Tracking primordial gravitational waves with machine learning

Renato C. Santos[1]

[1] Instituto Kunumi
Belo Horizonte, MG – Brazil

renato.costa87@gmail.com

Abstract. *We employ a neural network (NN) to predict the tensor-to-scalar ratio (r), a key parameter quantifying primordial gravitational waves (PGWs), directly from the CMBR B-mode polarization power spectra ($\mathcal{C}_\ell^B$). By selecting low-order multipoles ($\ell \leq 100$) based on domain knowledge, the NN achieved robust predictive accuracy for r. SHAP analysis revealed that the theoretically critical modes in the $\ell \leq 10$ regime were not primary drivers. This is attributed to the high intrinsic uncertainty of Cosmic Variance at these scales, suggesting the NN prioritizes statistically stable features over the highest signal-to-noise ratio. This work validates machine learning as a powerful tool for extracting crucial information about primordial physics from CMB data.*

Introduction

Shortly after the Big Bang, the Universe underwent an extremely rapid expansion phase known as inflation [Baumann 2009]. In its early moments, the Universe was an extremely hot gas of interacting particles. As the expansion proceeded, this gas cooled, causing fundamental particles to decouple from the thermal plasma. This process sequentially released neutrinos and, finally, photons approximately $380,000$ years after the Big Bang. The relic radiation from this last event is the Cosmic Microwave Background Radiation (CMBR) [Peebles 1993].

The currently accepted cosmological description, the Lambda-Cold Dark Matter (ΛCDM) Model [Collaboration 2020], accurately describes the CMBR as a homogeneous and isotropic blackbody spectrum with an average temperature of $T \approx 2.7K$ and tiny angular temperature fluctuations on the order of $10^{-4}K$.

A key prediction of the Inflationary epoch is the generation of primordial gravitational waves (PGWs)–fluctuations in the spacetime metric [Liddle and Lyth 2000]. While PGWs and primordial neutrinos are yet to be directly measured due to their weak signals, their signatures can be imprinted on the measurable CMBR. CMBR polarization is generally decomposed into E-modes (primarily generated by scalar perturbations, like density fluctuations) and B-modes (which can be uniquely sourced by tensor perturbations, such as PGWs) [Mukhanov 2005]. Detecting PGWs is commonly pursued by searching for their distinct impression on the B-mode polarization ($\mathcal{C}_l^B$). The parameter that quantifies the relative strength of the PGW signal (the "tensor" component) compared to the matter-field fluctuations (the "scalar" component) is the tensor-to-scalar ratio, r.

This project will leverage the power of Machine Learning (ML) Neural Networks [Rumelhart et al. 1986, LeCun et al. 2015, Schmidhuber 2015] to search for signatures

of primordial gravitational waves using simulated CMBR polarization B-mode data. The B-mode power spectrum, $\mathcal{C}_l^B$, describes how the B-mode polarization fluctuations vary across different angular scales, quantified by multipole modes, l.

Our approach involves using these $\mathcal{C}_l^B$ multipole modes as features for a neural network, with the tensor-to-scalar ratio, r, serving as the target. This study aims to demonstrate a novel method for extracting information about primordial gravitational waves using the characteristic B-mode signal, potentially shining new light on the capabilities of cosmological data analysis.

1. Methodology

The cosmological dataset used in this work was kindly provided by Prof. Camila Novaes, from the Instituto Nacional de Pesquisas Espaciais (INPE). While this data recently served as the basis for a related publication ([Santos et al. 2025]), the current project pursues a distinct and expanded direction by focusing on advanced explainability and diagnosing feature importance and selection.

As it was mentioned, the scalar-to-tensor ratio is yet to be measured. However, computer simulations based on the ΛCDM model can infer its value and also the values for the multipole moments, $\mathcal{C}_l$, for the power spectrum.

The target data was provided in a file named $CosmoID\_r\_tau\_As\_1to1000\_concat\_2dLHsampling\_wider.txt$. It contains three columns and 1000 rows. The first column stores values for the tensor-to-scalar ratio, r. The other two columns store values for another important parameters of the standard model of cosmology, τ, and As. However, they will be ignored for the purposes of this project.

The features were organized in files named by $cl\_cmb\_c\{i\}\_m\{j\}.dat$, where i varies from 1 to 1000. For a fixed i, j varies from 1 to 10. Each of these files have two columns and 512 rows. The second column represent values for the multipole moments of the B-mode polarization $\mathcal{C}_l^B$, while the other column have values for the E-mode polariation, $\mathcal{C}_l^E$. In this project, we will focus on multipole moments for the B-mode polaration. The 512 rows are associated to the 512 values for the multipoles, l. These 512 rows will become the 512 features (so we will have to transpose them) and they will be associated to a given value of the target r.

Since i varies from 1 to 1000 and j from 1 to 10, after transposing and joining the 10000 files for the features, one ends up with a dataframe with 10000 rows. However, there are only 1000 different values of r. The idea is that, for each fixed value of i, there will be 10 rows (one for each value of j) associated to the same value of r. It was designed so that it would mimic experimental noise in the data. That will force the model to adapt more closely to real data once it receives it. This is important to have in mind since one has to be careful to feed a neural network later on and avoid target leaking. The division of train, validation and test sets have to be grouped by different values of r, in order to avoid some rows, associated to the same values for the target, falling into the validation/test set and some on the training set.

This project was divided into eight notebooks as follows [RenatodaCostaSantos 2025]:

1 - Notebook 0 - Joining the data: This notebook is in charge of reading the raw data, adding column names, joining and transposing the dataframe for the features, and excluding irrelevant columns. It also adds the target by expanding the target dataframe from 1000 to 10000 rows, so that each row of the dataframe for the features will be linked to a given value of r. Lastly, it separates the final dataframe into training and test sets, saving them in different files.

2 - Notebook 1 - Data exploration: It checks for duplicates, verifies if there are null values, check for the data types and its normalization. This process is done for the training and test sets.

