optimiser as of now.

The model architecture is as shown below,



3.3. Training and Evaluation

As expected the performance is not that good. Just like the initial model the loss is not decreasing at all.

We get following performances from the model,

- Train mse (mean squared error) is 1780.1329
- Train mae (mean absolute error) is 38.9871
- CV mse (mean squared error) is 1070.5638
- CV mae (mean absolute error) is 27.6278
- Test mse (mean squared error) is 1803.7161
- Test mae (mean absolute error) is 37.4089

4. CONCLUSION

The Feed Forward Neural Network with BoW and Average W2V features model is performing the best but still not as good as the stacking ensemble model.

The main reason for deep learning models to not perform well is because we do not have much data. And also there is no sequence structure even though it is a text data.

This is one of those cases where a simple model performs better than a complex deep learning model.

5. REFERENCES

- AppliedRoots. [https://www.appliedroots.com/]
- Tensorflow API Doc. [https://www.tensorflow.org/api_docs/python/tf]
- Transformers. Hugging Face. [https://huggingface.co/docs/transformers/index]