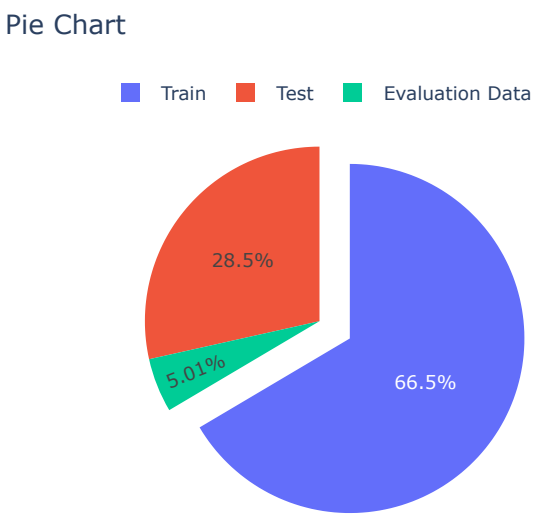
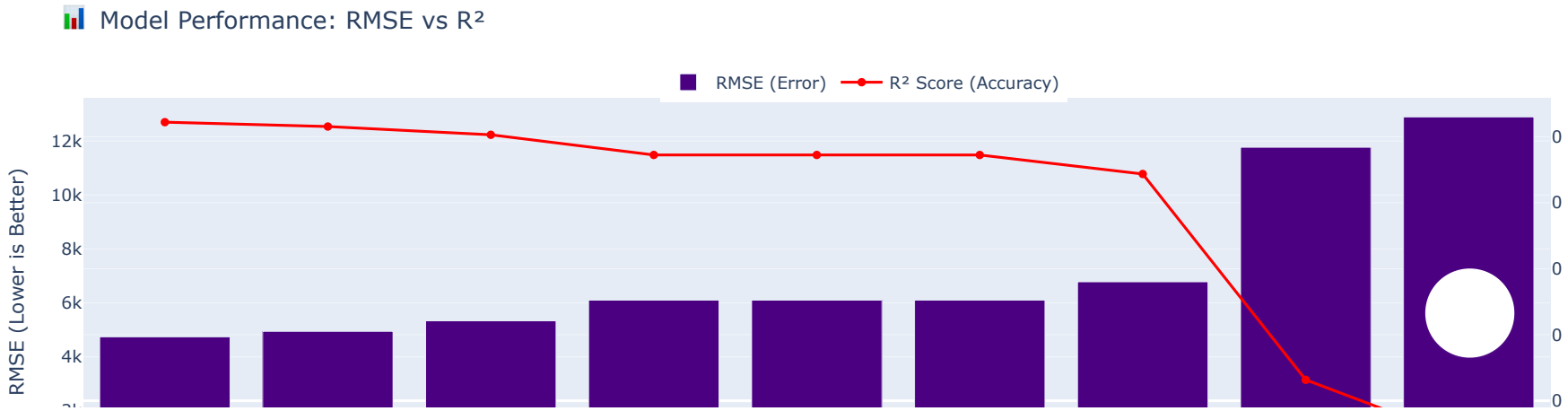
ML Model Build Report