3 - Notebook 2 - Modeling: This notebook builds the first neural network model. It separates the training data into features and target, saves a list of the names of the features in a file (to be used in the explicability notebook later on) and normalizes the data before feeding it into a neural network. All the relevant 510 features are used as input. The number of workers ruled by the n_jobs parameter is set to 1 to avoid some workers crashing while training the neural network (that happened on my machine every time n_jobs was different than 1 but could work in a more robust machine). We used a GroupKFold instance from the scikit-learn library to separate the training and validation sets and avoid target leaking. The training is performed using a grid of parameters that are fed to the HalvingRandomSearch class of the scikit-learn library. This class explores the hyper-parameter space while performing cross-validation at the same time. The HalvingRandomSearch also speeds up the training process by halving the number of candidates (the number of combinations of the hyper-parameter grid that will be used) at each iteration.

Some details about the neural network:

- It contains two hidden layers.
- It uses a Dropout layer after each hidden layer to help avoid overfitting.
  - Note: The dropout rate is a parameter explored by the HalvingRandomSearch instance.
- The number of neurons of the first hidden layer is a parameter to be explored (ranging from 32 to 256).
- The number of neurons of the second hidden layer is fixed at 32.
- It uses a ReLU activation function in the hidden layers and a linear activation function in the output layer (suitable for regression).

The parameter grid explores different values for:

- Number of neurons of the first hidden layer.
- Learning rate; controls how fast the weights are adjusted by the gradient at each step.
- Batch size; the number of rows (samples) to be processed at each step to calculate the gradient and update the weights.
- Dropout rate; the fraction of neurons to be randomly turned off in the hidden layers to regularize the model and help avoid overfitting the data.
- Epochs; the number of times all rows of the training dataset will be presented to the neural network to train it and update the neurons weights.

Once the model is trained, the four metrics below are displayed:

1. Mean Absolute Error (MAE): Calculates the average of the absolute difference between predictions and actual values.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- Interpretation: Represents the average magnitude of the error in the model's predictions, measured in the same units as the target variable.
- Key Feature: It is robust to outliers because it treats all errors linearly.

2. Mean Squared Error (MSE): Calculates the average of the squared differences between predictions and actual values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Interpretation: Represents the average squared error. Its units are the square of the target variable's units.
- Key Feature: It heavily penalizes large errors (outliers) due to the squaring operation.

3. Root Mean Squared Error (RMSE): The square root of the MSE.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- Interpretation: Represents the average magnitude of the error, returned to the original units of the target variable.
- Key Feature: It is sensitive to outliers (like MSE). $RMSE$ will always be greater than or equal to MAE ($RMSE \geq MAE$).

Finally, a data visualization is provided, comparing the 1,000 actual target values against 1,000 aggregated predicted values. Since the original 10,000 predictions were structured such that ten predictions correspond to a single actual value, the mean of each block of ten predictions was calculated to represent the predicted data point, and the standard deviation was used as the error bar.

4 - Notebook 2.1 - Modeling using only the first 100 multipoles: This notebook uses domain knowledge to select only the first 100 lower values for the multipoles and create a neural network machine learning model with them. The steps are exactly as described for notebook 2; the only difference is the selection of the first 100 features associated to the first 100 multipole moments. These modes correspond to the largest angular scales and set the fundamental shape of the CMBR fluctuations.

5 - Notebook 3 - Feature engineering, selection and modeling: This notebook has the same structure of Notebook 2, however it introduces two new ingredients; feature selection and engineering.

After talking to prof. Camila Novaes, the weighted mean of the features was suggested. This feature is a robust indicator of the average power amplitude in the regime of the first acoustic oscillations.

The range from $l = 2$ to $l = 511$ (which goes from large angular scales to the end of the first acoustic peak) summarizes the strength and height of the first large compression/expansion oscillation in the primordial plasma. A larger value for the $media\_ponderada\_Cls$ implies that the power spectrum is, on average, higher, which is generally the case in a universe with greater temperature variation (more initial structure).

For the feature selection process, the SelectKBest class from the scikit-learn library was employed. SelectKBest is a univariate feature selection method that ranks input features based on their individual relationship with the target variable and retains the top k features.

The $f\_regression$ statistical method (F-test for regression) was chosen as the scoring function. This test evaluates the linear dependency between each independent feature and the continuous target variable; a higher F-statistic indicates a stronger linear relationship and, thus, higher relevance for the regression task.

The parameter k was set to the variable $neuronios\_entrada$. This ensured that only the top k features, as determined by the $f\_regression$ score, were passed as input to the neural network model.

The remaining part of the modeling - cross-validation, hyper-parameter exploration, modeling and visualizations - follows the same structure as described for Notebook 2.

6 - Notebook 4 - Feature engineering, selection and modeling: In this notebook, another method for feature selection is used. First, the Pearson correlation between the features and the target is visualized in a heat map. The presence of strong linear correlation of some features with the target suggests the use of a linear regression model to select potential features for modelling. The LASSO, a linear regression that penalizes features with weak linear correlation with the target, was chosen to select the best features according to the linear criteria.

The feature engineering, modeling, metrics calculations and visualizations follow the same structure as described for notebook 3.

7 - Notebook 5 - Testing the models: This notebook aims to test the models generated in notebooks 2, 3 and 4. It starts by pre-processing the data, implementing feature engineering and normalization of the test data before applying the models on it.

The application of the models in the test data follows the same steps in the three cases:

1. Loading the selected features. These are used to filter the test set.

2. Normalization of the test data.

3. The normalized test features are saved in a to be used in the last notebook.

4. Application of the models in the test dataset.

5. Metrics calculations and visualization of the models performance in the test dataset.

8 - Notebook 6 - Explainability: The final step focuses on interpreting the models using SHAP (Shapley Additive exPlanations) values and visualizations. While powerful

machine learning algorithms like neural networks excel at prediction, they often function as ”black boxes”, making it challenging to understand the causality and reasons behind their specific predictions.

SHAP is a unified approach to explaining model predictions based on Shapley values from cooperative game theory. In essence, a SHAP value for a specific feature on a specific prediction is its contribution to the difference between the actual prediction and the average prediction (the baseline).

SHAP values ensure that every input feature gets a fair share of the prediction's outcome. They do this by measuring the individual contribution of each feature across all possible combinations of features, effectively capturing how features interact.

In this notebook, we read the training data's features and use them to create a SHAP KernelExplainer object. This explainer then calculates the SHAP values by applying itself to the test dataset, providing explanations for the model's decisions on unseen data.

