Estimating Solvency Ratios Using Machine Learning

CAS Innovation Project

Author: Oliver Stoll
Supported by: Santiago Brunner and Mahbod Tajdini

1.Summary

We have tested an idea about forecasting Solvency ratios using machine learning models. We analysed synthetic data sets which comprised economic input data and economic capital components as output. We applied machine learning models to learn the relationship between those inputs and outputs, compared different methodologies, tuned models and looked at explainability. Both XGBoost and neural networks showed strong predictive power. However, when doing a back-test with live data, XGBoost performed much better than a neural network, although on test data their scores were very similar. We also saw that only a few input variables determine the outcome of the economic capital model.

2.Introduction

Insurance companies need to manage their solvency in terms of maintaining sufficient capital to be able to serve their policyholders. This is important in particular in life insurance which often has an investment component and where contracts could be in force for 30 years and more. This requires a long-term view on insurance companies' risk profile. Traditionally, a very simplified factor model was in place to calculate a solvency requirement (Solvency I [1]). With the advent of more powerful computing resources, a new risk capital-based regime was developed. In the European Economic Area (EEA) this is called Solvency II; in Switzerland an equivalent methodology was developed as Swiss Solvency Test (SST). Both methodologies require identification and evaluation of risks, leading to a risk capital requirement which is confronted with available capital sources. The main KPI is the solvency ratio which is the relationship between available capital to required capital. This ratio should always be above 100%, usually with a comfortable margin on top. The objective of this project is to test whether solvency ratios can be reliably estimated using machine learning models trained on synthetic data generated from an actuarial projection system.

[1] Bundesanstalt für Finanzdienstleistungsaufsicht (BaFin), "Solvency I," accessed November 2, 2025, https://www.bafin.de/DE/Aufsicht/VersichererPensionsfonds/Allgemeines/SolvencyI/solvency_I_artikel.html.

2.1. Problem Statement

Insurance companies need to ensure that they are always able to compensate their insurance clients. For this reason, supervisory authorities are concerned with the solvency of their regulated insurance companies. Since the 2000s, in several markets risk-based concepts have been developed to measure that solvency through a solvency ratio ("SR")[2], e. g. Solvency II in the European Economic Area and the Swiss Solvency Test. Roughly speaking, insurance companies need to calculate a required risk capital that measures the risk of their business, and they have to compare that with sources of available capital. The ratio of available capital to required capital will then give the SR. If the SR is below 100%, the company is technically insolvent, and the regulator will impose measures such as capital increases or (as an exception) allow the use of bonus funds to recapitalize the insurer (e. g. German VAG § 140 (1) 1.). A SR above 100% is good, but anything near 100% will already trigger an increased supervision, so insurers tend to manage a SR somewhere between 150% and 300% (an even higher SR will mean that there is too much capital that could be better allocated elsewhere).

This SR has to be calculated on a quarterly basis for Solvency II. In particular for the life business, the process is quite time consuming. Usually, an actuarial projection system is used (such as commercial tools like Prophet or Risk Agility FM) which models all insurance tariffs plus the insurance policies in force and uses economic scenarios to project cashflows for a rather long timespan (e. g. 50-80 years).

The above-mentioned projection model is fed with stochastic scenarios, produced by an Economic Scenario Generator ("ESG"). The ESG is calibrated with prevailing market conditions as of t=0 and uses stochastic differential equations to produce a large set (usually between 1,000 and 5,000) of economic scenarios. These scenarios are generated so that they follow certain side conditions, e. g. to be market consistent and arbitrage free. The scenarios then generate different outcomes of the projection model, and basic statistics is performed to calculate quantiles needed for risk management. Options on the liability side can also be assessed e. g. by looking at the difference between deterministic runs vs. the median of the stochastic outcomes. Additional analyses could for instance look at the effectiveness of so-called management rules, where certain actions are implemented in the model that are triggered in distressed situations. When looking at the difference of runs with and without those management rules, one could assess the effectiveness of those built-in rules.

[2] Throughout this document, you will also read a German name of SR: "quote". Similarly, for own funds you will see "EM" ("Eigenmittel") and for required capital, "SCR" ("Solvency Capital Requirement").

These projections are run multiple times under different scenarios (e. g. an interest rate down shock) and differences in the weighted outcome are then considered as an associated risk capital[3]. Such individual risk capital components are then aggregated (assuming a certain level of diversification), leading to a final required capital and the calculation of the SR[4].
In addition to that quarterly reporting cycle, insurance companies must continuously monitor their solvency position. For this reason, management such as the CRO needs to be able to estimate the solvency situation on an ongoing basis. This continuous monitoring capability also supports the “Use Test” requirement, demonstrating that solvency models are effectively integrated into day-to-day risk management and decision-making.
Unfortunately, the above-described production process is time and labour intensive, which makes for instance daily (exact) recalculations nearly impossible.

2.2. Idea

This is why we explored the following idea:
Let’s create stochastic scenario sets for different initial market conditions. These initial market conditions are synthetic and cover a wide range of possible market parameters. Calibrate the ESG for each of these initial market conditions and generate a stochastic scenario set (e. g. 1,000 scenario sets for each initial market condition). Run the projection tool and calculate risk capital on its outcomes, for each synthetic scenario. If you repeat this for each set of market conditions, one gets a dataset that includes those market conditions as inputs, and resulting risk capital as output.
If we now have a way to learn how market conditions will influence risk capital (and solvency ratios), one could use that learned dependency to calculate updated solvency ratios by feeding our model with current market conditions. Such a calculation (inference) can be run with minimal effort and would allow a monitoring of the solvency situation e. g. on a daily basis through a risk management dashboard.
Based on this idea, a data set of 10,230 synthetic market conditions plus available capital, risk capital and a resulting solvency ratio was generated and used as a basis for our analyses.[5]

[3] Note this is a simplification of the process and just for illustrating the idea behind it.
[4] Generally speaking, determining available capital is rather straight-forward, although there are also rules for admissible capital types (called capital tiers in the SII context).
[5] This means there are 10,230 synthetic initial conditions of the ESG which produced 1,000 scenarios for each of those starting points. The projection tool was then run for each of these scenarios sets, and the results were evaluated of each scenario set, leading to risk capital requirements and a SR.

2.3. Assumptions and Constraints

This approach implicitly assumes that no major structural changes have taken place:
On the asset side, a different asset allocation would result in a change in the risk situation (both on the available and required capital side). This will have to trigger a recalculation of the initial setup.
On the liability side, a different business mix (different tariffs, new business, lapses etc.) would also require a recalculation. In practice, the business mix of a life insurer does change very slowly, so this would usually not pose a problem.

3.Data Description

The client produced a set of 10,230 scenarios sets (initial conditions and resulting capital). Input fields were:

- 8 fields storing asset side related data (3 for interest rate, 3 for volatility, 2 for default risk)
- 12 fields storing flags that govern management rules of the projection tool. These are rules triggering sales and purchases of assets etc. and govern actions in distressed situations.

Parameter	Description
ZSK1 **ZSK2** **ZSK3**	ZSK1 to ZSK3: "Zinsstrukturkurve" -- decomposition of the EUR 1-20 interest rate curve into three major components (principal components analysis).
Vola4	Volatility of interest rate market
Vola5	Volatility of stock market
Vola6	Volatility of real estate market
Verlust7	Market value loss stocks at t=0
Verlust8	Market value loss real estate at t=0

Management rules	
	1. Management rules regarding dynamic asset allocation:
MR9	Target fixed income asset allocation
MR10	Minimum fixed income asset allocation
MR11	Target and minimum fixed income asset allocation in "economic distressed situations"
MR12	Target real estate asset allocation
MR13	Target real estate asset allocation in "economically distressed situations"
MR14	Max. step size for annual adjustment of target and minimum asset allocation of fixed income and real estate
MR15	2. Target duration of fixed income reinvestment
MR16	3. Return requirement of statutory equity
MR17	4. Target percentage of equity compared to technical provisions
MR18	5. Target of profit sharing component for other sources ("übriges Ergebnis") for new business
MR19	6. Flag for increase in interest rate bonus
MR20	7. Target period for management of hidden reserves of fixed income

Table 1: Input data.

Above inputs entered the data generation process in different areas:

- Input for economic scenario generator: new stochastic scenario sets are generated based on those inputs: Parameters 1-6.
- Input for market value adjustment of assets as at t=0: Parameters 1-3 and 5-8.
- Input parameters for stochastic projection model: Parameters 9-20.

Output: Three interdependent output columns (third column is calculated)	
EM	"Eigenmittel": own funds
SCR	Solvency Capital Requirement: economic risk capital
Quote	Solvency ratio = EM /SCR

Table 2: Output data.

Available capital, ‘EM’, is mainly determined by the market value of the asset portfolio as at t=0. The SCR is calculated with the use of stochastic projection systems. First individual SCR components, like SCR for interest rate risk (“SCR_IR”), are calculated. In the case of SCR_IR, this is the outcome of the difference of the median of the stochastic base run and the worst result from both an interest rate up and interest rate down run. Those capital components are then aggregated using a prescribed diversification method to yield a total SCR.

We use Seaborn visualizations to examine the distribution of input parameters and their dependencies:

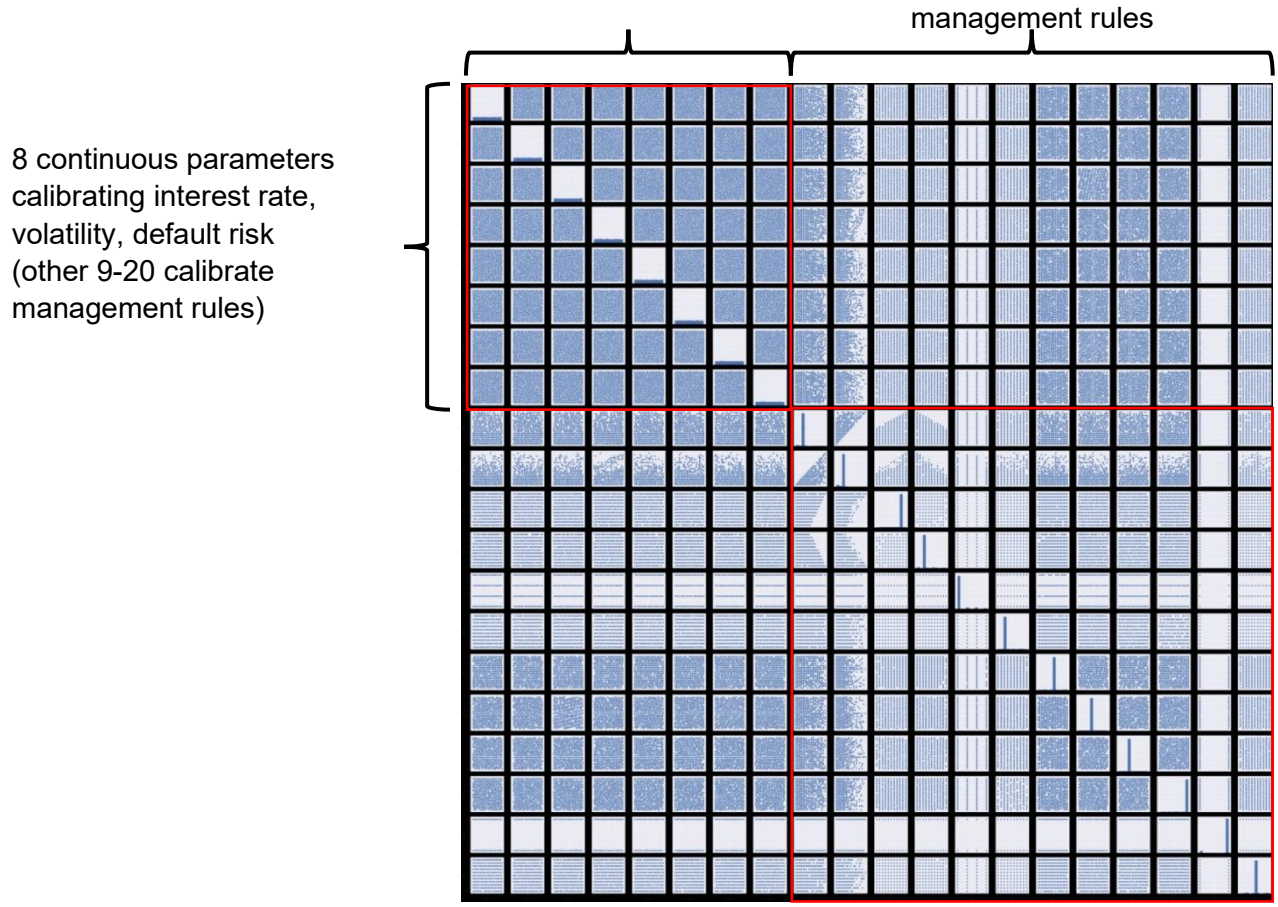

Figure 1: Distribution of input parameters (seaborn.pairplot).

In figure 1, you see pairplots of the input parameters, each blue dot depicts one actual combination in the data set. In the top left part, you can see that the 1x1 boxes, showing combinations of market parameters, uniformly fill those boxes. So there are no clusters and the full potential interval of parameterizations is used.
In the lower right part, some boxes contain discrete values or are not filled evenly. This is due to
a) some management rules act like flags and use discrete parameters;
b) some pairs of management rules are not independent and thus create a visible pattern in the 1x1 box.

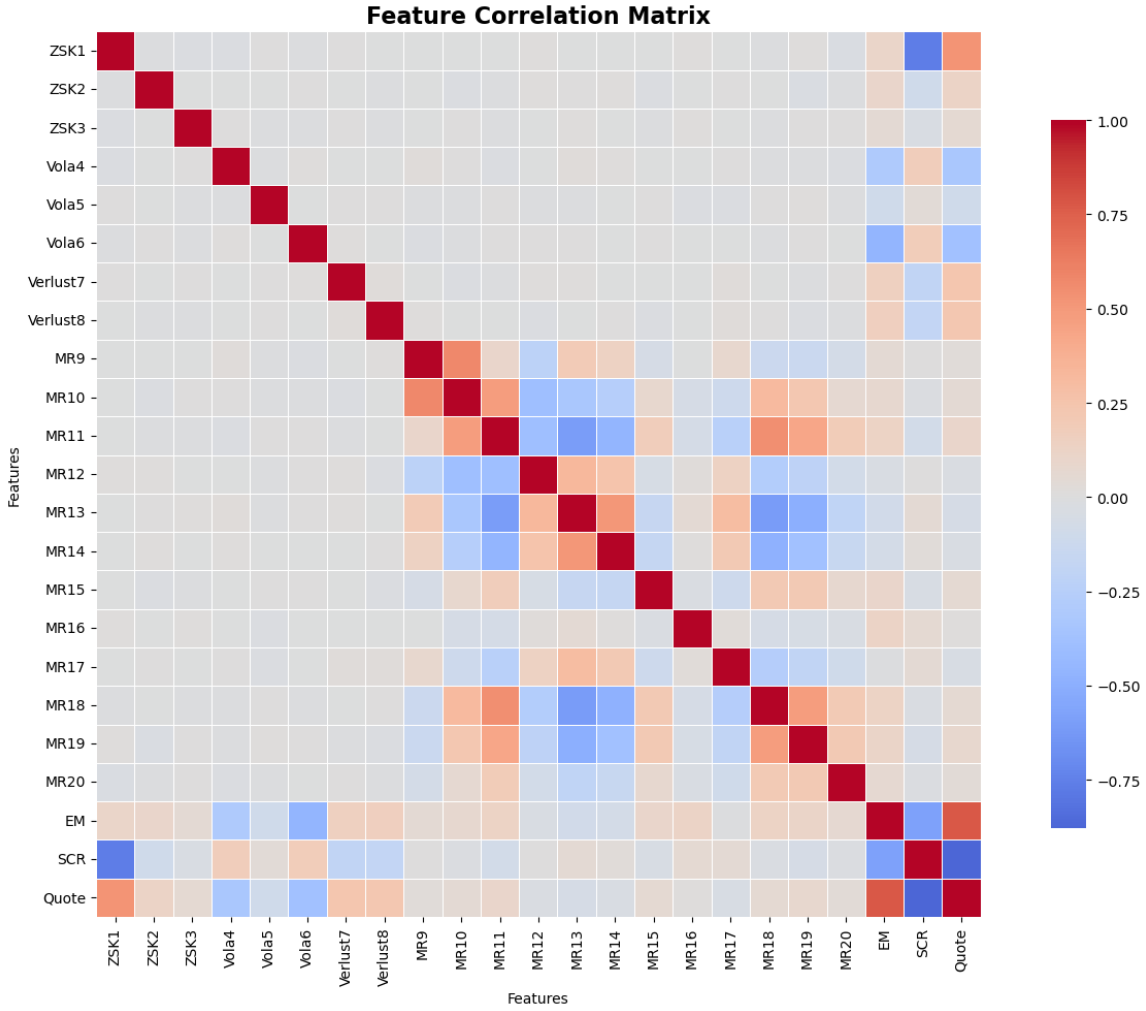

Figure 2: Correlation between input parameters.

Some highlights:

- Economic input parameters filled their input space quite evenly (e. g. if it is the interval between 0 and 1, the interval was covered quite comprehensively).
- Some management rules are not continuous but are rather flag like.
- Most management rules cannot be set independently.
- We see no correlation between the economic inputs.
- There are correlations between economic inputs and outputs as for instance:
    - Increasing ZSK will reduce SCR and increase solvency ratio;
    - Vola4 has the opposite effect.
- There are correlations between outputs as such:
    - SCR and EM are anti-correlated: in good scenarios, SCR decreases and EM increases, in distressed scenarios this is opposite.
    - The SR (“quote”), as a quotient between EM and SCR, increases if EM increases and decreases as SCR increases.

EM:	
	Mean: EUR 1,114M, Std: EUR 433M
	Min: EUR -4,740M, Max: EUR 2,109M
	Median: EUR 1,180M
SCR:	
	Mean: EUR 633M, Std: EUR 293M
	Min: EUR 224M, Max: EUR 2,149M
	Median: EUR 505M
SR:	
	Mean: 217%, Std: 101%
	Min: -247%, Max: 465%
	Median: 243%

Table 3: Statistics related to target variables.

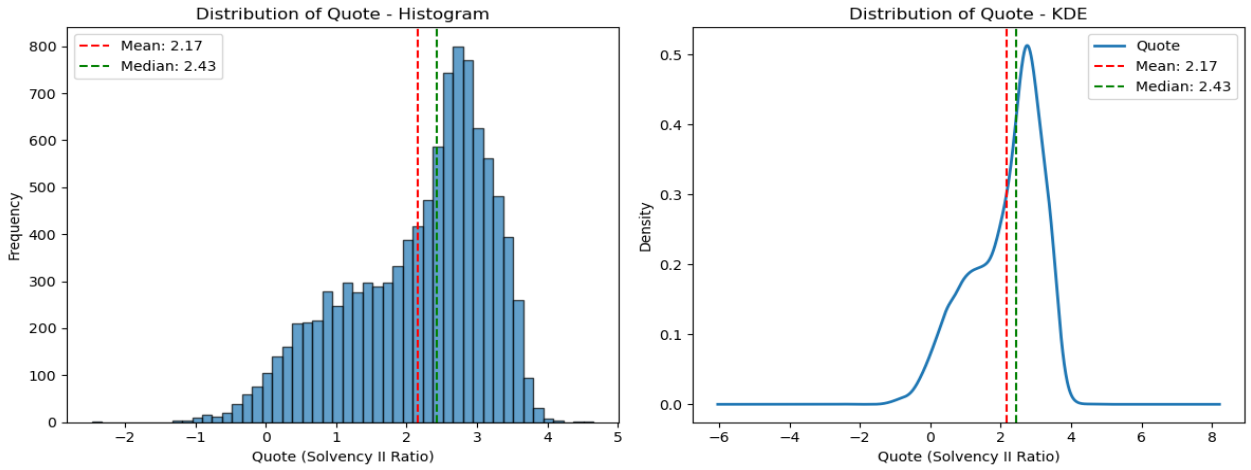

Figure 3: Histogram of the SR.

The histogram and its smoothed version (kernel density estimate) show how the SR is distributed in the data set. A big hump can be seen between roughly 150% and 300%, and a “shoulder” at around 100%.

We identified one outlier that appeared implausible and is likely due to a failure of the actuarial projection model, as it led to very extreme EM and SCR values (EM around -4.7b (nearly double than the second lowest EM) and SCR 1.9b). We excluded that point from our analyses as such a value would not make sense economically. This point can be clearly seen in figure 4, which shows the distribution of EM and SCR in a scatter plot.