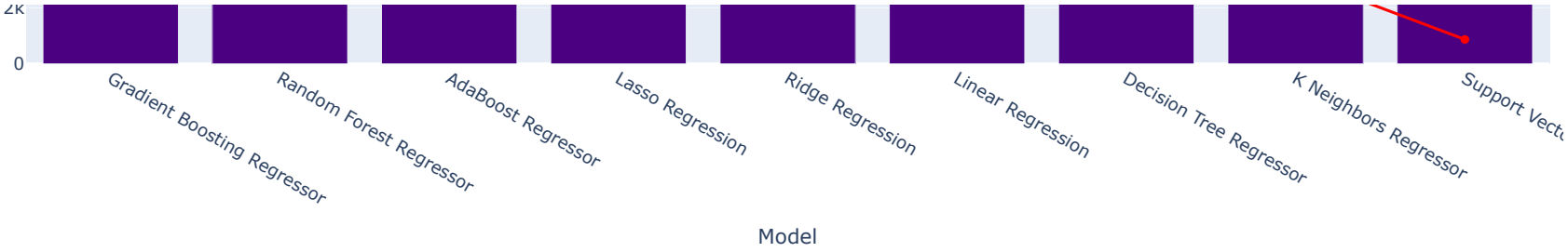
Data Split

Based on the provided data split summary, we can see that the size of the training dataset ('train_data_size') is 889, which is relatively large compared to the sizes of the testing and evaluation datasets ('test_data_size' = 381, 'evaluation_data_size' = 67). This suggests that the model will have plenty of data to learn from during training, which can potentially lead to better performance. However, it's important to note that overfitting can still occur even with a large dataset, so regular evaluation and hyperparameter tuning may be necessary to achieve optimal results.



Initial Model Comparision





Based on the results of the k-fold cross validation with 5 folds, we can see that there are several models that have performed well. However, after analyzing the results further, I have narrowed down the top 3 models that perform the best in terms of R2 value. These models are:

1. Gradient Boosting Regressor - This model has an R2 value of 0.844, which is the highest among all the models. The gradient boosting regressor is a powerful algorithm that combines multiple weak models to create a strong predictive model. It is particularly effective in handling complex data sets with many features and non-linear relationships between the features and the target variable.
2. Random Forest Regressor - This model has an R2 value of 0.8308, which is the second-highest among all the models. The random forest regressor is a popular algorithm that uses a large number of decision trees to estimate the target variable. It is known for its ability to handle high-dimensional data sets and can provide accurate predictions even with a small number of observations.
3. AdaBoost Regressor - This model has an R2 value of 0.8058, which is the third-highest among all the models. The adaptive boosting regressor is a variant of the gradient boosting regressor that uses a different algorithm to combine the weak models. It is known for its ability to handle data sets with a large number of features and can provide accurate predictions even with a small number of observations.

Based on these results, it seems that the gradient boosting regressor, random forest regressor, and adaptive boosting regressor are the top 3 models that perform best in terms of R2 value. However, it's important to note that the performance of a model can vary depending on the specific data set and the evaluation metric used. Therefore, it's always a good idea to evaluate multiple models and compare their performance using different metrics to determine the best model for a particular task.

Fine Tunning Model PreProcessing Parameters

Based on the information provided in the input, here is a summary of the top 3 models and their corresponding preprocessing parameters:

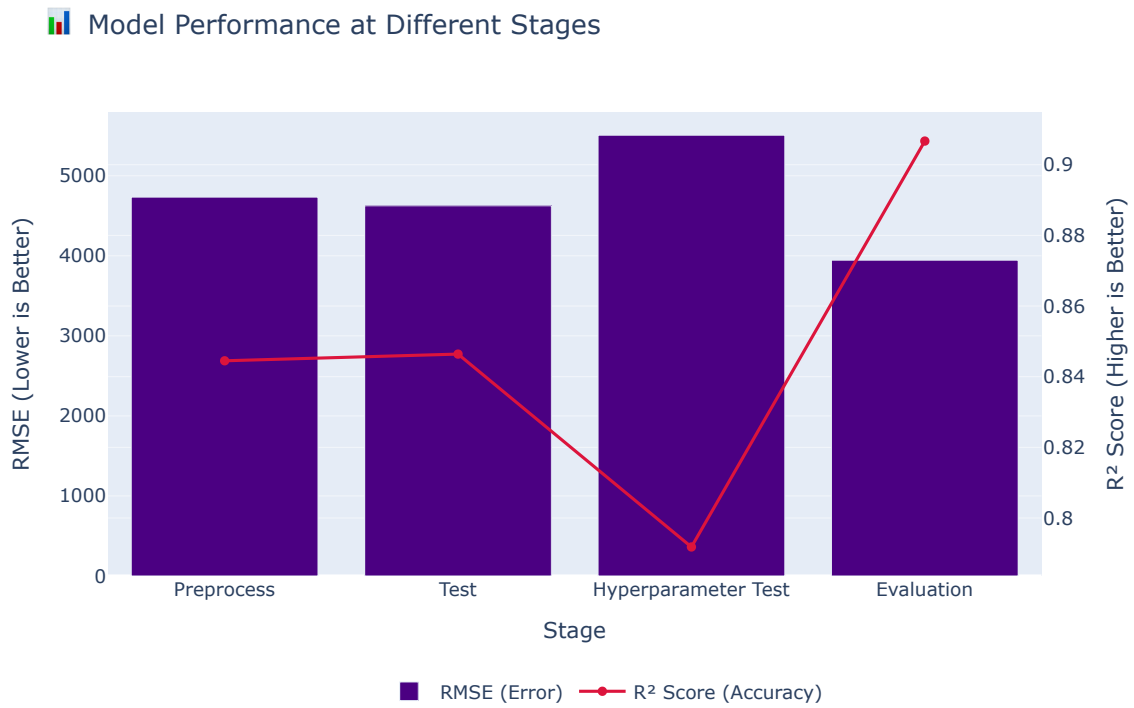
1. Gradient Boosting Regressor:
    - Preprocessing parameters: {'Transformation': True, 'Transform_Method': 'yeo-johnson', 'Normalization': True, 'Normalize_Method': 'robust', 'Outlier': False, 'Multicollinearity': False, 'Target Transformation': False, 'Feature Selection': False}
2. Random Forest Regressor:

- Preprocessing parameters: {'Transformation': True, 'Transform_Method': None, 'Normalization': True, 'Normalize_Method': None, 'Outlier': False, 'Multicollinearity': False, 'Target Transformation': False, 'Feature Selection': False}

3. AdaBoost Regressor:
   - Preprocessing parameters: {'Transformation': True, 'Transform_Method': 'yeo-johnson', 'Normalization': True, 'Normalize_Method': None, 'Outlier': False, 'Multicollinearity': False, 'Target Transformation': False, 'Feature Selection': False}

The top model found is the Gradient Boosting Regressor with preprocessing parameters {'Transformation': True, 'Transform_Method': 'yeo-johnson', 'Normalization': True, 'Normalize_Method': 'robust', 'Outlier': False, 'Multicollinearity': False, 'Target Transformation': False, 'Feature Selection': False}. This model combination achieved the best performance in predicting the target variable.