With the calculated SHAP values, three plots were generated to interpret the model's behavior:

1. Summary Plot (Feature Importance): This plot aggregates the SHAP values for every feature across all observations. It provides a global view of which features were most important for the model's predictions overall.

2. Force Plot (Local Explanation): This visualization shows the contribution of each feature to a single observation's prediction. It visually demonstrates how positive (driving the prediction higher) and negative (driving the prediction lower) feature contributions push the prediction from the baseline value to its final result.

3. Dependence Plot (Feature Relationship): A scatter plot that illustrates the relationship between a single feature's value and its SHAP value. Crucially, by coloring the points based on a related interacting feature, this plot clearly visualizes feature interactions and their complex relationship with the target variable.

This process was applied to interpret the three models developed in notebooks 2, 3, and 4.

2. Results

All four models were evaluated in the training set using the HalvingRandomSearch class. It explores de hyper-parameter grid and performs cross-validation at the same time. Results four the four metrics pointed out in the methodology section are displayed at Table 1 and Table 2 for the winning model. Similar tables are also available for the remaining models in the Appendix section.

Table 1. Feature selection via domain knowledge

Metric	Value
Number of features used	100
Best Mean R^2	0.9494

Table 2. Best Parameters and Metrics - Feature selection via domain knowledge

Parameter/Metric	Value
Best Parameters	
batch_size	16
epochs	20
dropout_rate	0.108234
learning_rate	0.009702
neurons	189
Best Model Metrics (Training Data)	
MSE (Mean Squared Error)	0.00000166
RMSE (Root Mean Squared Error)	0.00129025
MAE (Mean Absolute Error)	0.00101141
R^2 (Adjusted)	0.9920

The value for the R^2 metric suggest a almost perfect fit. This can be visualized by the scattering plot distribution and its standardized residual displayed at Figure 1. For the remaining models, the scatter plot graphs are available at the Appendix section.

To determine whether the model's performance remains consistent on unseen data and to check for overfitting or underfitting, the model was applied to the test dataset. The results for the best-performing model are shown in Table 3.

Table 3. Performance Evaluation on Test Set (100 features by domain knowledge)

Metric	Value
MSE (Mean Squared Error)	0.00000193
RMSE (Root Mean Squared Error)	0.00138818
MAE (Mean Absolute Error)	0.00111311
R^2 (Adjusted)	0.9906

Table 4 summarizes the performance on the test dataset, clearly linking the metrics to the feature selection method used for each model.

Table 4. Summary of the results in the test dataset for different types of feature selection

Selection Method	Features	R^2 (Adjusted)	MAE	MSE
Domain Knowledge	100	0.9906	0.001113	0.00000193
LASSO	347	0.9749	0.001847	0.00000512
SelectKBest	100	0.9741	0.001729	0.00000530
All features	510	0.9653	0.002158	0.00000708

The results reveal a strong conclusion about the impact of feature selection on the model's ability to generalize.

The model using only 100 features selected by domain knowledge achieved the highest R2 and the lowest errors.

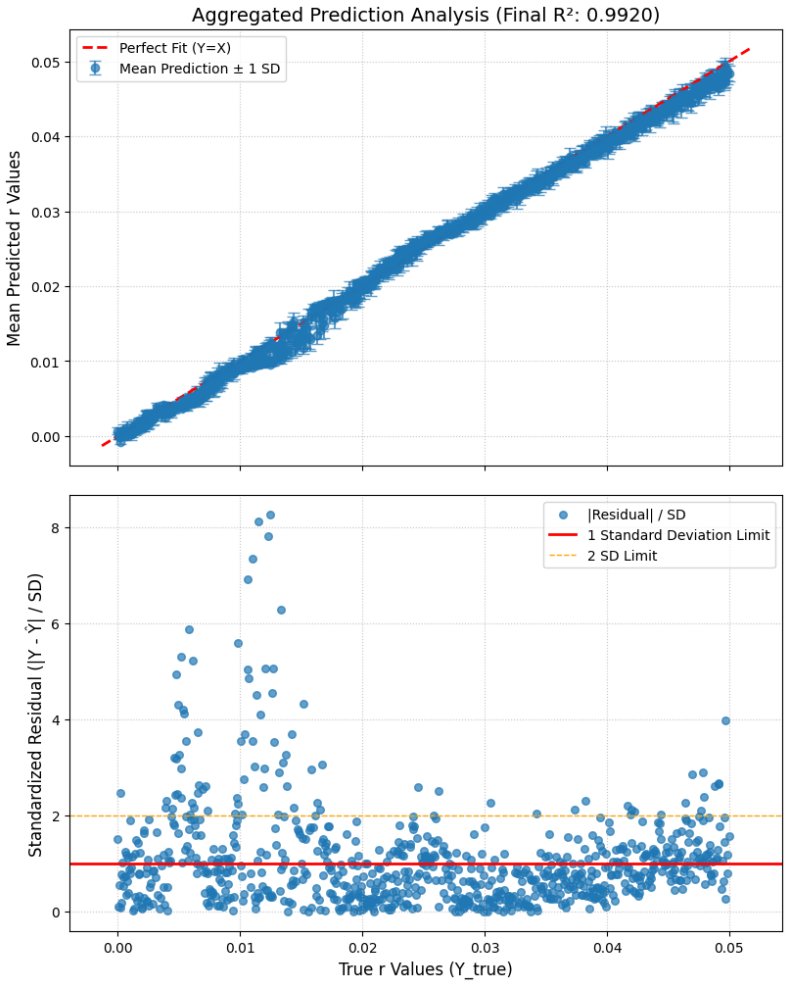

Figure 1. Dispersion and residuals of training data predictions using domain-selected features.

This shows that fewer, high-quality features identified by expert knowledge significantly outperformed models using automated selection or all available data. This indicates that the 100 features capture the most relevant variance while minimizing noise.

Table 5. Comparison for the same number of features but different methods of feature selection)

Selection Method	Features	Test R^2	MAE
Domain Knowledge	100	0.9906	0.0011
SelectKBest	100	0.9741	0.0017

Comparing the two models that used 100 features (Table 5) shows that domain knowledge outperformed the automated SelectKBest. This highlights that simply selecting the top-performing features statistically (SelectKBest) can be less effective than selecting features based on understanding the underlying physical or conceptual process.