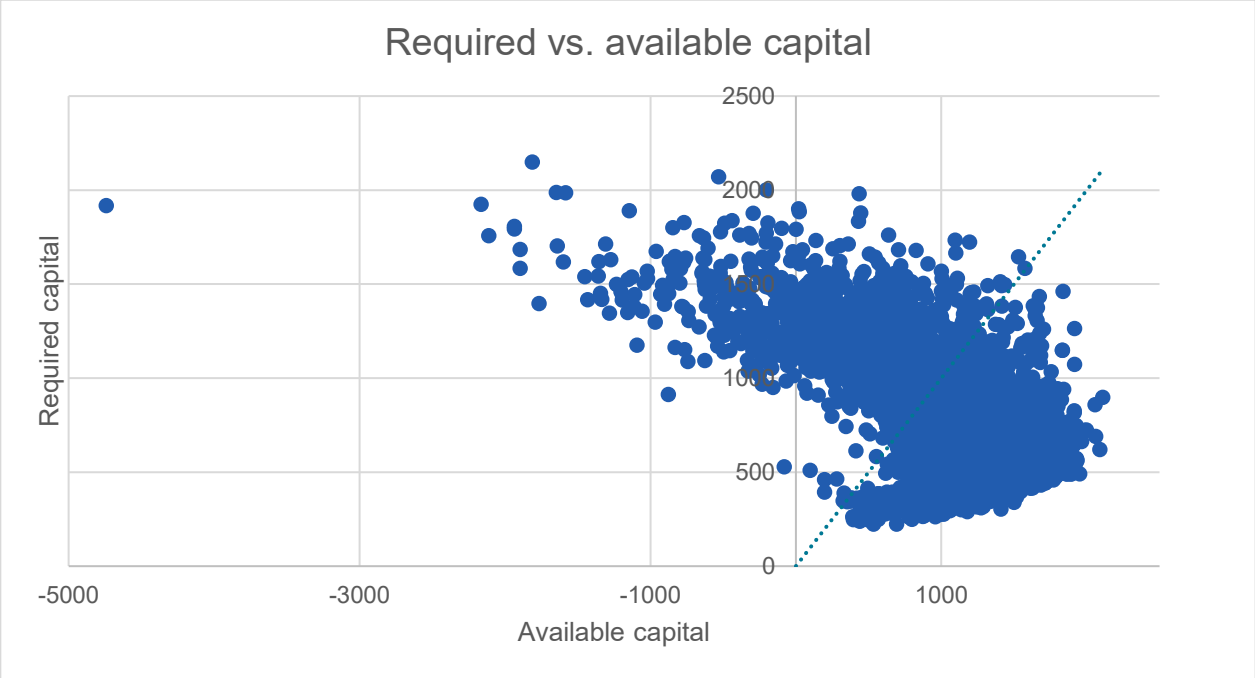

Figure 4: Required vs. available capital scatter plot (outlier near the top left corner).

4.Methodology and Implementation

This section describes the machine learning workflow.

4.1. Preprocessing

Except for the above-mentioned removal of one data point, we used the data set as is. As a first step, we divided the set into a training, validation and test set, at a 60% - 20 % - 20% split.
We then normalized the data based on the training set, and applied the same transformation to the validation and test set in order to avoid data leakage. The normalization was mainly done as the outputs come in different orders in magnitude (SR vs. capital figures).
As the outputs EM and SCR are (negatively) correlated[6], we tested a two-dimensional output approach, i. e. our models predicted EM and SCR at the same time (ratio was then calculated) In particular for the regression models, we also directly estimated the SR, which proved to outperform the indirect approach: the cases of high SCR, low EM are difficult to predict, and the error was compounded when deriving the SR as a quotient of EM and SCR instead of its direct calculation.

[6] Typically in distressed scenarios, own funds decrease while capital requirements increase.

4.2. Model Selection

We tested models of the following model families:

- Regression
- Tree based models: Random Forest and XGBoost
- Neural networks: MLPRegressor and PyTorch

Within regression, we looked at seven model configurations:

- Dummy Regressor
- Linear Regression
- Ridge with cross validation grid search (CV)
- Lasso with CV
- Elastic Net
- Quadratic
- Cubic

With PyTorch, we did a CV where we tested the width and depth of the net, dropout, batch norms, learning rates, weight decays. Similarly, with MLPRegressor, we tested the width and depth of the net, activation functions, values for alpha (L^2penalty) and initial learning rates.

4.3. Model Comparison and Quantitative Performance

Baseline linear and regularized regression models (Linear, Ridge, Lasso) were insufficient for capturing the complex, non-linear dynamics of the solvency calculations. They served as a poor fit, achieving an R^2 of only 0.69 and a test RMSE of around 0.16-0.20. A quadratic polynomial model offered a significant improvement, reducing the RMSE by c. 40% to 0.11. This confirms the presence of strong non-linear relationships. However, a cubic model began to overfit, with its test performance deteriorating. Advanced, non-linear models (specifically Gradient Boosting and neural networks) demonstrated markedly superior performance, confirming their ability to effectively emulate the non-linear dynamics. The performance of the best models on the test set is summarized below[7].

[7] The full comparison can be found in the appendix.

Target	Best Model	Val RMSE	Test RMSE	R^2 (test)	Approach
SR	**XGBoost Tuned (2 + 1-dim)**	**0.0807**	**0.0802**	**0.9453**	Multi-output
	PyTorch NN (2 + 1-dim)	0.0896	0.0831	0.9413	Multi-output
	MLP Regressor (1-dim)	0.0955	0.0838	0.9402	Direct
SCR	**MLP Regressor Tuned (2 + 1-dim)**	**0.0733**	**0.0823**	**0.9367**	Multi-output
	XGBoost Tuned (2 + 1-dim)	0.0764	0.0840	0.9341	Multi-output
EM	**MLP Regressor Tuned (2 + 1-dim)**	**0.0543**	**0.0541**	**0.9306**	Multi-output
	XGBoost Tuned (2 + 1-dim)	0.0555	0.0562	0.9251	Multi-output

Table 4: Validation and Test Performance (Normalised RMSE and R^2).[8]

The results show that both XGBoost and tuned neural networks (both sklearn's MLPRegressor and the PyTorch implementation) achieved excellent R^2 values between 0.925 and 0.945 across all targets.

- For the primary target, the solvency ratio (SR), the tuned XGBoost model performed marginally best. Among the top models, you can also find MLP Regressor directly predicting the SR (see below for a brief discussion).

- For the individual components 'EM' (Eigenmittel / own funds) and 'SCR' (Solvency Capital Requirement), the tuned multi-output neural networks held a slight edge.

The performance difference between the top two model families is not statistically significant, implying both are viable candidates for a production environment.
A key methodological finding related to the target variable. As mentioned above, directly predicting the SR ratio proved more robust in the regression setting compared to the indirect approach (i.e., predicting 'EM' and 'SCR' separately and then calculating their ratio). This was because small prediction errors in 'EM' and 'SCR'—especially in distressed scenarios with low 'EM' and high 'SCR'—were compounded when divided, leading to high instability in the final SR prediction. However, from the view of an actuarial expert, we would be more interested in calculating the two components, EM and SCR, within the same model, as this would give more insight into its prediction, in particular how both elements interact.[9]
We would expect small performance improvements if the models were fine-tuned even further. In addition, the model choice should be stable in time, i. e. we should confirm model performance on data sets produced for other periods in time (e. g. one or two years later), which weren't available to us for this project. This could be an area for further investigation.

[8] "1-dim" means a model that only predicts one output variable, also named as "direct".
"2+1-dim" denotes a model that predicts EM and SCR, and calculates SR as a ratio (also shown as "multi-output").
[9] E. g. is a distressed SR due to low EM or high SCR? Is SCR more sensitive than EM to certain inputs?

4.4. Visual Evaluation

Visual diagnostics created in the notebook confirm these quantitative findings.

- **Actual vs. Predicted Plots[10]:** For the top-performing XGBoost and neural network models, these scatterplots show a tight diagonal clustering around the 45° line. This indicates that the models are well-calibrated and make accurate predictions across the entire range of solvency ratios, including the more extreme high- and low-variance regions. In contrast, the plot for the quadratic model showed a distinct curvature, visually confirming its systematic underestimation of extreme values.

Figures 5 to 9 show the predicted vs. actual diagram for the SR.

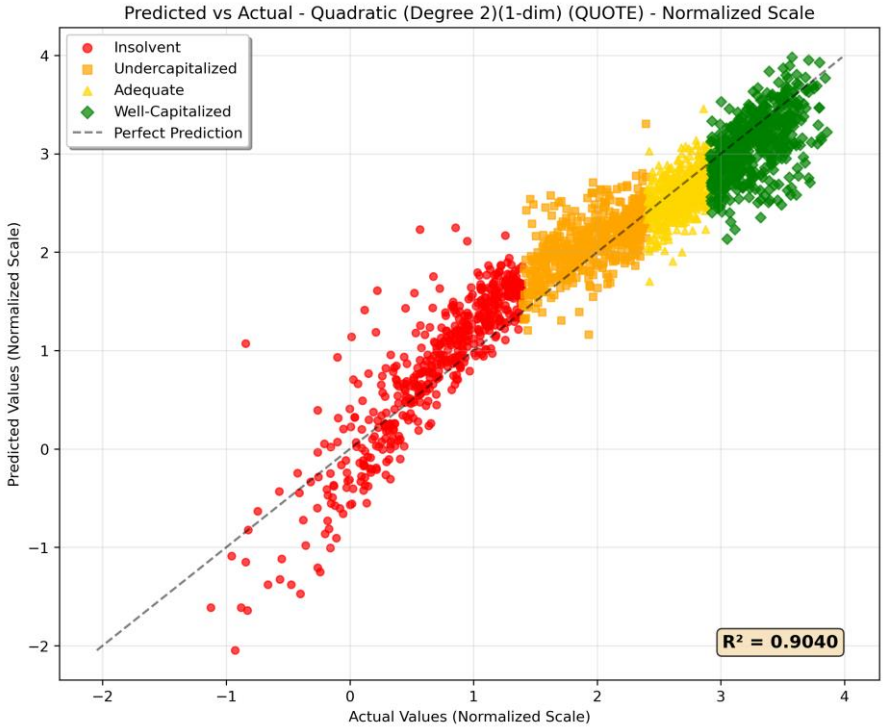

Figure 5: Quadratic regression; has issues in particular with low solvency regimes.

[10] Note that there are two scales, normalized vs. calculated. Calculated means one can directly read the real solvency ratio, while normalized gives a distorted ratio. For the purpose of illustration, both scales provide similar insights. See also the comment in 6.

