VARNA FREE UNIVERSITY

Hyperparameter Adjustment in Regression-based Neural Networks for Predicting Support Case Durations

Master Thesis Presentation

Hristo Hristov

Academic supervisor: Assoc Prof. Galina Momcheva

11.04.2020

Outline

1. Background and problem description
2. Challenge
3. Research goal
4. Methodology
5. Summary of Results
6. Conclusion

1. Background and problem description

- Support plays a critical role in the software development process
- Prompt action and response are vital for the perception of the good service
- Often times users are unaware of how long the resolution takes
- Need for a model that makes accurate predictions
- Challenge: **text data of high cardinality**

1. Background and problem description

Regression & Neural Networks

- By combining a regression model with the computational potential of neural network we aim at getting:
  - Powerful model for accurate predictions
  - Flexibility for processing unknown inputs

2. Challenge

2.1. Text data

- Must be represented numerically
- Simple forms of representation tend to produce poor predictions
- The representation must be numerically consistent among the various features
- Feature engineering as a super set of hyperparameter engineering

2. Challenge

2.2. Cardinality

- Several features have high cardinality
  - User id
  - Support desk handler
  - Symptom
- The model must be able to process unique values

2. Challenge

2.3. Case Duration Standardization

- The dependent variable is also standardized
  - Easier for the network to predict
- Min = 0 and max = 341.28 days
- The huge range compromises the model's prediction capacity
  - Could be compensated with more data

2. Challenge

2.4. Overfitting

- We are only using l2 regularizer with a value of 0.001
- Other methods such as l1 or dropout are not used
- All methods have an identical baseline
- Result: the best encoding method can stand out naturally, without specific overfitting counter-measures

3. Research goal

Cardinality

- Compare and contrast different string encoding methods in the context of the presented data set
  - Analyze the results by using established loss metrics
  - Determine the best performing encoding method
  - Explain the differences

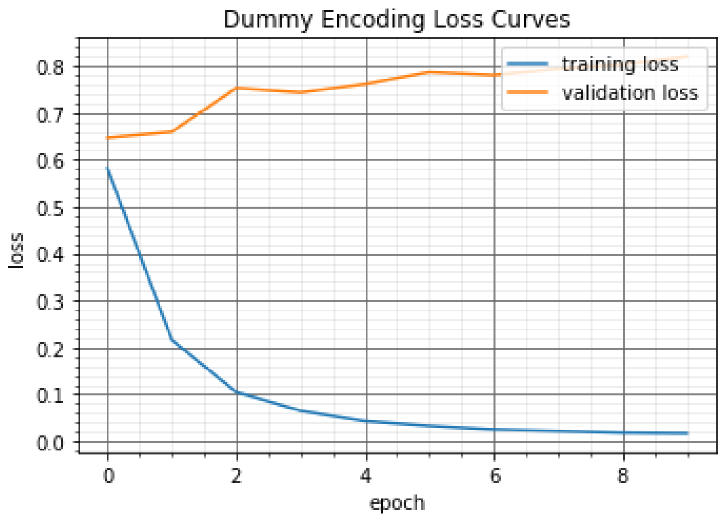
4. Methodology

4.1. One-hot encoder

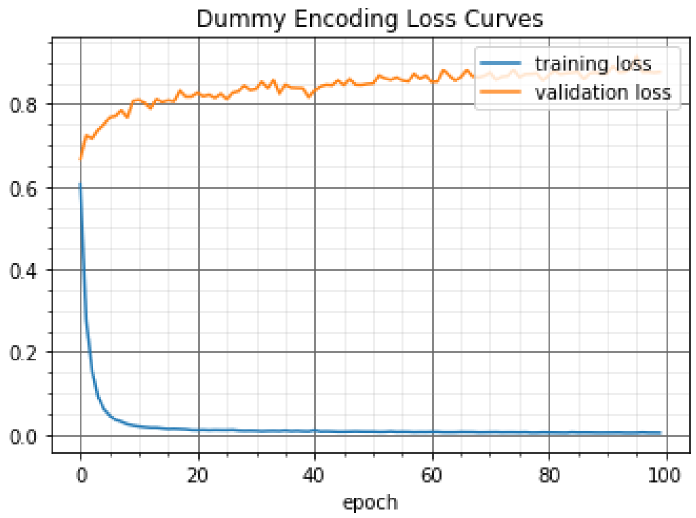
- The most simple encoding method
- All unique values are converted into features
- The new feature values are 0 or 1
- No explicit hyperparameter
  - Possible to artificially adjust cardinality