Final Model Build



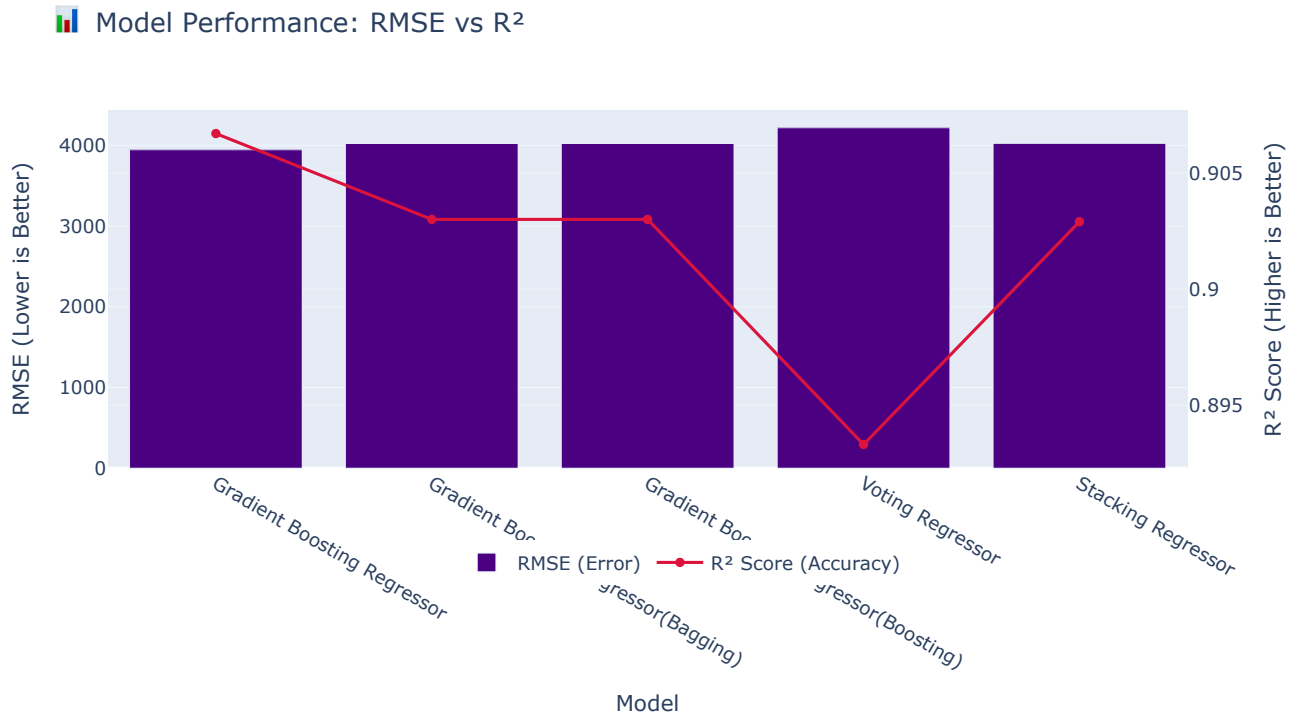
Based on the data provided in the question, the final model built has achieved the following performance metrics:

- R2 score for the training data: 0.8445
- RMSE (Root Mean Squared Error) for the training data: 4732.9684

- R2 score for the test data: 0.8464
- RMSE for the test data: 4624.9717
- R2 score for the hyperparameter tuning results: 0.7918
- RMSE for the hyperparameter tuning results: 5506.2722
- R2 score for the evaluation results: 0.9067
- RMSE for the evaluation results: 3945.0074

Overall, the final model built has achieved good performance on both the training and test data, with an R2 score of 0.8445 or higher and an RMSE of 4732.9684 or lower for the training data. The hyperparameter tuning results also showed good performance, with an R2 score of 0.7918 and an RMSE of 5506.2722. The evaluation results also indicated good performance, with an R2 score of 0.9067 and an RMSE of 3945.0074.

Ensemble Model Comaprison



Based on the provided summary, here is a comparison of the Gradient Boosting Regressor with other models:

1. Gradient Boosting Regressor (GBR):
   - R2: 0.9067
   - RMSE: 3945.0074

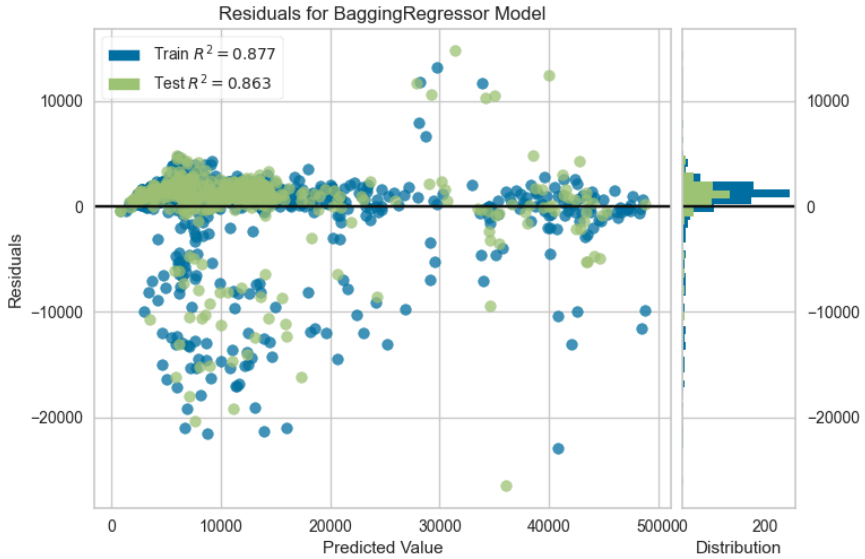
- Summary: The GBR model has a high R2 value of 0.9067, indicating strong linearity between the predictor variables and the target variable. However, its RMSE of 3945.0074 suggests that it is not overly accurate in predicting the target variable.

