Experiment

Creator **Olusanmi Hundogan** Created **Feb 29, 2024, 19:18** Last updated **Apr 24, 2024, 13:28**

Files

- analyze_feature_scaling.py
- experiment_feature_scaling_validation.py

Motivation

The features and target of the model often follow a highly skewed distribution which makes traditional feature scaling techniques impractical. This experiment investigates other techniques and their effect on the final outcome on the model performance. For this purpose we compared a number of confiugrations to select the best feature and target scaling method for the model.

The configurations where:

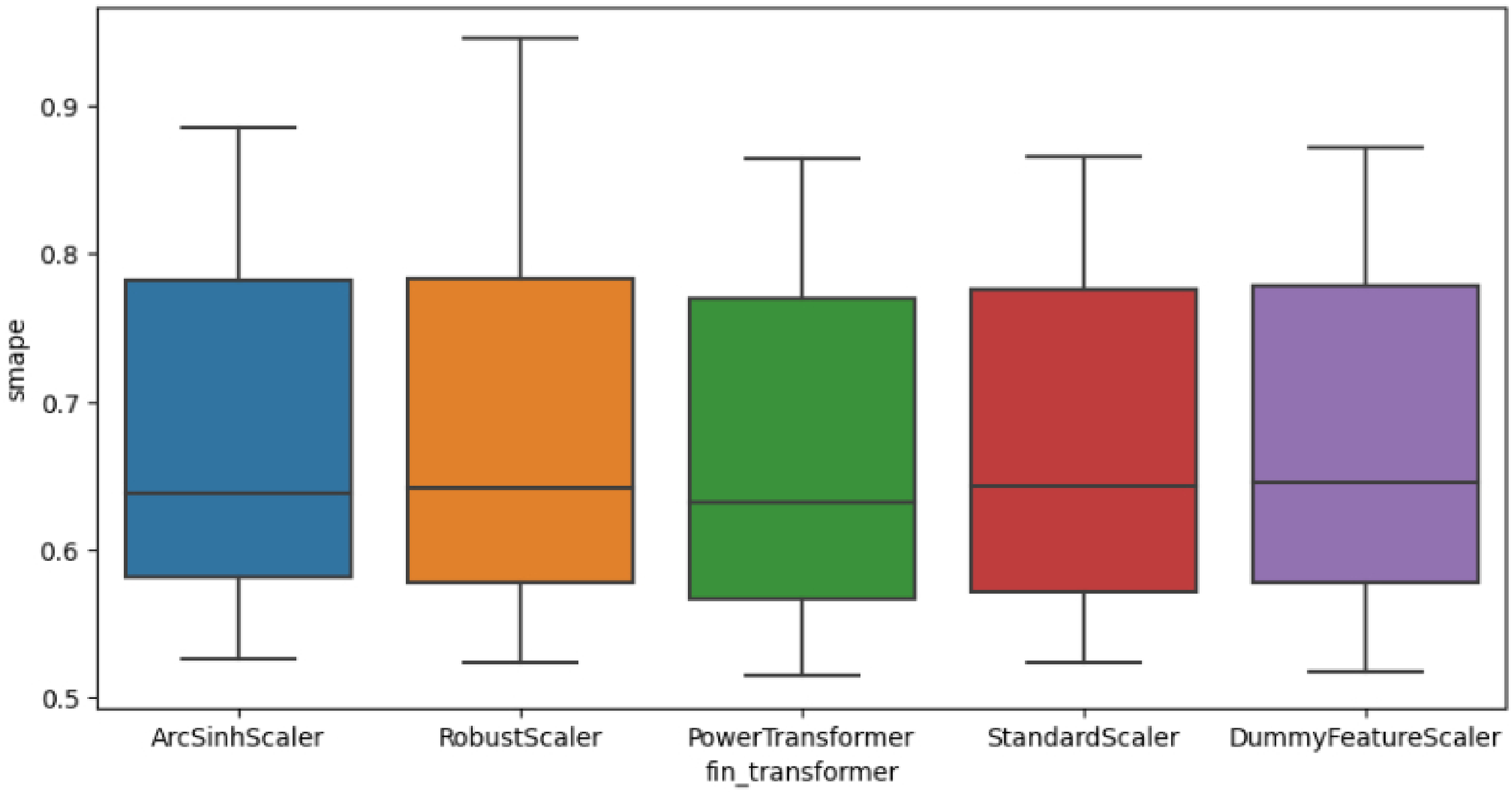
- Feature Scalers
  - ArcSinH
  - RobustScaler
  - PowerTransformer
  - StandardScaler
  - Dummy
- Target Scalers
  - ArcSinH
  - Log
  - Dummy