4. Methodology

4.1. One-hot encoder



	Train loss	Val loss	Test loss
10	0.0177	0.7949	1.0902
100	0.0042	0.8776	1.2711



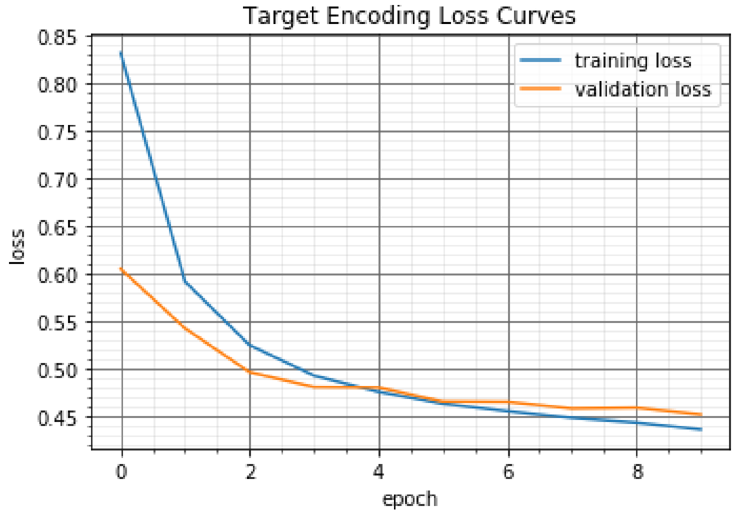
4. Methodology

4.2. Target encoder

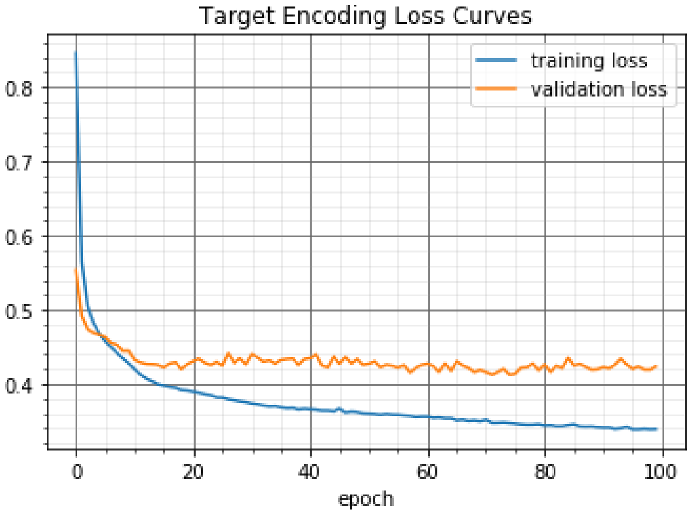
- Calculate an average of the target value for each unique feature value
- Replace with the calculated mean
- Grouping of the cases with common unique features
- Hyperparemeter: weight of overall mean

4. Methodology

4.2. Target encoder



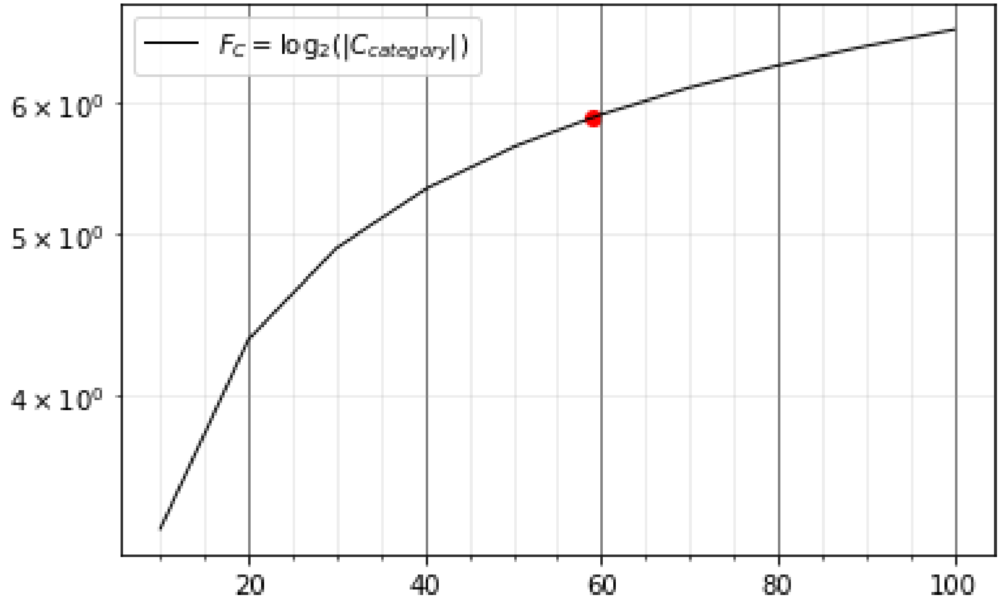
	Train loss	Val loss	Test loss
10	0.4363	0.4520	0.6064
100	0.3390	0.4236	0.6708