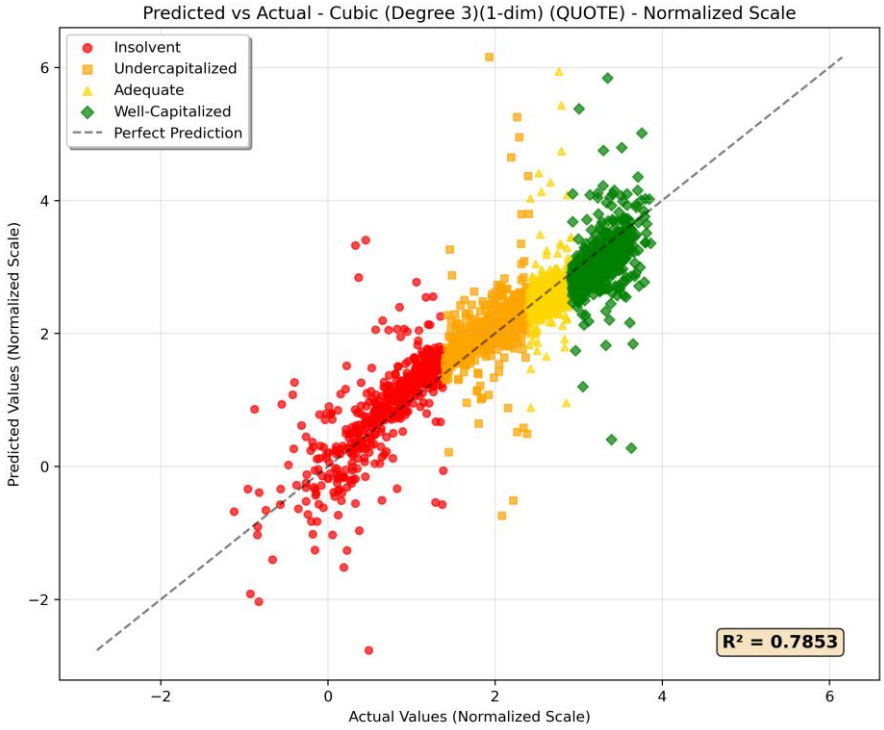

Figure 6: Cubic regression: already in the overfitting regime (compare to quadratic regression).

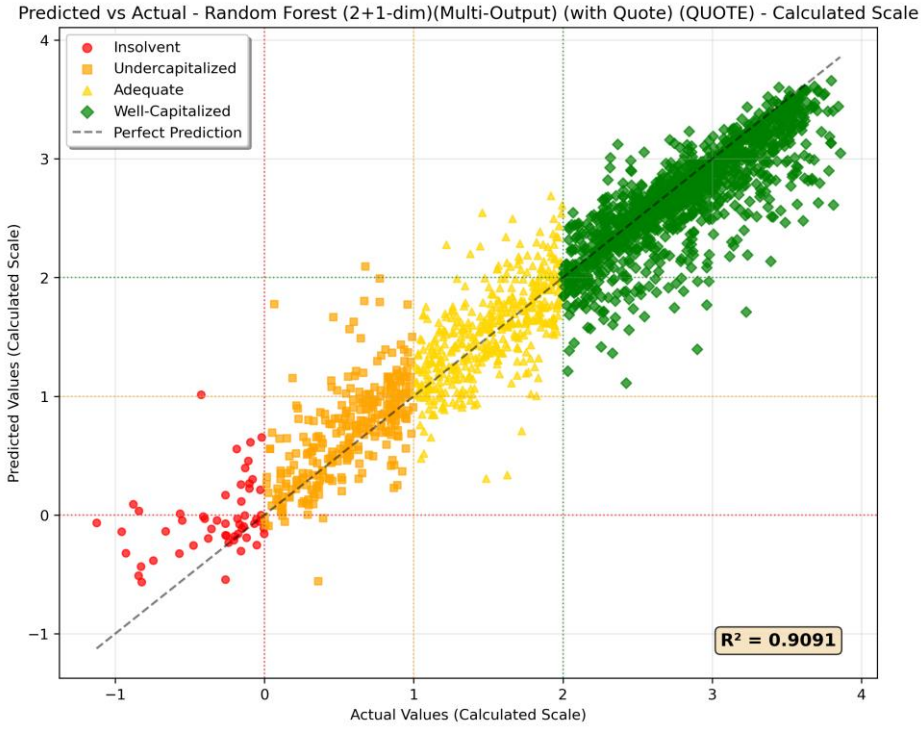

Figure 7: Random forest (SR calculated).

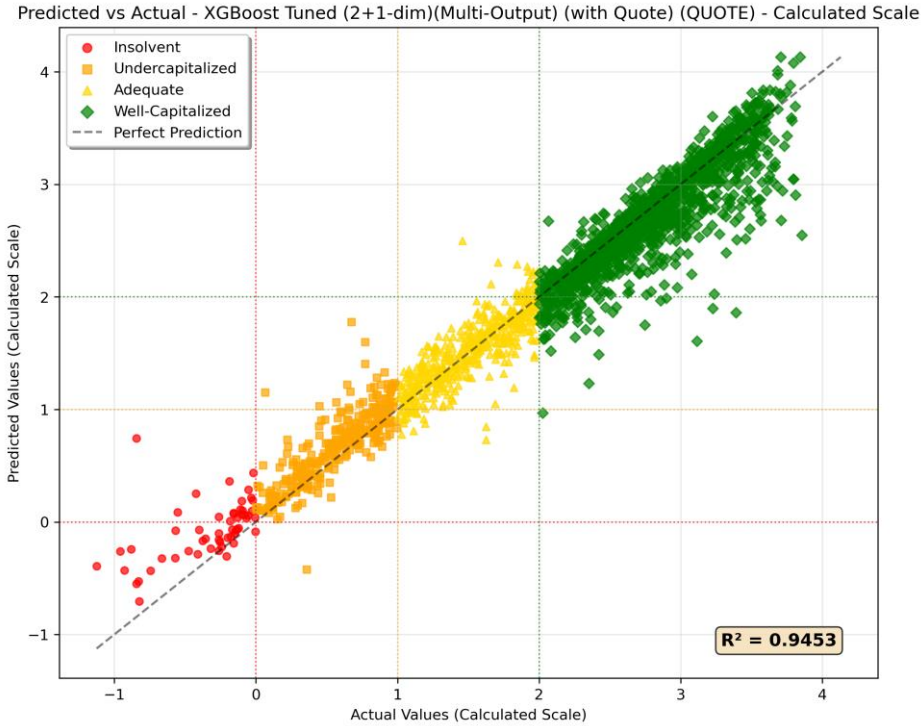

Figure 8: XGBoost works remarkably better than plain random forests.

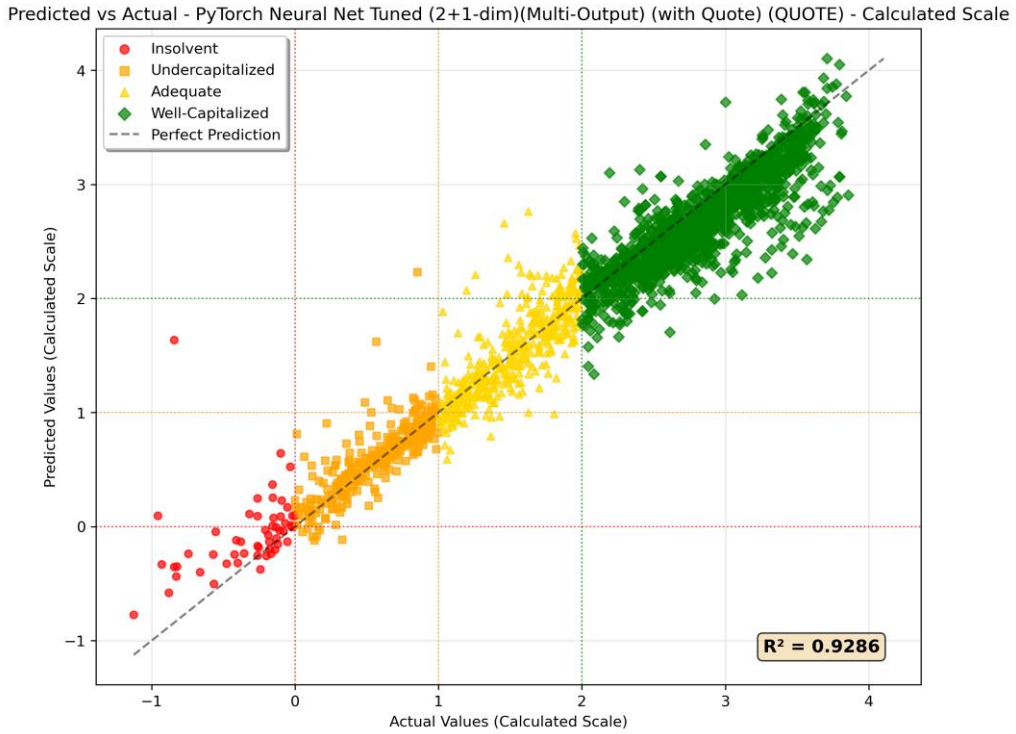

Figure 9: PyTorch: Although R^2 is slightly lower, a visual inspection shows a very similar pattern compared to XGBoost.

- **Residual Plots:** The residual plots for the XGBoost and neural network models demonstrated symmetric, near-zero-mean errors, consistent with unbiased estimates.

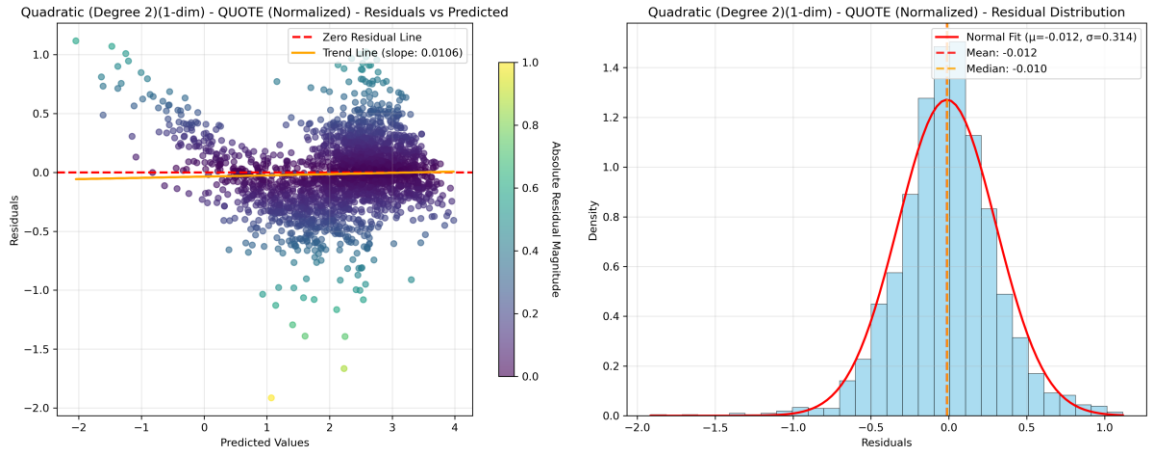

Figure 10: Residuals, quadratic regression.