2. Gradient Boosting Regressor (Bagging):
   - R2: 0.903
   - RMSE: 4022.1087
   - Summary: The Bagging GBR model has a similar R2 value of 0.903 as the original GBR model, indicating that it too has strong linearity between the predictor variables and the target variable. However, its RMSE is higher than the original GBR model's, at 4022.1087.
3. Gradient Boosting Regressor (Boosting):
   - R2: 0.903
   - RMSE: 4022.1087
   - Summary: The Boosting GBR model has an R2 value of 0.903, indicating strong linearity between the predictor variables and the target variable, similar to the Bagging model. However, its RMSE is also high, at 4022.1087.
4. Voting Regressor:
   - R2: 0.8933
   - RMSE: 4217.5356
   - Summary: The Voting Regressor model has a lower R2 value of 0.8933, indicating weaker linearity between the predictor variables and the target variable compared to the GBR models. Its RMSE is also higher than the GBR models, at 4217.5356.
5. Stacking Regressor:
   - R2: 0.9029
   - RMSE: 4023.606
   - Summary: The Stacking Regressor model has a high R2 value of 0.9029, indicating strong linearity between the predictor variables and the target variable. However, its RMSE is slightly lower than the GBR models, at 4023.606.

Based on these comparisons, it appears that the Gradient Boosting Regressor (GBR) models have the highest accuracy in predicting the target variable, based on both R2 and RMSE values. However, it's important to note that these models are not mutually exclusive, and a hybrid model that combines elements of each could potentially perform even better.