The model with all 510 features performed the worst, confirming that irrelevant or redundant features introduce noise and hurt generalization.

3. Discussion

3.1. Domain knowledge, cosmic variance and feature selection

The modeling approach established that employing domain knowledge for feature engineering was key to achieving robust predictions for the tensor-to-scalar ratio, r. Specifically, the best-performing model utilized features confined to the low-order multipole modes ($\ell \leq 102$) of the CMBR B-mode power spectrum ($\mathcal{C}_\ell^B$).

These low-ℓ modes correspond to the largest angular scales in the universe and are dominated by the primordial tensor component (PGWs), making them the most sensitive indicators of r.

However, the specialist consensus is that the tensor signal is most strongly concentrated in the ultra-low ℓ regime ($\ell \leq 10$). Therefore, theory dictates that features in this range should have the highest predictive power for r. Critically, these ultra-low modes are also the most impacted by Cosmic Variance—the unavoidable statistical uncertainty inherent in observing only a single realization of the universe. This phenomenon introduces significant noise into the $\mathcal{C}_\ell$ values at $\ell \leq 10$, which the neural network must navigate.

3.2. Interpreting the SHAP-Driven feature preference

To understand how the model navigates this inherent uncertainty, we analyzed its structure using SHAP (SHapley Additive exPlanations) values (Figure 2). The results revealed a significant deviation from the direct theoretical expectation:

Anomaly in Ultra-Low-ℓ Importance: Instead of the $\ell \leq 10$ modes dominating the prediction, the model assigned the highest global importance to features in the intermediate low-ℓ range (e.g., ℓ between 20 and 80).

Weakness of $\ell \leq 10$: The expected importance of the $\ell \leq 10$ modes was surprisingly low. This suggests the neural network may be down-weighting these features due to the high intrinsic noise imposed by Cosmic Variance, favoring a combination of slightly higher-ℓ modes that offer more statistical stability (i.e., less variance) despite carrying a smaller fractional tensor signal.

3.3. Implications for Cosmological Data Analysis

The observed feature preference has two critical implications for future astrophysical research:

Optimal Feature Synthesis: The ML model is likely capturing a highly complex multi-feature interaction, suggesting that a holistic analysis of the low-ℓ spectrum, rather than focusing on a few individual modes, is necessary to extract the r parameter. The network effectively synthesizes the information across the most predictive angular scales.

Data Set Integrity and Model Diagnosis: If the intermediate ℓ modes are indeed the dominant features, it suggests a need to re-examine how the simulations handle the transition between the ultra-low-ℓ dominance and the onset of the Lensing B-mode, particularly how the statistical uncertainty is propagated. These results highlight the power of Interpretable Machine Learning (IML) as a diagnostic tool to validate and challenge underlying cosmological assumptions or data generation methods.

Low Importance of High-ℓ Modes ($\ell > 80$): The model assigned low importance to multipoles approaching $\ell = 100$. This lack of importance is expected because, in this regime, the primordial $\mathcal{C}_\ell^B$ signal starts to fall significantly, and the spectrum becomes increasingly dominated by the secondary Lensing B-mode, which is not directly correlated with the primordial parameter r. The model correctly identifies these higher-ℓ features as being less predictive of the target variable.

These findings suggest that the most robust prediction for r may not come from the theoretically purest signal ($\ell \leq 10$) but from an optimal trade-off between the signal strength (strongest at $\ell \leq 10$) and the statistical certainty (higher at intermediate ℓ).

3.4. Leveraging Interpretable Machine Learning

The utility of the IML toolkit extends beyond diagnosing the anomaly. The dependence plot (Figure 3) shows that the most important feature contributes positively to the target prediction: a higher value for the feature pushes the predicted r value higher, consistent with the physical expectation that increased B-mode power is directly proportional to a higher r.

The force graph (Figure 4) provides a granular analysis of individual predictions, visually demonstrating how specific features push the prediction above or below the mean. This visualization is critical for establishing trust in the model, allowing cosmologists to trace the prediction for r back to specific multipole measurements, thus linking the ML outcome directly to measurable physics.

4. Conclusion

This project successfully established a robust Machine Learning (ML) framework for predicting the tensor-to-scalar ratio (r) using the low-ℓ regime of the CMBR B-mode spectrum. Beyond prediction, the use of Interpretable Machine Learning (IML) provided critical diagnostic feedback. We confirmed the cosmological premise that the primordial signal is concentrated at low-ℓ but discovered that the most predictive power was not in the theoretically purest ($\ell \leq 10$) range. Instead, the model found an optimal trade-off between the strong signal (at ultra-low ℓ) and the high statistical certainty (at intermediate ℓ). This finding suggests that future B-mode data analysis strategies should incorporate complex, multi-feature synthesis to mitigate the impact of Cosmic Variance, offering concrete, actionable insights for both cosmological simulation development and observational data processing.

References

Baumann, D. (2009). Tasi lectures on inflation. *arXiv e-prints*.

Collaboration, P. (2020). Planck 2018 results. VI. Cosmological parameters. *Astronomy & Astrophysics*, 641:A6.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Liddle, A. R. and Lyth, D. H. (2000). *Cosmological Inflation and Large-Scale Structure*. Cambridge University Press.

Mukhanov, V. (2005). *Physical Foundations of Cosmology*. Cambridge University Press.