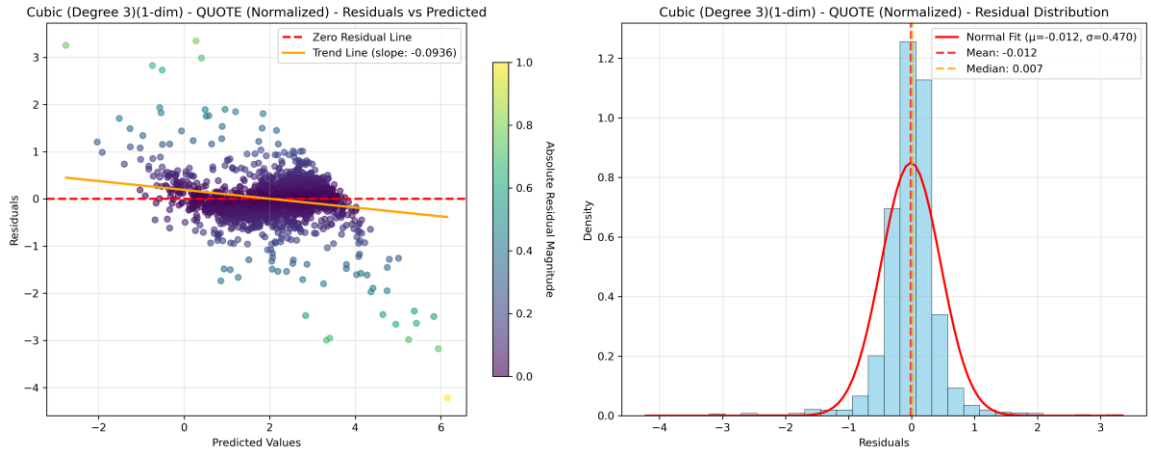

Figure 11: Cubic regression. Here we can see that cubic regression is already worse than quadratic regression. Note the different scales.

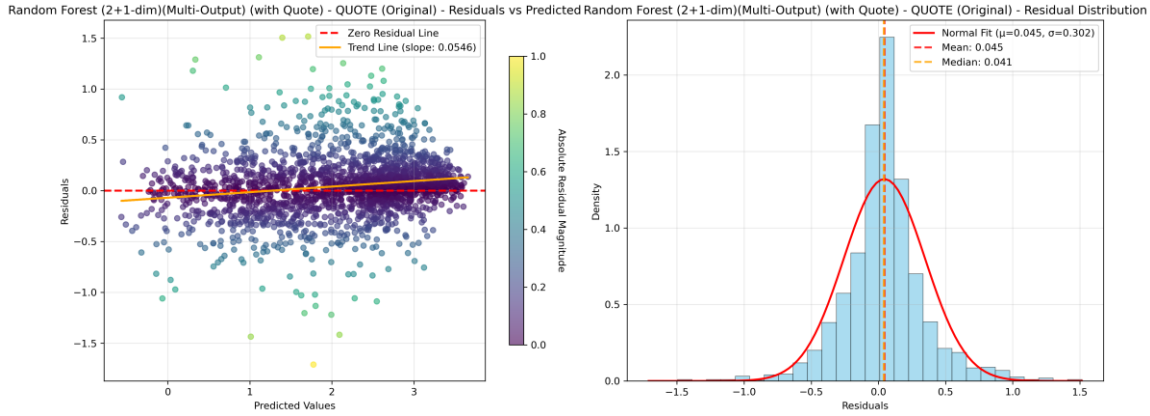

Figure 12: Residuals for random forest.

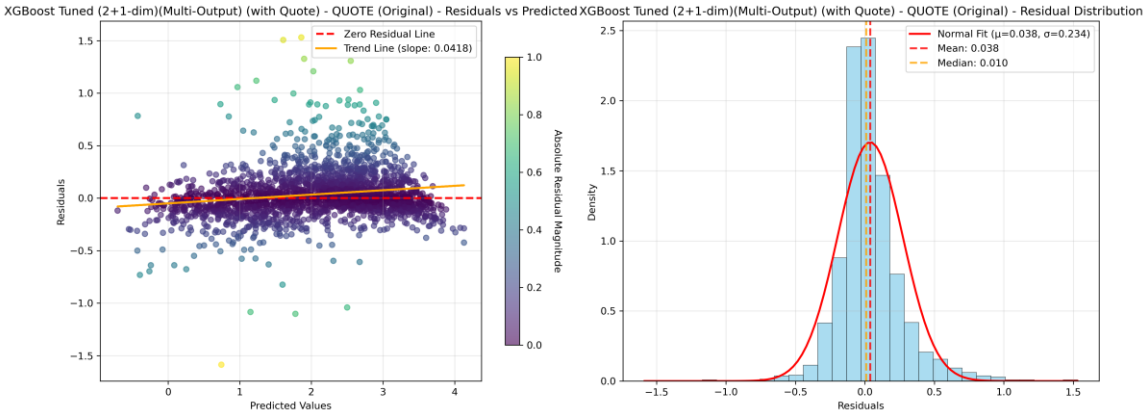

Figure 13: XGBoost: Note the lower standard deviation of the residuals compared to plain random forest.

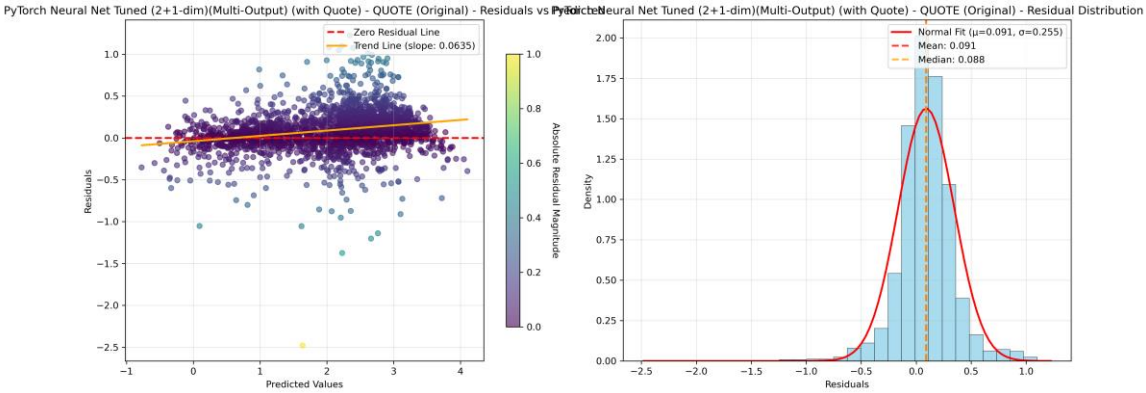

Figure 14: Neural networks perform comparably to XGBoost.

4.5. Model Explainability (XAI)

For a model to be used in a regulatory context like Solvency II, it must satisfy the "Use Test", making interpretability a critical requirement: being able to interpret results will allow making informed management decisions. To assess model interpretability, we applied SHAP (SHapley Additive exPlanations) analysis to the tuned XGBoost model[11].

In this context we can clearly see the advantages of having a multi-output model which predicts SCR and EM simultaneously: we can determine the influence of input parameters separately, and will indeed see below that variables have differing influence on the outputs.

[11] We analyzed variable importance for both XGBoost and MLPRegressor, but results are very similar and differences seems to be of stochastic nature, so we are only showing figures for XGBoost here (MLPRegressor can be found in the Jupyter notebook).

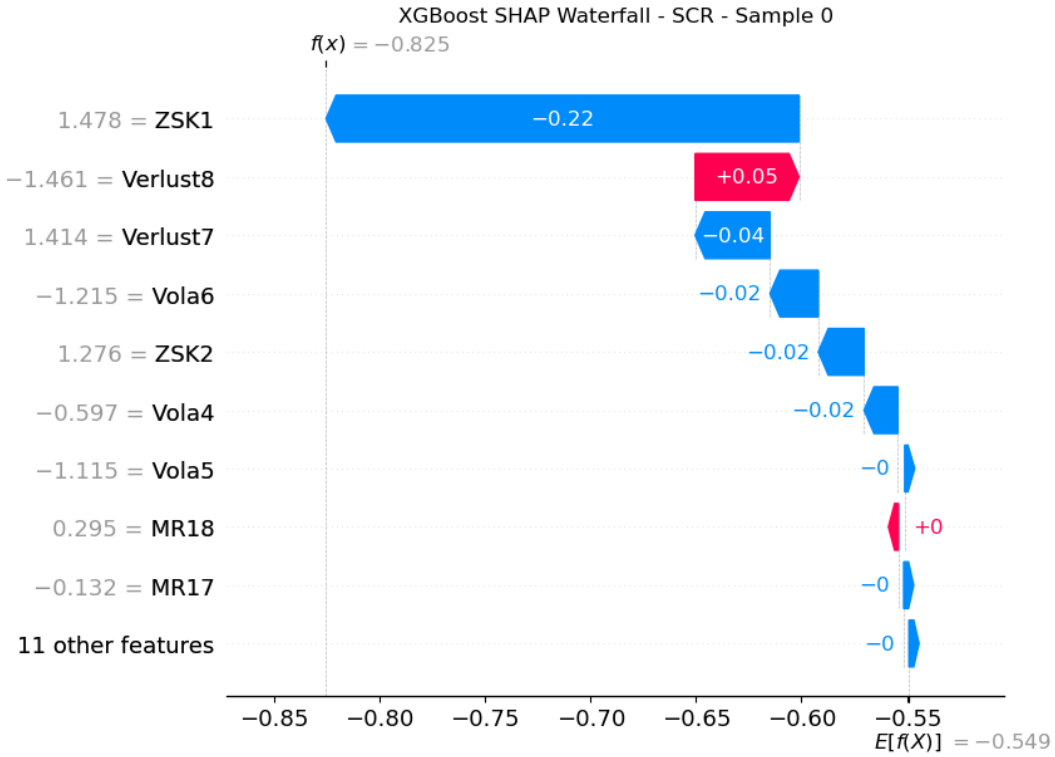

Figure 15: SCR movement can mainly be explained by ZSK1 and Verlust8 and Verlust7.