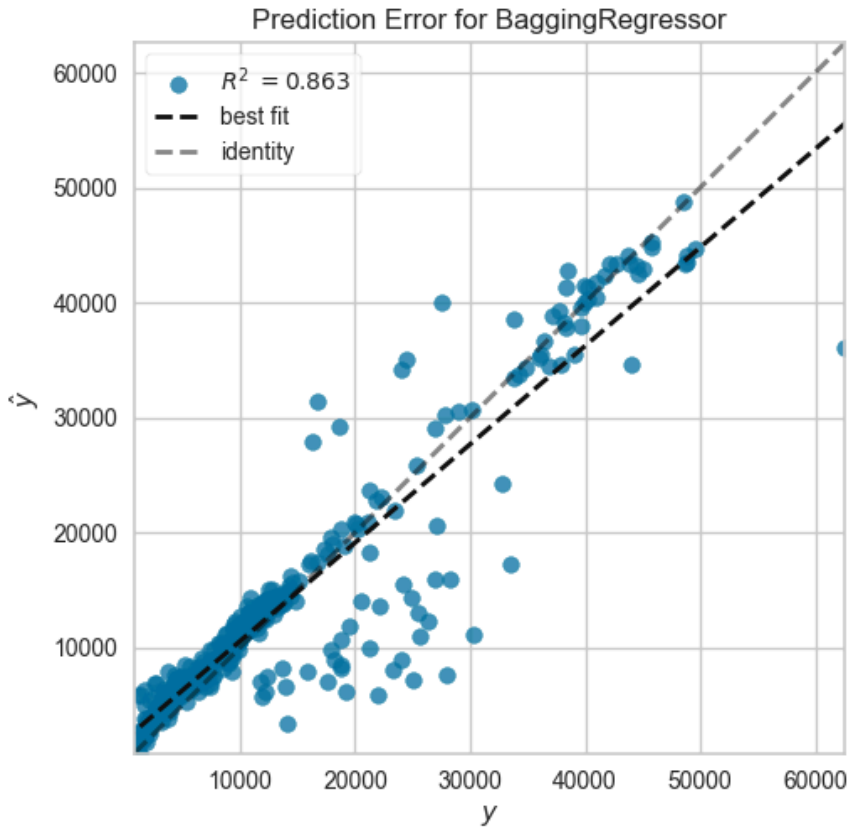
Model Analysis

Residuals Plot



A residual plot in machine learning is a graphical representation of the difference between the predicted values and the actual values for a given dataset. It helps in identifying if the model is overfitting or underfitting the data. To read a residual plot, look for a clear pattern of residuals around zero. If the residuals are randomly distributed around zero, it indicates that the model is a good fit to the data. However, if the residuals are consistently positive or negative, it suggests overfitting or underfitting, respectively. Additionally, check the scale of the x-axis and y-axis to ensure they are appropriate for the dataset.

Error Plot



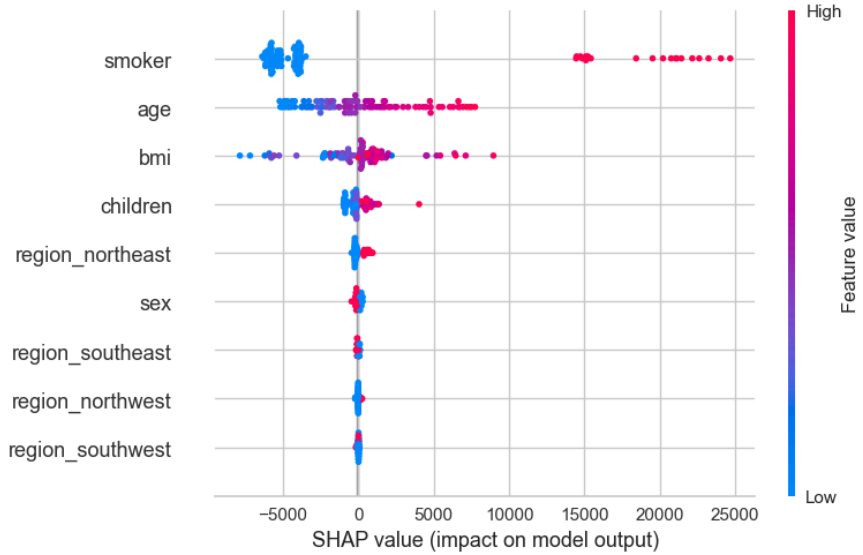
To read an error plot in Machine Learning, you need to understand the following:

1. X-axis: It represents the value of the feature or attribute that is being used to separate the data into different classes.
2. Y-axis: It represents the error rate or misclassification rate, which is the percentage of times when the model misclassifies the data point.
3. Plot shape: The shape of the error plot can help you identify the nature of the problem. For example, a flat error plot indicates that the model is overfitting, while a steep error plot suggests that the model is underfitting.
4. Error range: The range of errors in the plot can give you an idea of how consistent the model's performance is. A narrower range of errors indicates better consistency.
5. Labels: The labels on the X-axis and Y-axis help you identify the classes being modeled and the error rate, respectively.

By analyzing these aspects of the error plot, you can get a better understanding of your model's performance and identify areas for improvement.