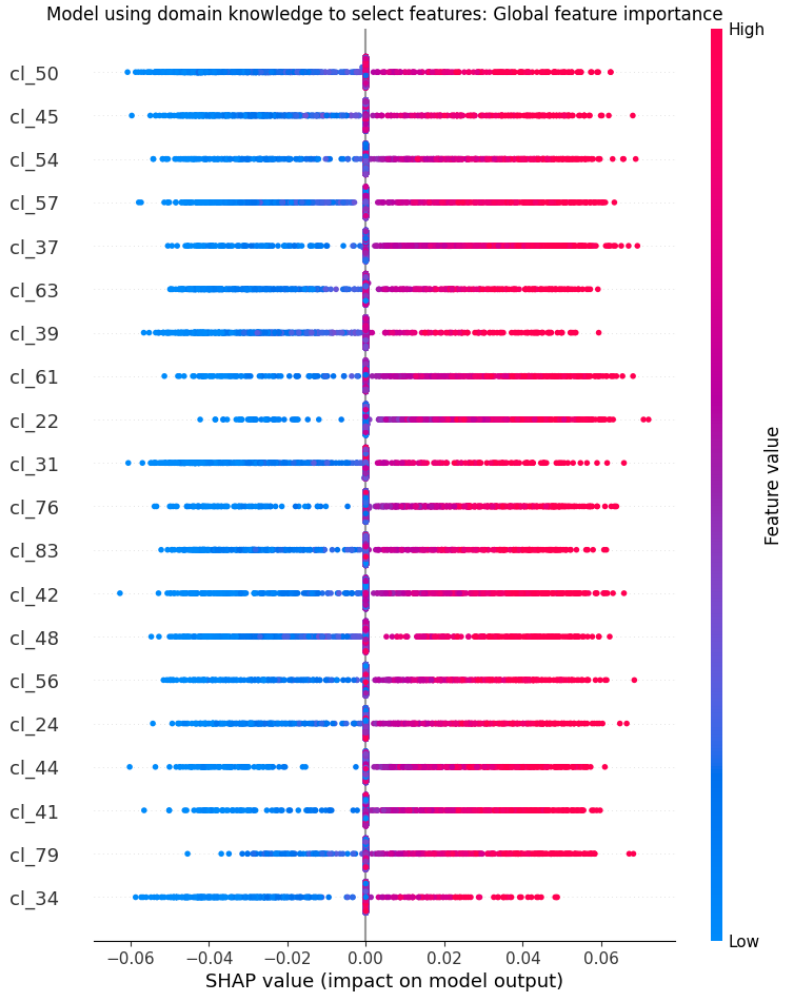

Figure 2. Feature importance according to their shap values.

Peebles, P. J. E. (1993). *Principles of Physical Cosmology*. Princeton University Press.

RenatodaCostaSantos (2025). Em busca de ondas gravitacionais primordiais. `https://github.com/RenatodaCostaSantos/capacitacao`.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

Santos, L., Novaes, C. P., Ferreira, E. G. M., and Baccigalupi, C. (2025). Fast end-to-end framework for cosmological parameter inference from cmb data using machine learning.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

5. Appendix

5.1. Results in the training set

Results for the model built using all features are displayed at Table 6 and Table 7. For the model build using the SelectKBest class from the scikit-learn library the results are

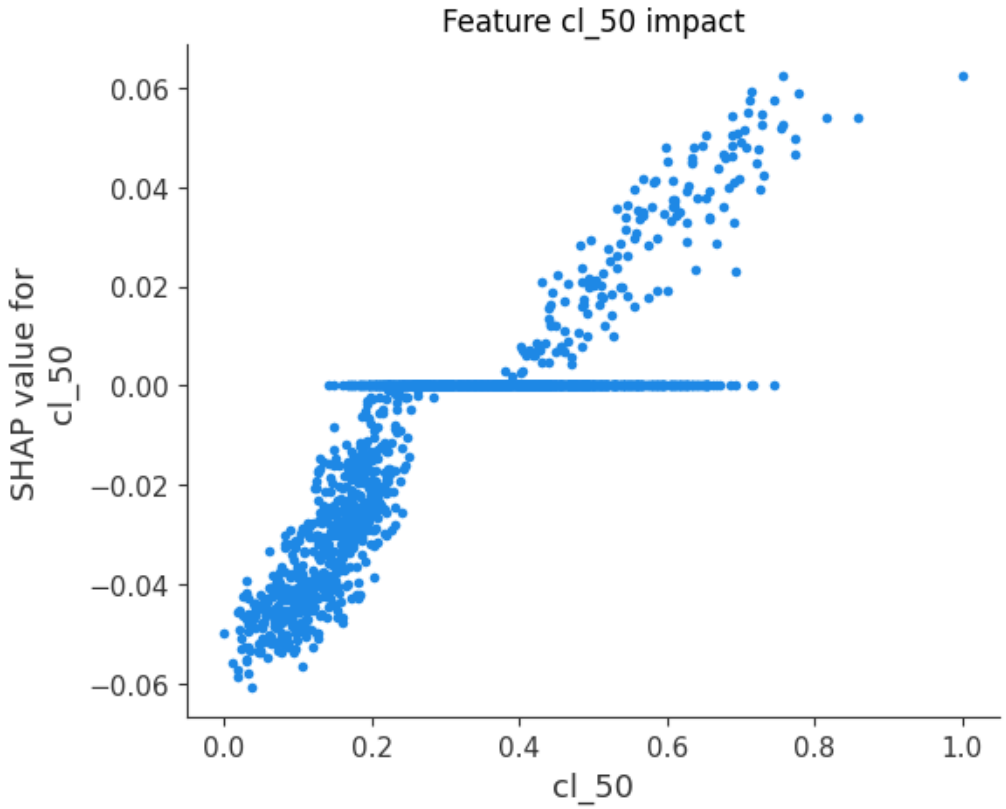

Figure 3. Impact of feature cl_{50} in all observations.

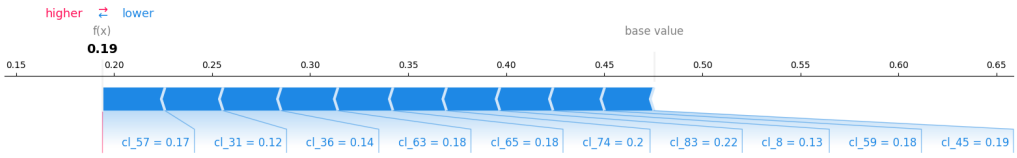

Figure 4. Impact of some features in the prediction of a given observation.

found at Table 8 and Table 9. Lastly, the results obtained from the model using a LASSO regression model to select the best features are displayed at Table 10 and Table 11.

Table 6. General Results - All features

Metric	Value
Number of features used	510
Best Mean R^2	0.9311

The standardized residual and scatter distribution of the training data for the real versus predicted outcomes are displayed at Figure 5, Figure 6 and Figure 7 respectively.