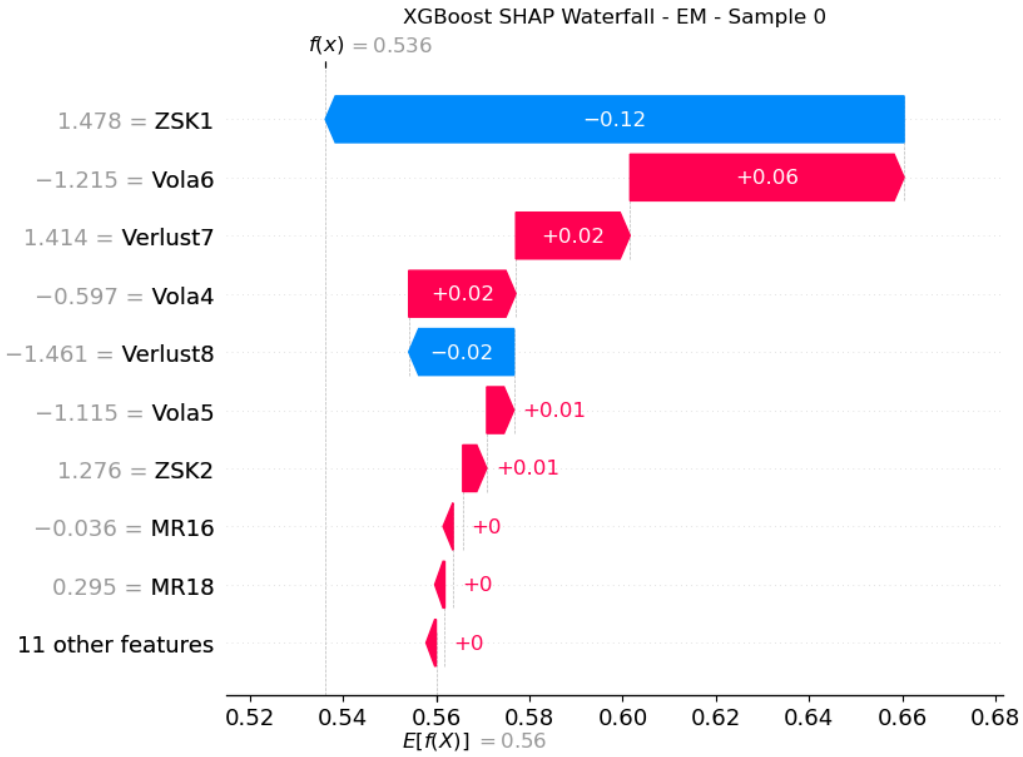

Figure 16: EM is also mainly influenced by ZSK1, but in contrast to SCR, also by Vola6.

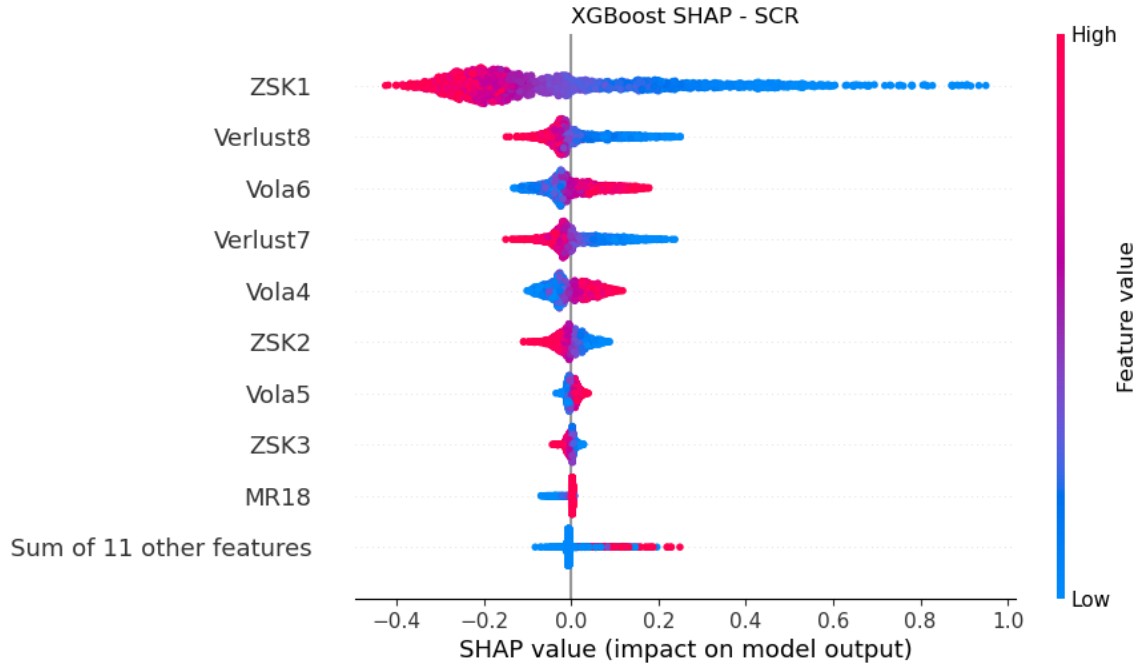

Figure 17: In particular low ZSK1 values will increase SCR.

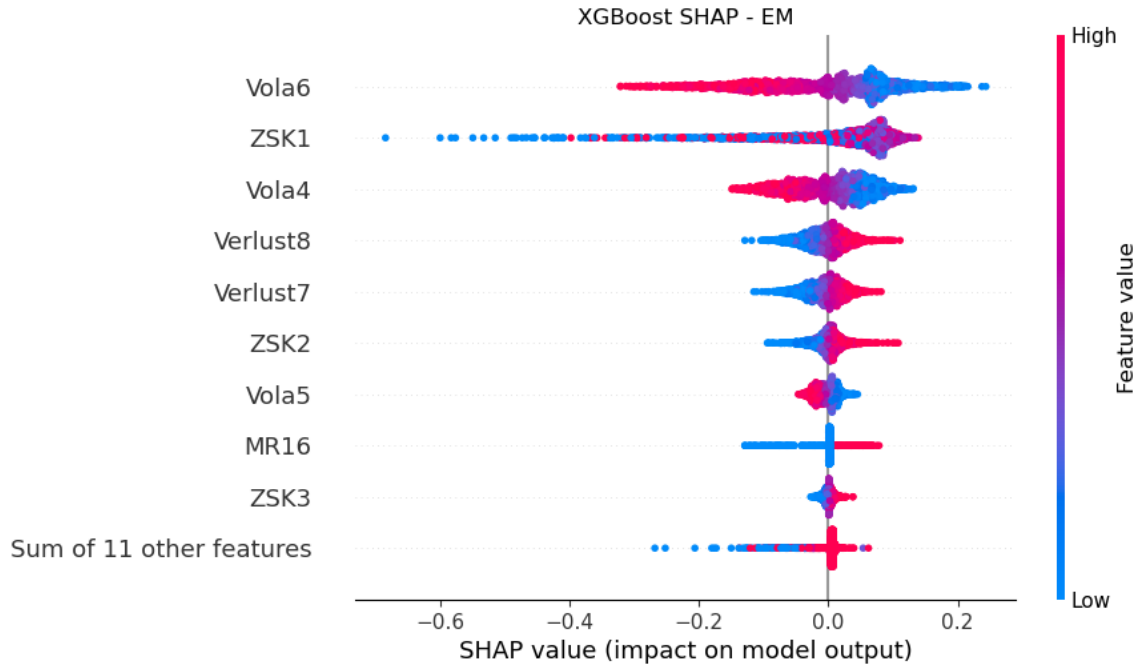

Figure 18: As expected, low ZSK1 values will decrease EM.

The SHAP beeswarm and feature importance plots from the notebook reveal that the model's predictions are driven by a small number of key economic factors, which aligns perfectly with our expectations.

The most influential features were:

1. **ZSK1:** The primary component of the interest rate curve ("Zinsstrukturkurve").
2. **Vola4:** The volatility of the interest rate market.
3. **Vola6:** The volatility of the real estate market.
4. **Verlust7 & Verlust8:** The initial market value losses for stocks and real estate, respectively.

This result is highly plausible. The solvency position of a life insurer, with its long-duration liabilities, is known to be extremely sensitive to the level and volatility of interest rates. The "Management Rules" (MR9–MR20), while included in the model, had a much lower impact, suggesting their effect is secondary to the main economic drivers in this dataset.

4.6. Back Testing

A key objective of this project is to create a "near-casting" tool for use between official reporting periods. To validate this capability, we performed an out-of-time backtest.
The models were trained on the initial synthetic dataset, which represented the market conditions at year-end (t=0). We then obtained a separate backtesting dataset, which contained the actual economic parameters and the actual calculated solvency outputs (EM, SCR, and SR) for the four subsequent quarters (Q1, Q2, Q3, and Q4).
We fed the new economic parameters for these four quarters into our trained models and compared their predictions against the true, traditionally calculated solvency ratios.

model	Q1	Q2	Q3	Q4	RMSE	Method
Actual EM	0.38	0.38	0.35	0.25		
Actual SCR	0.13	0.17	0.23	0.30		
Actual SR	290%	226%	153%	84%		
xgboost_tuned	256%	218%	165%	106%	0.21	EM/SCR
random_forest_multi	247%	221%	159%	138%	0.35	EM/SCR
xgboost_basic	219%	214%	167%	113%	0.40	EM/SCR
nn_mlp	268%	253%	222%	143%	0.49	direct
mlp_multi_tuned	272%	257%	216%	153%	0.50	EM/SCR
torch_nn_multi	287%	267%	231%	152%	0.56	EM/SCR
torch_nn_multi_tuned	280%	269%	237%	157%	0.60	EM/SCR
poly2_multi	214%	216%	206%	178%	0.66	EM/SCR
linear_multi	186%	189%	185%	171%	0.72	EM/SCR
mlp_reg_multi	253%	282%	248%	169%	0.72	EM/SCR
ridge_multi	185%	188%	184%	170%	0.72	EM/SCR

cubic	249%	252%	242%	205%	0.79	direct
dummy	217%	217%	217%	217%	0.83	direct
lasso_cv	238%	242%	239%	223%	0.86	direct
elastic_net	239%	243%	239%	224%	0.87	direct
ridge_cv	239%	243%	239%	224%	0.87	direct
linear_mse	239%	243%	240%	224%	0.87	direct
quadratic	255%	260%	253%	226%	0.90	direct

Table 5: Results from back-testing.

As we can see, actual ratios decreased dramatically from high 290% to well below 100%[12]. In particular, regression models struggled to capture the downward trend.