Design

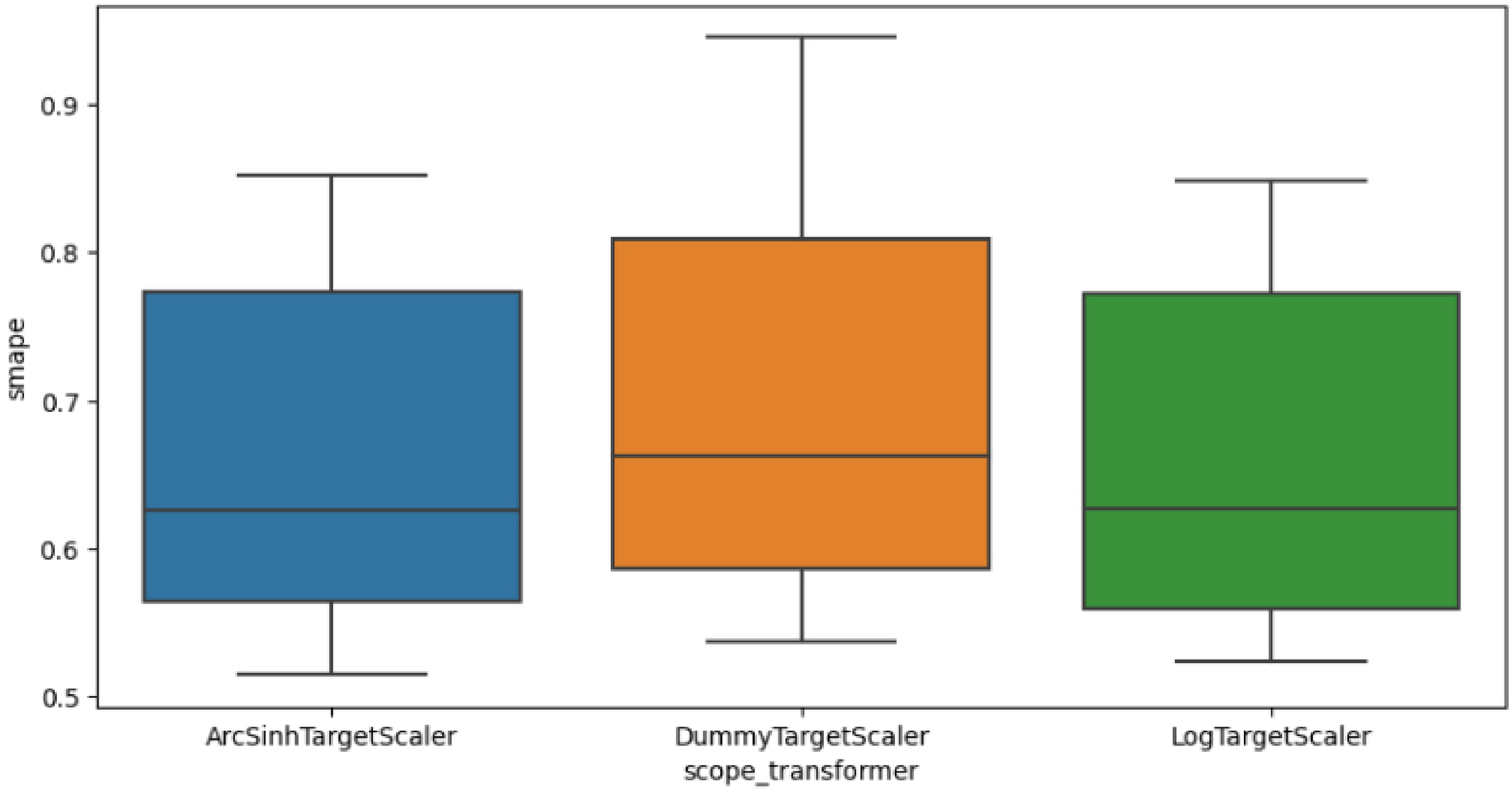
For this experiment, we train an MMA model for each configuration and evaluate the sMAPEvalue all three scopes. We ran the experiment for 10 repetitions each. We ran the experiment with a PCAof 40 components and one without dimensionality reduction. However, for the analysis, we will only on the data generated without dimensionality reduction and average over the scopes.

Results and Insight

In terms of feature scalers, the results show a slight advatage of using the PowerTransformerfollowed by the ArcSinHScaler. Both methods are meant to handle highly skewed features. However, the margins are small.



For the scope scaling techniques, only the Dummy method performs significantly worse. The other methods are on average similar in performance. However, the ArcSinH method has slightly better results on the 1 quartile and slightly worse on the 4 quartile. (Lower sMAPEis better. Hence a lower minimum is better as well.)