5.2. Results in the test set

As shown in the scatter plots (Figures 8 to 10), the models exhibit similar metrics and dispersion in predictions to the training results. This consistency validates the robustness

Table 7. Parameters and metrics - All features

Parameter/Metric	Parameter/Metric
Best parameters	
batch_size	16
epochs	20
dropout_rate	0.129618
learning_rate	0.003649
neurons	72
Métricas do Melhor Modelo (Treino)	
MSE (Erro Quadrático Médio)	0.00001120
RMSE (Erro Quadrático Médio)	0.00334706
MAE (Erro Absoluto Médio)	0.00256100
R^2 (Ajuste)	0.9465

Table 8. General Results - 100 Features via SelectKBest

Metric	Value
Number of features used	100
Best Mean R^2	0.9110

Table 9. Best Parameters and Metrics - 100 Features via SelectKBest

Parameter/Metric	Value
Best Parameters	
batch_size	16
epochs	20
dropout_rate	0.129618
learning_rate	0.003649
neurons	72
Best Model Metrics (Training Data)	
MSE (Mean Squared Error)	0.00000892
RMSE (Root Mean Squared Error)	0.00298646
MAE (Mean Absolute Error)	0.00219557
R^2 (Adjusted)	0.9574

Table 10. General Results - via LASSO regression selection

Metric	Value
Number of features used	347
Best Mean R^2	0.9413

of the modeling process and confirms the lack of over/underfitting.

Table 11. Best Parameters and Metrics - via LASSO regression selection

Parameter/Metric	Value
Best Parameters	
batch_size	16
epochs	20
dropout_rate	0.129618
learning_rate	0.003649
neurons	72
Best Model Metrics (Training Data)	
MSE (Mean Squared Error)	0.00000958
RMSE (Root Mean Squared Error)	0.00309494
MAE (Mean Absolute Error)	0.00249929
R^2 (Adjusted)	0.9542

Table 12. Performance Evaluation on Test Set (all features)

Metric	**Value**
MSE (Mean Squared Error)	0.00000708
RMSE (Root Mean Squared Error)	0.00266114
MAE (Mean Absolute Error)	0.00215817
R^2 (Adjusted)	0.9653

Table 13. Performance Evaluation on Test Set (SelectKBest for feature selection)

Metric	**Value**
MSE (Mean Squared Error)	0.00000530
RMSE (Root Mean Squared Error)	0.00230246
MAE (Mean Absolute Error)	0.00172895
R^2 (Adjusted)	0.9741

Table 14. Performance Evaluation on Test Set (LASSO for feature selection)

Metric	**Value**
MSE (Mean Squared Error)	0.00000512
RMSE (Root Mean Squared Error)	0.00226329
MAE (Mean Absolute Error)	0.00184686
R^2 (Adjusted)	0.9749

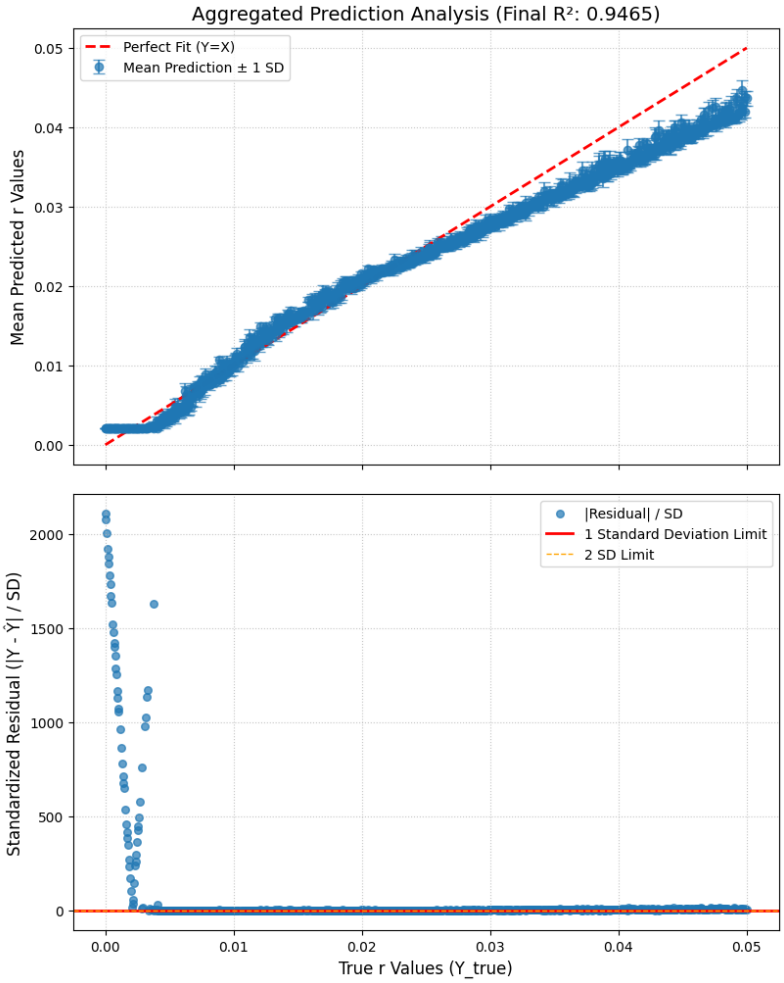

Figure 5. Dispersion and residuals of training data predictions using all features