4. Methodology

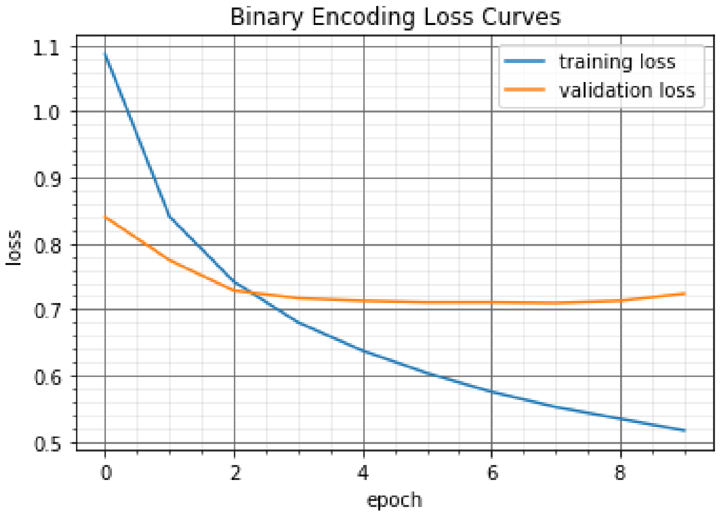
4.3. Binary encoder

- Convert feature ordinal value into binary value
- Dimensionality is increased at a log scale
- No explicit hyperparameter
  - Possible to artificially adjust cardinality

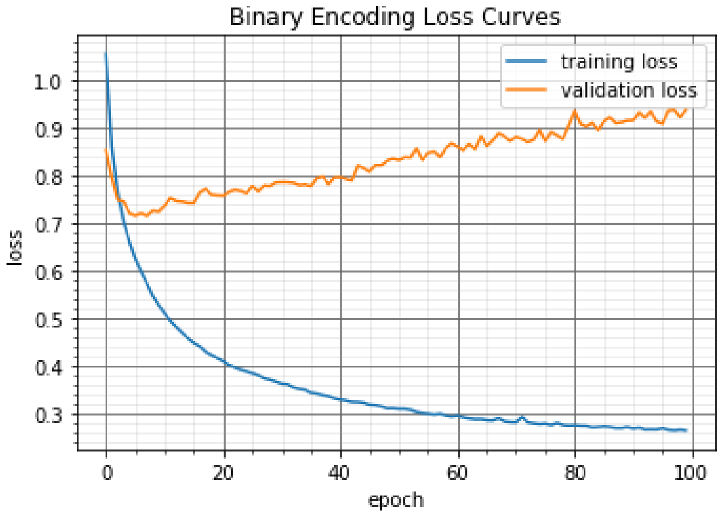


4. Methodology

4.3. Binary encoder



	Train loss	Val loss	Test loss
10	0.5168	0.7240	0.7429
100	0.2635	0.9362	1.6971



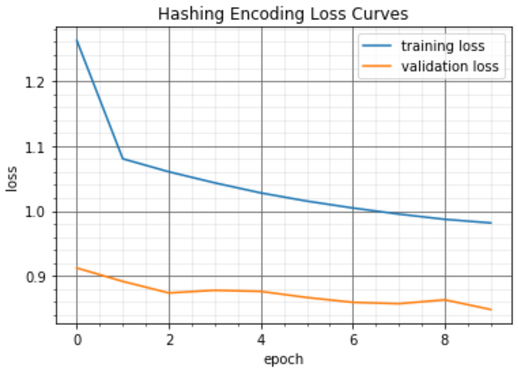
4. Methodology

4.4. Hashing encoder

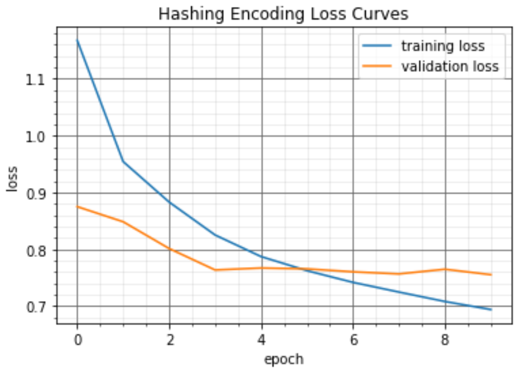
- Suitable for high-cardinality feature vectors
- Values are hashed, converted to integer and mapped to an index in a vector by modulus-dividing by the vector size hyperparameter
- The hashing functions is also a hyperparameter
- Tests showed worse performance after standardization