Explainable AI

Global Explainability - Summary Plot

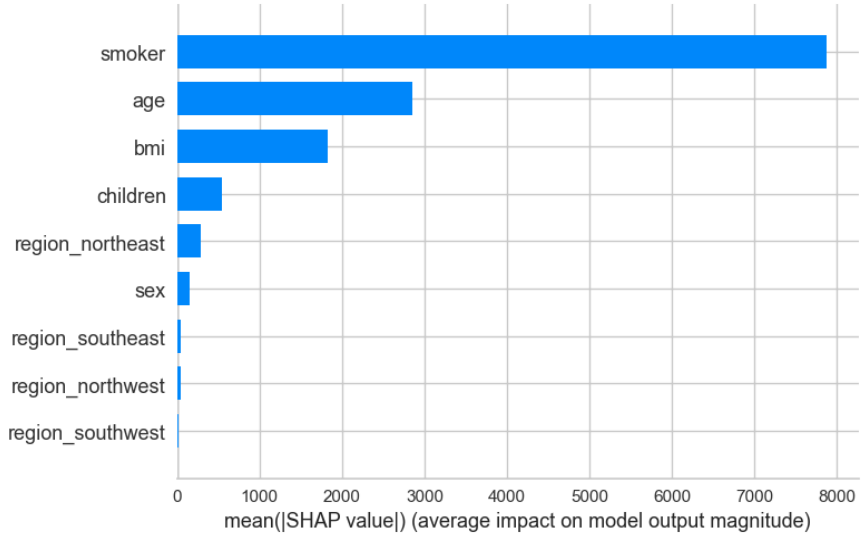


To read a SHAP Summary Plot, follow these steps:

1. Features are sorted by importance from top to bottom: The most important features are located at the top of the plot.
2. SHAP Values indicate impact: The X-axis represents SHAP values (impact on model output). Negative SHAP values (left) decrease the prediction, while positive SHAP values (right) increase the prediction.
3. Colors represent feature values: A color gradient (from blue to red) is used to represent feature values. Blue indicates lower feature values, while red indicates higher feature values.
4. Maximum 200 words: This plot provides a concise overview of the features' impact on the model output, allowing for quick and easy interpretation of the results.

In summary, the SHAP Summary Plot is a visualization tool that helps to explain the predictions made by a machine learning model by highlighting the most important features and their impact on the output. By analyzing this plot, you can gain insights into which features have the greatest influence on the model's predictions and how they relate to each other.

Global Explainability - Feature Importance



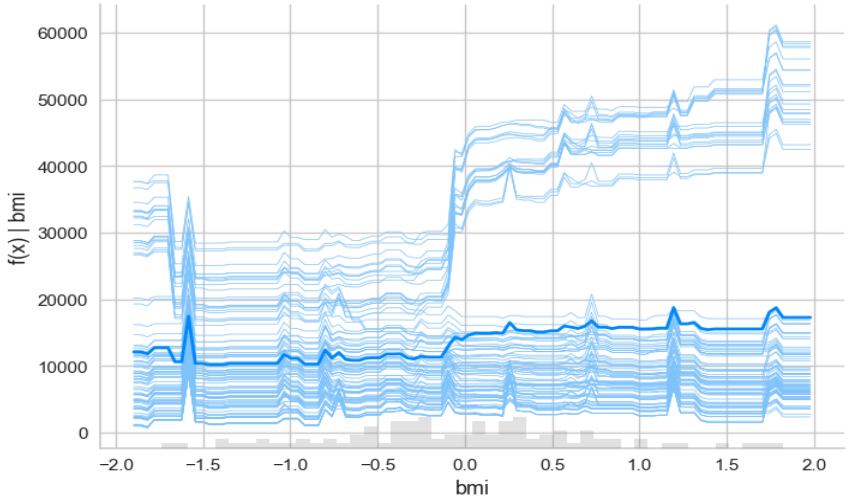
A Feature Importance Plot is a visualization tool that helps you understand the importance of different features in a machine learning model. To read a Feature Importance Plot, follow these steps:

1. Identify the features: Look at the x-axis of the plot, which represents the features of the dataset. Identify the features that are listed on the x-axis.
2. Understand the scale: The y-axis of the plot typically shows the importance of each feature. The scale can vary depending on the method used to calculate the importance, but common scales include a percentage or a logarithmic scale.
3. Interpret the results: Look for features that are highlighted in the plot as being important. These are the features that the model is using most frequently to make predictions. Pay attention to the rank order of the features, as this can indicate which features are most influential.
4. Analyze the patterns: Observe any patterns in the importance of the features. For example, you may notice that certain features are consistently more important than others across different models or datasets.
5. Relate to the problem: Think about how the importance of each feature relates to the problem you're trying to solve. For instance, if a feature is highly important for one task but not for another, it may indicate that the feature is specific to that task.

By following these steps, you can effectively read and interpret a Feature Importance Plot and gain insights into the behavior of your machine learning model.

Global Explainability - Partial Dependence Plot

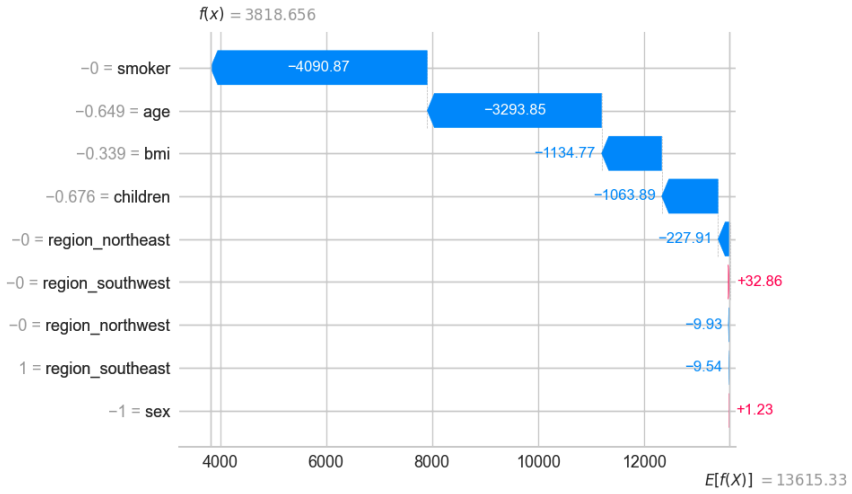
bmi



A partial dependence plot is a visualization tool used in global explainability to illustrate the relationship between a specific predictor and the predicted outcome variable. It displays the average prediction for different levels of the predictor, while holding all other predictors constant. This helps identify which predictors have the greatest influence on the outcome variable for a given level of the partial predictor. By analyzing these plots, you can gain insights into how the machine learning model is making predictions and where to focus further investigation for maximum explainability.

Local Explainability - SHAP Waterfall Plot

Instance 3



Interpreting a SHAP waterfall plot is straightforward and can provide valuable insights into how a machine learning model makes predictions. Here's a step-by-step guide to help you understand the plot:

1. Start from the base value (leftmost value at the bottom): This represents the model's expected prediction for the given input.
2. Observe how each feature modifies the base value: The waterfall plot shows the incremental effect of each feature on the predicted value. Features that push the prediction higher are in red, while those pulling it lower are in blue.
3. Identify the most impactful features: The largest bars (longest red/blue bars) indicate the strongest contributors to the prediction. These features have the greatest influence on the model's output.
4. Determine if the prediction is higher or lower than expected: If most contributions are positive (red), the prediction is likely above average. Conversely, mostly negative (blue) contributions suggest a below-average prediction.
5. Analyze the impact of categorical and numerical features: Binary features often have a sharp impact on the prediction, while continuous features gradually modify the prediction.
6. Evaluate the feature importance order: The most impactful features are placed at the top of the plot, while less significant ones are towards the bottom. This can help you identify which features are driving the model's predictions.
7. Interpret the final prediction (f(x)): The predicted value is obtained by summing the base value and the feature effects.

By following these steps, you can effectively interpret a SHAP waterfall plot and gain insights into how your machine learning model makes predictions.