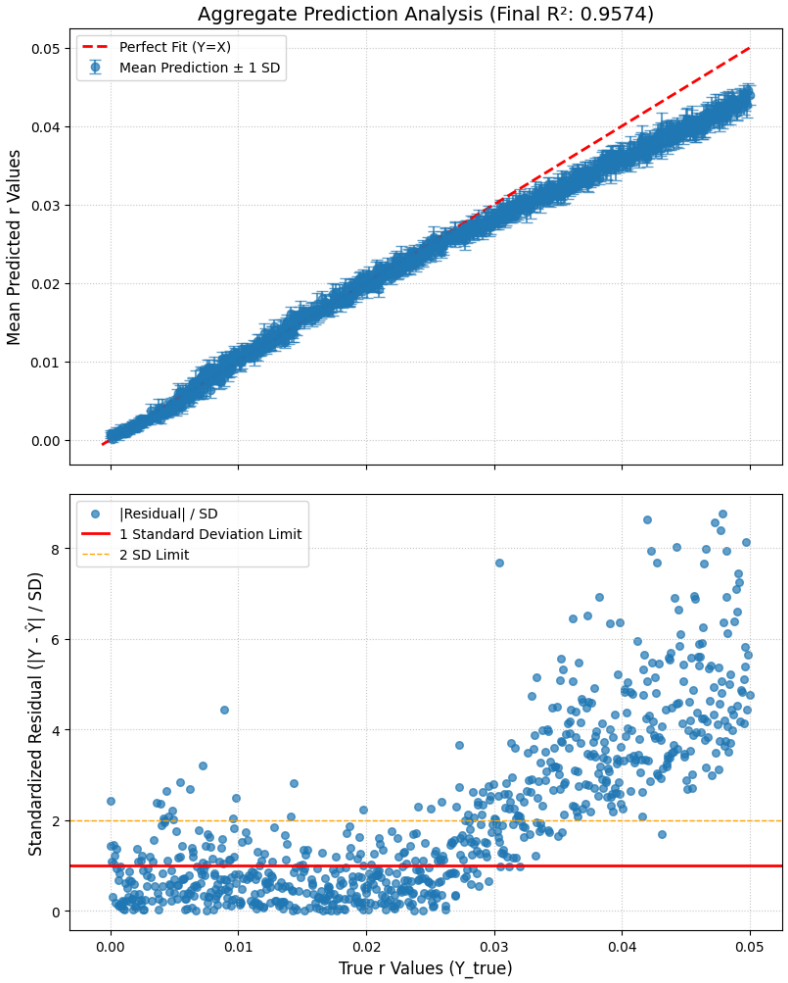

Figure 6. Dispersion and residuals of training data predictions - features selected via SelectKBest

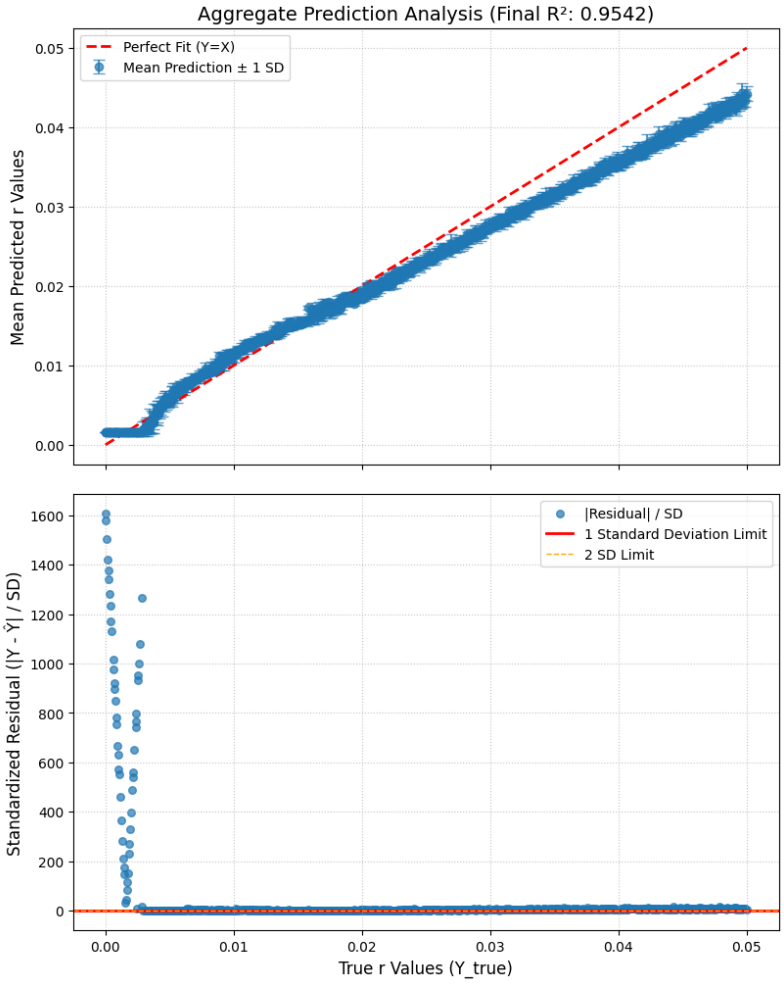

Figure 7. Dispersion and residuals of training data predictions - features selected via LASSO