The following table shows the sMAPEvalue summaries. The highlighted values denote the highest (red) and lowest (blue) value in the column. The interaction of both feature and target scaling reveals, that the combination of PowerTransformerand ArcSinH yields the best results in terms of the median sMAPE. This combination also outperforms the other combinations on average and the minimum value.

fin_transformer	scope_transformer	mean	std	min	25%	50%	75%	max
ArcSinhScaler	ArcSinhTargetScaler	0.658049	0.106133	0.526036	0.582416	0.626903	0.774611	0.851590
	DummyTargetScaler	0.694521	0.115008	0.543861	0.589285	0.666344	0.816506	0.885468
	LogTargetScaler	0.657118	0.105787	0.529306	0.561052	0.621835	0.770434	0.848125
DummyFeatureScaler	ArcSinhTargetScaler	0.656779	0.108945	0.517469	0.547387	0.635069	0.782452	0.832411
	DummyTargetScaler	0.688724	0.104790	0.558237	0.597833	0.661963	0.787132	0.872212
	LogTargetScaler	0.656630	0.101887	0.525986	0.573447	0.623095	0.765341	0.833314
PowerTransformer	ArcSinhTargetScaler	0.650699	0.102703	0.515080	0.564542	0.617879	0.764572	0.829382
	DummyTargetScaler	0.683555	0.112943	0.539602	0.575953	0.653786	0.820444	0.864499
	LogTargetScaler	0.654637	0.106691	0.529033	0.559679	0.621082	0.767837	0.821005
RobustScaler	ArcSinhTargetScaler	0.659451	0.096062	0.528938	0.573205	0.637434	0.762223	0.817834
	DummyTargetScaler	0.698723	0.113391	0.564507	0.594890	0.665680	0.809193	0.945304
	LogTargetScaler	0.659915	0.105931	0.523437	0.556459	0.633432	0.782277	0.829364
StandardScaler	ArcSinhTargetScaler	0.655342	0.099066	0.531864	0.563580	0.623301	0.754848	0.822109
	DummyTargetScaler	0.687376	0.113319	0.537170	0.582612	0.656115	0.807426	0.865761
	LogTargetScaler	0.652667	0.098960	0.523640	0.561298	0.623794	0.762674	0.836148

Decisions

Update 29.01.24 :

The results show a clear favorite for the PowerTransformation, ArcSinH scaling method. However, these results are subject to change as we will soon have more feature data. In the next experiment, we will likely skip computing the results for the PCAcase.

Update 25.03.24 :

Resultson the newer data suggest, that ArcSinH-ArcSinH also performs well. The differences are miniscule. However, it appears that ArcSinhScalerperforms better than the PowerTransformerin general.

fin_transformer	scope_transformer	mean	std	min	25%	50%	75%	max
ArcSinhScaler	ArcSinhTargetScaler	0.347400	0.013292	0.317119	0.342650	0.347473	0.354338	0.366027
	DummyTargetScaler	0.384481	0.016647	0.353955	0.375076	0.386971	0.394511	0.408405
	LogTargetScaler	0.346545	0.011787	0.317603	0.345917	0.348163	0.353749	0.359648
DummyFeatureScaler	ArcSinhTargetScaler	0.384007	0.013375	0.357620	0.382043	0.384849	0.393116	0.402323
	DummyTargetScaler	0.432432	0.018166	0.396561	0.424822	0.433566	0.440025	0.459341
	LogTargetScaler	0.386554	0.015505	0.356905	0.379309	0.387122	0.395379	0.413345
PowerTransformer	ArcSinhTargetScaler	0.350744	0.013716	0.323491	0.345042	0.353977	0.360384	0.366615
	DummyTargetScaler	0.390487	0.016643	0.357005	0.381800	0.395660	0.401980	0.411003
	LogTargetScaler	0.350335	0.007975	0.332295	0.350678	0.352297	0.355279	0.358300
RobustScaler	ArcSinhTargetScaler	0.384204	0.016995	0.347612	0.377083	0.387152	0.395672	0.408067
	DummyTargetScaler	0.425779	0.011592	0.398984	0.420814	0.429741	0.432537	0.439162
	LogTargetScaler	0.385545	0.012884	0.363174	0.375213	0.388600	0.392945	0.402629
StandardScaler	ArcSinhTargetScaler	0.387251	0.013440	0.362955	0.379871	0.388869	0.396656	0.407037
	DummyTargetScaler	0.424823	0.015762	0.396884	0.412865	0.426208	0.438479	0.442549
	LogTargetScaler	0.385395	0.013561	0.361182	0.382789	0.386485	0.394716	0.403402

Update 24.05.24 :

In the end the differences appear to be miniscule as long as we use either PowerTransformeror ArcSinhScalerinstead of RobustScalerfor the preprocessing.