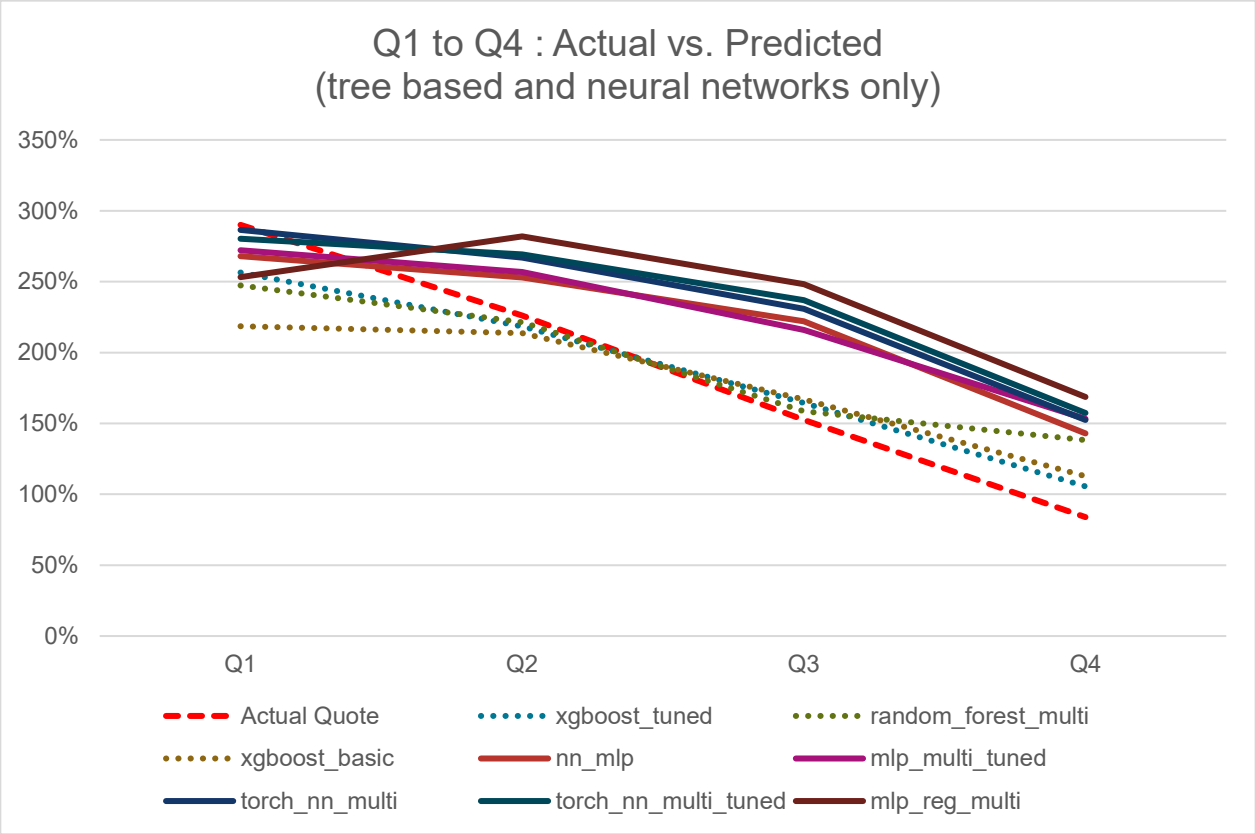

Figure 19: Comparison of SR movement, actual vs. model output.

Although nearly all models were predicting lower Q4 figures, every model stayed above 100%, and only the XGBoost models were showing a Solvency ratio converging to the 100% threshold. This is especially important when one puts this in contrast to the neural networks. Both XGBoost and neural networks worked quite well on the test set, but neural networks were only hesitantly following the dip starting from Q2 on.

[12] Raw figures. The actual ratio turned out to be well above 100% due to the use of transitional measures in accordance with Solvency II.

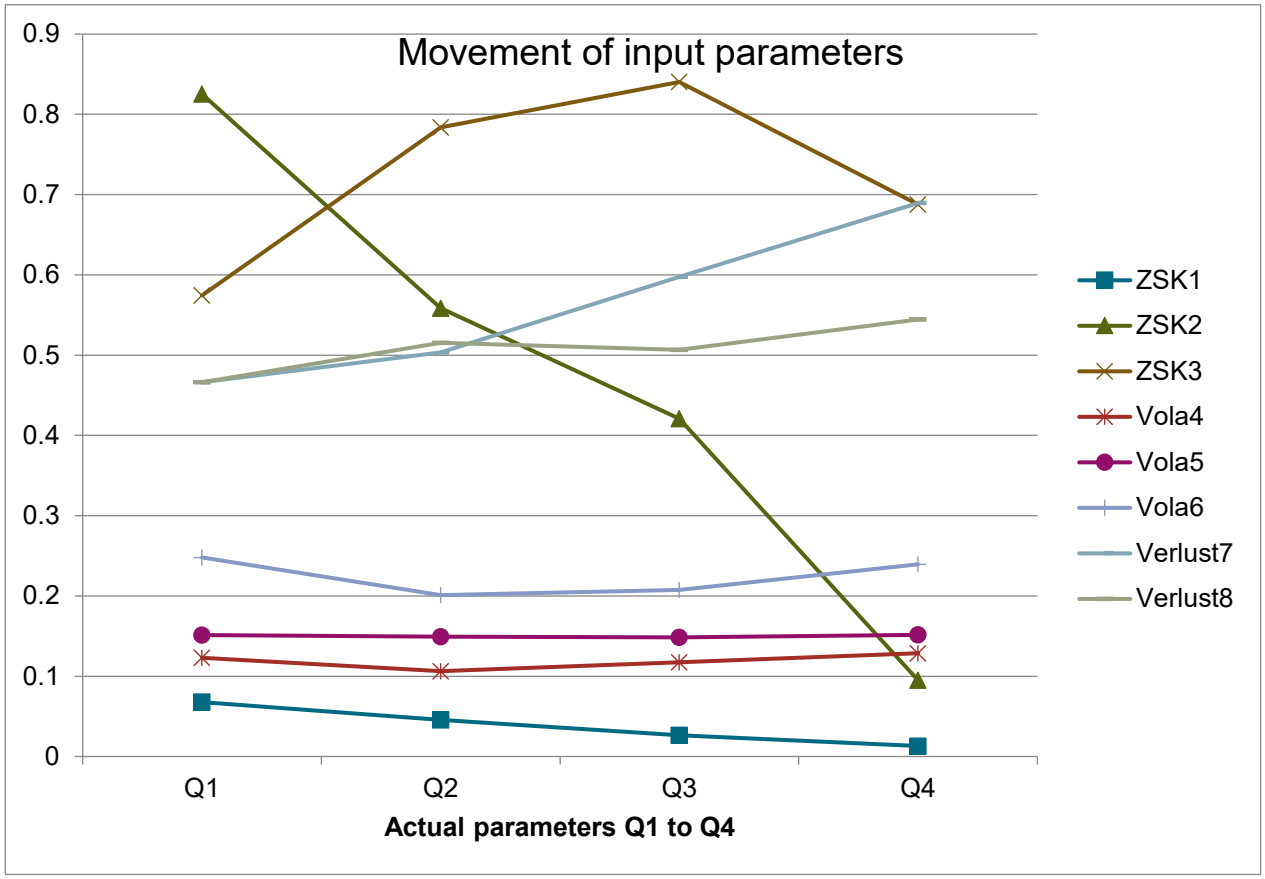

Figure 20: Movement of input parameters (MR9-MR20 did not change).

Looking at the economic input parameters, we see that the biggest change was due to ZSK2, which moved from the top of the unit interval to the very bottom, a very strong change. The default value “Verlust7” (equity loss) also increased and ZSK3 was volatile.

Looking at the SHAP waterfall though, we see the most influential parameter by far is ZSK1 for both EM and SCR. ZSK 1 dropped to nearly zero (from 0.068 to 0.013). According to the BeeSwarm plots (blue dots), this will increase SCR and, at the same time, decrease EM. In fact, SCR increased by 31% and EM decreased by 33% (Q4 vs. Q1).

4.7. Model Governance

- The model was developed on the GitHub platform. We had separate branches by user, and a main branch where the final model was committed to. New features were added after regular conference calls and we performed technical reviews as well as validations as subject matter experts.
- If the model were to be used in practice, an important aspect is to assess when the model needs to be retrained. Apart from triggers resulting from the underlying business, as mentioned in 2.3, this could in particular be necessary if input features deviate very strongly from their initial values. A dashboard that shows predicted SR would also have to display the movement of the input features and needs to give an assessment of the strength of the prediction. This assessment could be based on the combination of the importance of each input parameter and its drift.
- For Solvency II purposes, the SR needs to be (exactly) calculated on a quarterly basis, so one could retrain the model on the same cadence. Otherwise, one could decide on an annual frequency and use the quarterly results as a quality control of the model.
- As mentioned in other parts of the document, further research needs to be performed on the temporal stability of the model. This would also need to cover model selection, model architecture and model explainability[13].

4.8. Discussion of Findings

The experiments demonstrate that classical linear and regularised regressions are insufficient to reproduce the complex nonlinear mapping from market factors to solvency outcomes.
The quadratic model is already better than linear regressions but struggles with non-linearities at the low SR regimes.
In contrast, tree-based ensembles (Random Forest, XGBoost) and neural networks deliver markedly superior accuracy.
Between them, XGBoost offers slightly higher stability and performed better in the back-test, making it a pragmatic choice for production use under Solvency II.
Neural networks, however, achieve comparable or marginally lower RMSE and demonstrate good bias handling even in distressed (low-solvency) regimes, suggesting potential for further improvement with larger data sets or improved architectures.
Direct prediction of the SR performed similarly to (and sometimes better than) the indirect EM/SCR approach, though the latter is conceptually attractive as it explains the relationship SR = EM/SCR.
Combining both outputs in a joint multi-output network proved effective, in most cases improving accuracy for EM and SCR due to their strong correlation.

5.Conclusion

This analysis shows that the idea of predicting solvency ratios using economic market inputs is feasible and provides meaningful outcomes. Both XGBoost and neural networks are good model candidates.

[13] I. e. model type (XGBoost vs. NN); depth and width of NN; which input variables will be most important.

It is evident that non-linearities in the underlying process make predictions more difficult. In particular, if market inputs change substantially, model performance will be impaired. In relatively stable markets, a forecast should be quite accurate, especially if the most important input parameter, ZSK1, remains within a narrow band.

6.Potential areas of improvement

- We should have normalized both EM and SCR using the same min-max function. This would then have naturally preserved the EM/SCR ratio and thus given the exact Solvency ratio without a need for an inverse transformation.
- We could explore approaches where we predict EM, SCR and SR, and compare the predicted SR with the predicted calculated (EM/SCR). An additional idea is to introduce a penalty if these two quantities are too far away, like a L^2 norm $(SR - EM/SCR)^2$.
- We could invest more into fine-tuning XGBoost and neural networks.
- If possible, we should do further model development by using training data as of different points of time. The goal is to find the best model that equally works for longer time spans (not just for one point in time).

7.Acknowledgements

Parts of the text were reviewed with the support of AI-based language tools (ChatGPT and Google Gemini) for clarity and grammar. Those tools were also used for code suggestions in the Jupyter notebook. All analysis, methodology, and conclusions are the author's.

8.Bibliography

James, Gareth, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. *An Introduction to Statistical Learning: With Applications in Python*. Cham: Springer, 2023.