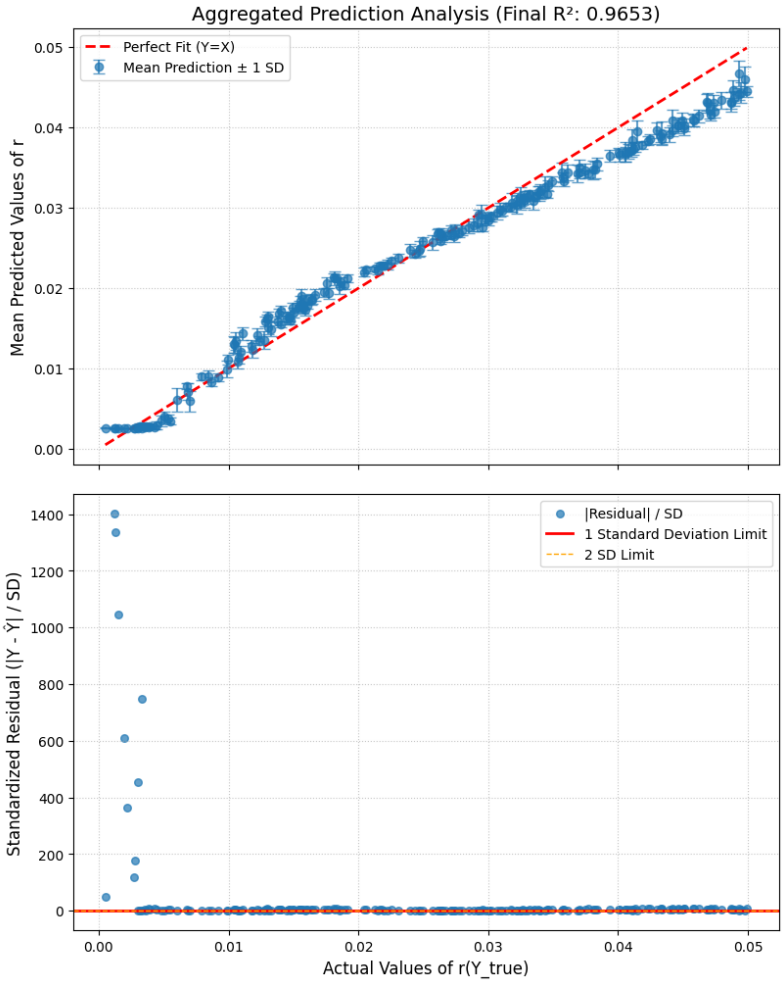

Figure 8. Inference - all features

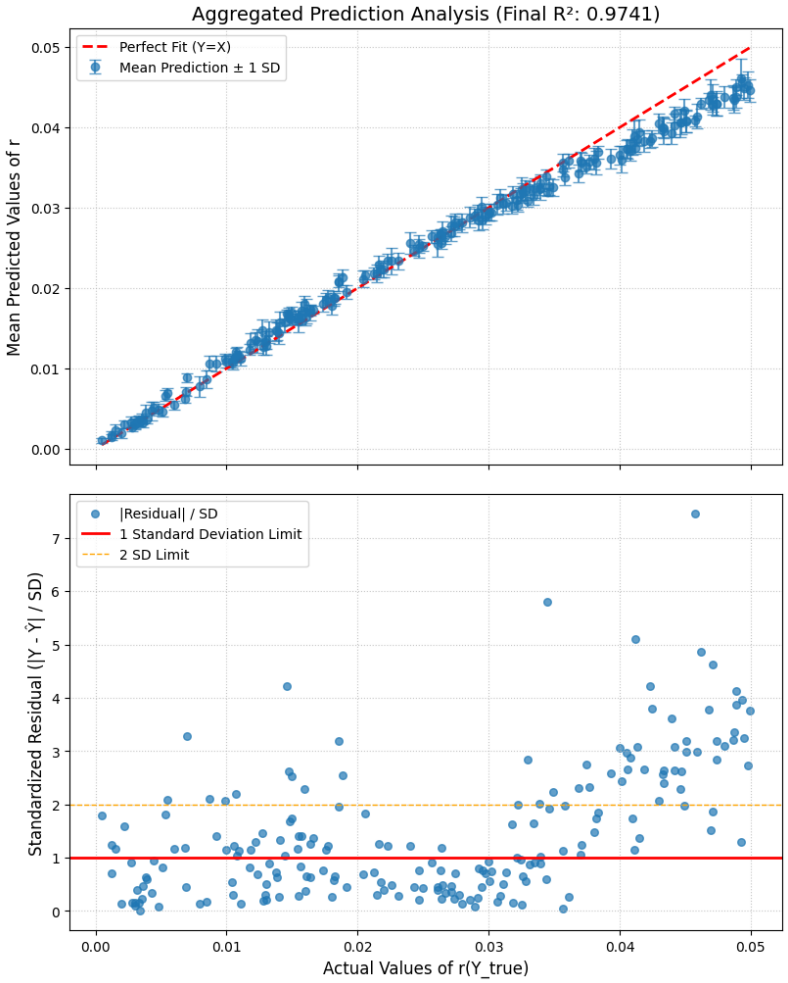

Figure 9. Inference - features select by SelectKBest

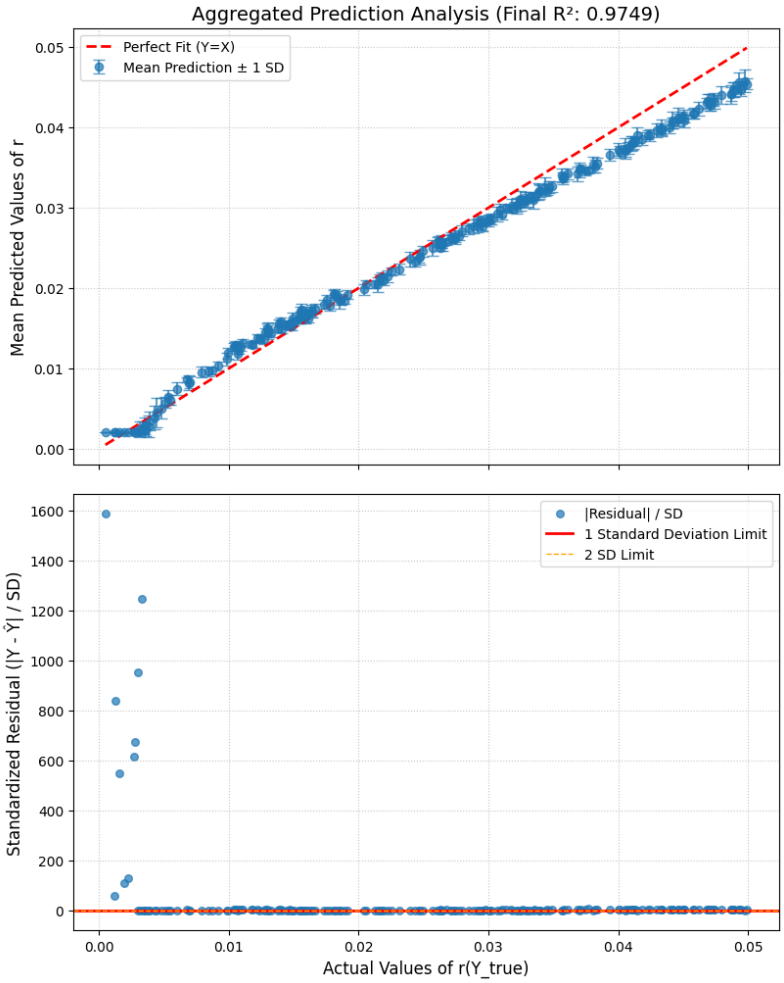

Figure 10. Inference - features selected by LASSO