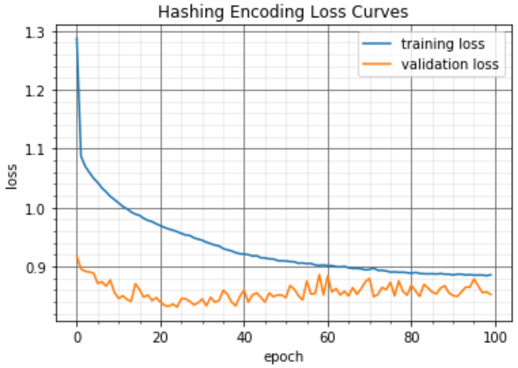
4. Methodology

4.4. Hashing encoder

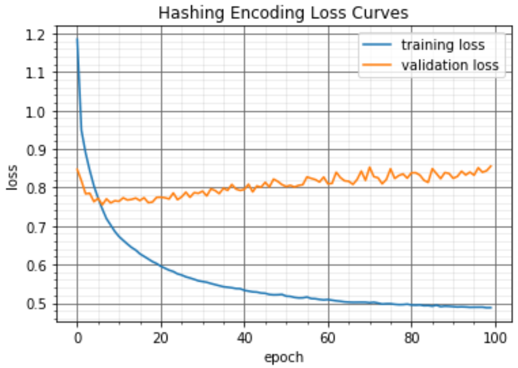


	V = 15		
	Train loss	Val loss	Test loss
10	0.9817	0.8485	0.7306
100	0.8860	0.8525	0.7331





	V = 50		
	Train loss	Val loss	Test loss
10	0.6940	0.7553	0.7360
100	0.4882	0.8556	0.8551



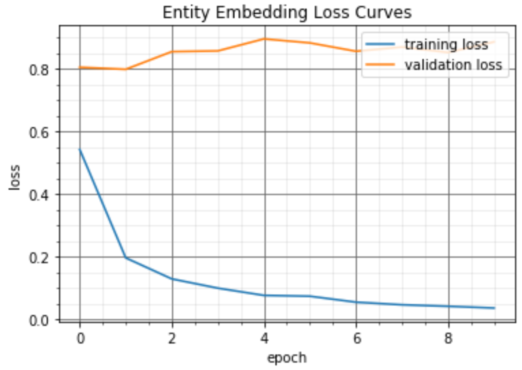
4. Methodology

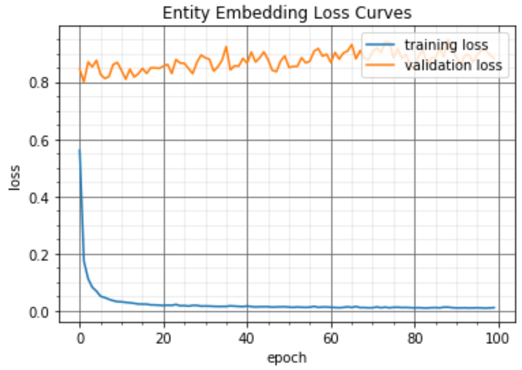
4.5. Entity embeddings

- The most complex approach in terms of network topology
- For each feature vector there are 3 layers
  - Input
  - Embedding
  - Reshape
- Hyperparameter: embedding size

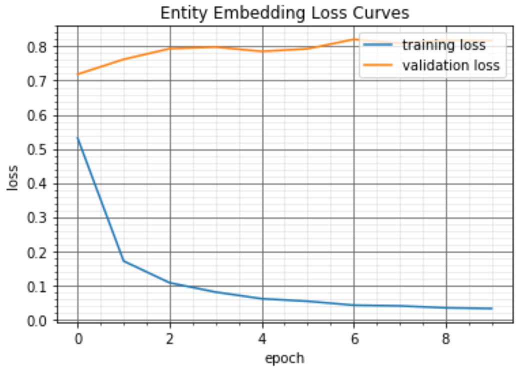
4. Methodology

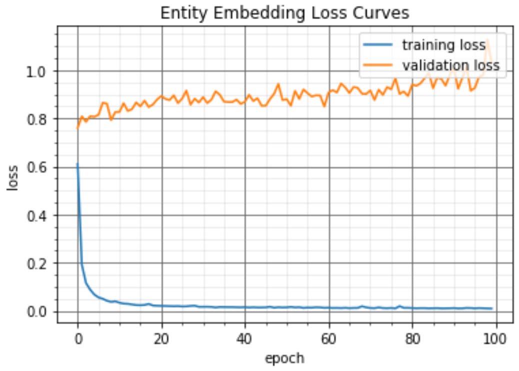
4.5. Entity embeddings





EE = 50			
	Train loss	Val loss	Test loss
10	0.0356	0.8856	1.1423
100	0.0103	0.9805	1.2280