Lones, Michael A. "Avoiding Common Machine Learning Pitfalls." *Patterns* 5 (October 11, 2024). https://doi.org/10.1016/j.patter.2024.101046

Appendix

Detailed comparison of model performance

```
================================================================================================
QUOTE MODELS - DETAILED COMPARISON
================================================================================================
Sorted by Validation RMSE (Normalized Scale):
 Rank                          Model_Name        Model_Type  Val_RMSE_Norm  Test_RMSE_Norm  Test_R2_Norm Quote_Calc_Method
    1 XGBoost Tuned (2+1-dim)(Multi-Ou...     XGBoost Joint         0.0807          0.0802        0.9453      SCR+EM→Quote
    2 XGBoost Basic (2+1-dim)(Multi-Ou...     XGBoost Joint         0.0851          0.0832        0.9411      SCR+EM→Quote
    3 PyTorch Neural Net (2+1-dim)(Mul...      multi_output         0.0896          0.0831        0.9413      SCR+EM→Quote
    4 Neural Net (MLPRegressor)(1-/2+1... Individual Direct         0.0955          0.0838        0.9402            Direct
    5 PyTorch Neural Net Tuned (2+1-di...      multi_output         0.0989          0.0888        0.9328      SCR+EM→Quote
    6 Random Forest (2+1-dim)(Multi-Ou...      multi_output         0.1015          0.1033        0.9091      SCR+EM→Quote
    7         Quadratic (Degree 2)(1-dim) Individual Direct         0.1108          0.1062        0.9040            Direct
    8 MLPRegressor (2+1-dim)(Multi-Out...      multi_output         0.1273          0.1256        0.8657      SCR+EM→Quote
    9 Quadratic (2+1-dim)(Multi-Output...      multi_output         0.1541          0.1632        0.7734      SCR+EM→Quote
   10 MLPRegressor Tuned (2+1-dim)(Mul...      multi_output         0.1822          0.1055        0.9053      SCR+EM→Quote
   11             Cubic (Degree 3)(1-dim) Individual Direct         0.1822          0.1588        0.7853            Direct
   12               Ridge with CV (1-dim) Individual Direct         0.2012          0.1907        0.6904            Direct
   13                 Elastic Net (1-dim) Individual Direct         0.2012          0.1907        0.6903            Direct
   14      Linear Regression (MSE)(1-dim) Individual Direct         0.2012          0.1907        0.6905            Direct
   15               Lasso with CV (1-dim) Individual Direct         0.2013          0.1910        0.6894            Direct
   16                    Dummy Regressor Individual Direct         0.3420          0.3429       -0.0006            Direct
   17 Ridge Regression (2+1-dim)(Multi...      multi_output         1.2388          4.4102     -164.5481      SCR+EM→Quote
   18 Linear Regression (2+1-dim)(Mult...      multi_output         1.2853          6.6605     -376.5974      SCR+EM→Quote
```

Figure 21: Model performance (SR).

```
================================================================================================
SCR MODELS - DETAILED COMPARISON
================================================================================================
Sorted by Validation RMSE (Normalized Scale):
 Rank                          Model_Name             Model_Type  Val_RMSE_Norm  Test_RMSE_Norm  Test_R2_Norm Quote_Calc_Method
    1 MLPRegressor Tuned (2+1-dim)(Mul...  multi_output Component         0.0733          0.0823        0.9367               N/A
    2 XGBoost Tuned (2+1-dim)(Multi-Ou... XGBoost Joint Component         0.0764          0.0840        0.9341               N/A
    3 PyTorch Neural Net (2+1-dim)(Mul...  multi_output Component         0.0765          0.0856        0.9316               N/A
    4 XGBoost Basic (2+1-dim)(Multi-Ou... XGBoost Joint Component         0.0775          0.0855        0.9317               N/A
    5 MLPRegressor (2+1-dim)(Multi-Out...  multi_output Component         0.0784          0.0814        0.9381               N/A
    6 Neural Net (PyTorch MLP)(2+1 dim...  Multi-Output Component         0.0787          0.0860        0.9309               N/A
    7 PyTorch Neural Net Tuned (2+1-di...  multi_output Component         0.0812          0.0872        0.9289               N/A
    8 Neural Net (MLPRegressor)(1-/2+1...  Multi-Output Component         0.0857          0.0991        0.9083               N/A
    9 Neural Net (MLPRegressor)(1-/2+1...       Individual Direct         0.0886          0.0932        0.9189               N/A
   10 Random Forest (2+1-dim)(Multi-Ou...  multi_output Component         0.0950          0.0998        0.9070               N/A
   11 Quadratic (2+1-dim)(Multi-Output...  multi_output Component         0.0997          0.1053        0.8964               N/A
   12         Quadratic (Degree 2)(1-dim)       Individual Direct         0.0997          0.1053        0.8964               N/A
   13 Quadratic (Degree 2)(1-dim) (Mul...  Multi-Output Component         0.0997          0.1053        0.8964               N/A
   14 Cubic (Degree 3)(1-dim) (Multi SCR)  Multi-Output Component         0.1542          0.1397        0.8176               N/A
   15             Cubic (Degree 3)(1-dim)       Individual Direct         0.1542          0.1397        0.8176               N/A
   16 Linear Regression (2+1-dim)(Mult...  multi_output Component         0.1686          0.1674        0.7381               N/A
   17      Linear Regression (MSE)(1-dim)       Individual Direct         0.1686          0.1674        0.7381               N/A
   18 Linear Regression (MSE)(1-dim) (...  Multi-Output Component         0.1686          0.1674        0.7381               N/A
   19                 Elastic Net (1-dim)       Individual Direct         0.1687          0.1675        0.7380               N/A
   20   Ridge with CV (1-dim) (Multi SCR)  Multi-Output Component         0.1687          0.1675        0.7380               N/A
   21               Ridge with CV (1-dim)       Individual Direct         0.1687          0.1675        0.7380               N/A
   22 Ridge Regression (2+1-dim)(Multi...  multi_output Component         0.1687          0.1675        0.7380               N/A
   23   Lasso with CV (1-dim) (Multi SCR)  Multi-Output Component         0.1687          0.1677        0.7372               N/A
   24               Lasso with CV (1-dim)       Individual Direct         0.1687          0.1677        0.7372               N/A
   25     Elastic Net (1-dim) (Multi SCR)  Multi-Output Component         0.3229          0.3275       -0.0017               N/A
   26         Dummy Regressor (Multi SCR)  Multi-Output Component         0.3229          0.3275       -0.0017               N/A
   27                     Dummy Regressor       Individual Direct         0.3229          0.3275       -0.0017               N/A
```

Figure 22: Model performance (SCR).

```
================================================================================================
EM MODELS - DETAILED COMPARISON
================================================================================================
Sorted by Validation RMSE (Normalized Scale):
 Rank                          Model_Name              Model_Type  Val_RMSE_Norm  Test_RMSE_Norm  Test_R2_Norm Quote_Calc_Method
    1 MLPRegressor Tuned (2+1-dim)(Mul...  multi_output Component         0.0543          0.0541        0.9306               N/A
    2 XGBoost Tuned (2+1-dim)(Multi-Ou... XGBoost Joint Component         0.0555          0.0562        0.9251               N/A
    3 PyTorch Neural Net (2+1-dim)(Mul...  multi_output Component         0.0561          0.0587        0.9184               N/A
    4 XGBoost Basic (2+1-dim)(Multi-Ou... XGBoost Joint Component         0.0571          0.0583        0.9195               N/A
    5 Neural Net (PyTorch MLP)(2+1 dim...  Multi-Output Component         0.0587          0.0590        0.9174               N/A
    6 PyTorch Neural Net Tuned (2+1-di...  multi_output Component         0.0622          0.0642        0.9023               N/A
    7 Neural Net (MLPRegressor)(1-/2+1...  Multi-Output Component         0.0641          0.0712        0.8799               N/A
    8 MLPRegressor (2+1-dim)(Multi-Out...  multi_output Component         0.0651          0.0623        0.9079               N/A
    9         Quadratic (Degree 2)(1-dim)       Individual Direct         0.0763          0.0746        0.8682               N/A
   10 Quadratic (Degree 2)(1-dim) (Mul...  Multi-Output Component         0.0763          0.0746        0.8682               N/A
   11 Quadratic (2+1-dim)(Multi-Output...  multi_output Component         0.0763          0.0746        0.8682               N/A
   12 Random Forest (2+1-dim)(Multi-Ou...  multi_output Component         0.0783          0.0799        0.8489               N/A
   13 Neural Net (MLPRegressor)(1-/2+1...       Individual Direct         0.0822          0.0745        0.8685               N/A
   14  Cubic (Degree 3)(1-dim) (Multi EM)  Multi-Output Component         0.1425          0.1173        0.6740               N/A
   15           Cubic (Degree 3)(1-dim)       Individual Direct         0.1425          0.1173        0.6740               N/A
   16      Linear Regression (MSE)(1-dim)       Individual Direct         0.1556          0.1525        0.4484               N/A
   17 Linear Regression (MSE)(1-dim) (...  Multi-Output Component         0.1556          0.1525        0.4484               N/A
   18 Linear Regression (2+1-dim)(Mult...  multi_output Component         0.1556          0.1525        0.4484               N/A
   19    Ridge with CV (1-dim) (Multi EM)  Multi-Output Component         0.1556          0.1526        0.4479               N/A
   20             Ridge with CV (1-dim)       Individual Direct         0.1556          0.1526        0.4479               N/A
   21 Ridge Regression (2+1-dim)(Multi...  multi_output Component         0.1556          0.1526        0.4479               N/A
   22               Elastic Net (1-dim)       Individual Direct         0.1557          0.1528        0.4463               N/A
   23    Lasso with CV (1-dim) (Multi EM)  Multi-Output Component         0.1558          0.1529        0.4456               N/A
   24             Lasso with CV (1-dim)       Individual Direct         0.1558          0.1529        0.4456               N/A
   25         Dummy Regressor (Multi EM)  Multi-Output Component         0.2067          0.2054       -0.0001               N/A
   26                   Dummy Regressor       Individual Direct         0.2067          0.2054       -0.0001               N/A
   27      Elastic Net (1-dim) (Multi EM)  Multi-Output Component         0.2067          0.2054       -0.0001               N/A
```

Figure 23: Model performance (EM).

Examples for optimized model architectures

Fine-tuned xgboost model

```
'xgboost_tuned': {
    'name': 'XGBoost Tuned (Multi-Output)',
    'model': xgb.XGBRegressor(
        objective='reg:squarederror',
        n_estimators=200,
        learning_rate=0.05,
        max_depth=8,
        min_child_weight=3,
        subsample=0.8,
        colsample_bytree=0.8,
        reg_alpha=0.1,
        reg_lambda=1.0,
        random_state=RANDOM_STATE,
        n_jobs=-1
    ),
    'loss': 'MSE'
}
```

Best neural networks determined by grid search

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
Best parameters from Randomized Search for PyTorch Multi: {'optimizer__weight_decay': 0.001, 'optimizer__lr': 0.001,
    'module__hidden_sizes': (256, 128), 'module__dropout': 0.2, 'module__batchnorm': False, 'max_epochs': 300}


Fitting 5 folds for each of 20 candidates, totalling 100 fits
Best parameters from Randomized Search for MLPRegressor: {'max_iter': 500, 'learning_rate_init': 0.01, 'hidden_layer_sizes':
    (256, 128, 64), 'alpha': 0.001, 'activation': 'relu'}
```