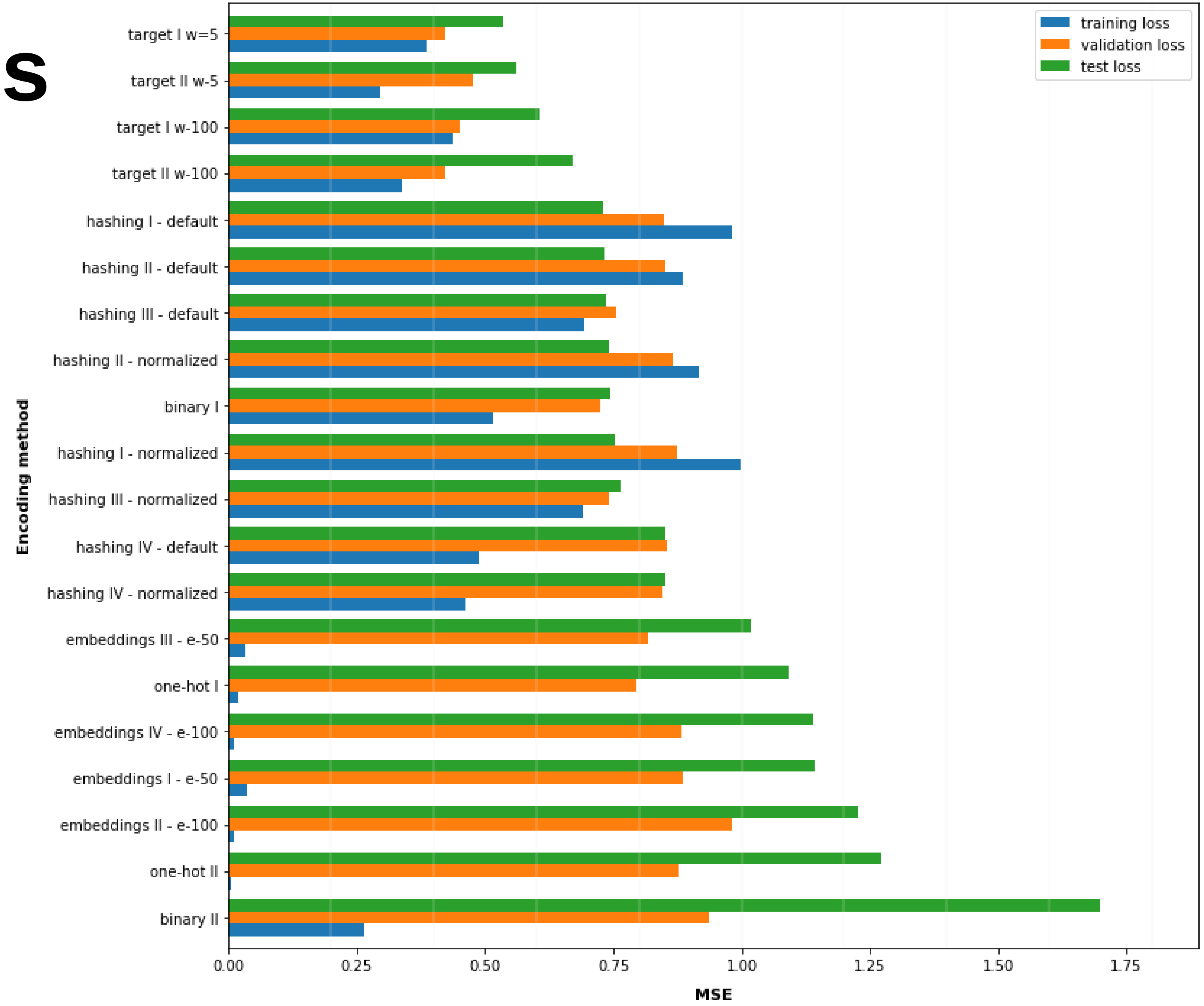




EE = 100			
	Train loss	Val loss	Test loss
10	0.0334	0.8171	1.0170
100	0.0108	0.8828	1.1383

5. Summary of results

Comparative table



6. Conclusion

Key findings

- Averaging over groups of unique feature values performs best
- Increasing dimensionality as a factor of cardinality compromises the model
- No approach is data-agnostic

THANK YOU