Explainable Machine Learning

Agenda

Introductions

Motivation

DALEX

LIME

Further Concepts

Mango Solutions

- Data science services
- Cross sector
- ~ 70 people
- London and Chippenham
- We 🧡 R, Python and Spark

Motivation

Business and Government Decisions Increasingly Rely on Machine Learning and AI

- Healthcare
- Banking
- Crime
- Education
- Recruitment
- …

We Choose to Trust ML Algorithms based on their Accuracy

=

ACCURACY

TRUST

More Accurate ML Algorithms are also Less Interpretable

INTERPRETIBILITY

ACCURACY

- Overfitting
- Correlation
- Noise
- Truth

Interpretable Models are Key in High-Stake Decisions…

- **Healthcare:** cancer detection
- **Banking**: loan lending
- **Crime:** detention and bail
- **Education:** teachers' promotion / redundancy
- **Recruitment:** interviews

Check out Weapons of Math Destruction by Cathy O'Neil

.. And in Low-Risk Decisions, too!

TRUST

- Sanity check
- Generalizability
- Fairness

PREDICT

- Foresight of model behaviour

IMPROVE

- Feature and model improvement

Explainable Machine Learning and AI (XML/XAI)

Techniques in Artificial Intelligence [and Machine Learning] that (…) make model predictions easily understood by humans. It contrasts with the concept of the black box in machine learning where even their designers cannot explain why the AI arrived at a specific decision. [1]

[1] https://en.m.wikipedia.org/wiki/Explainable_artificial_intelligence

DALEX

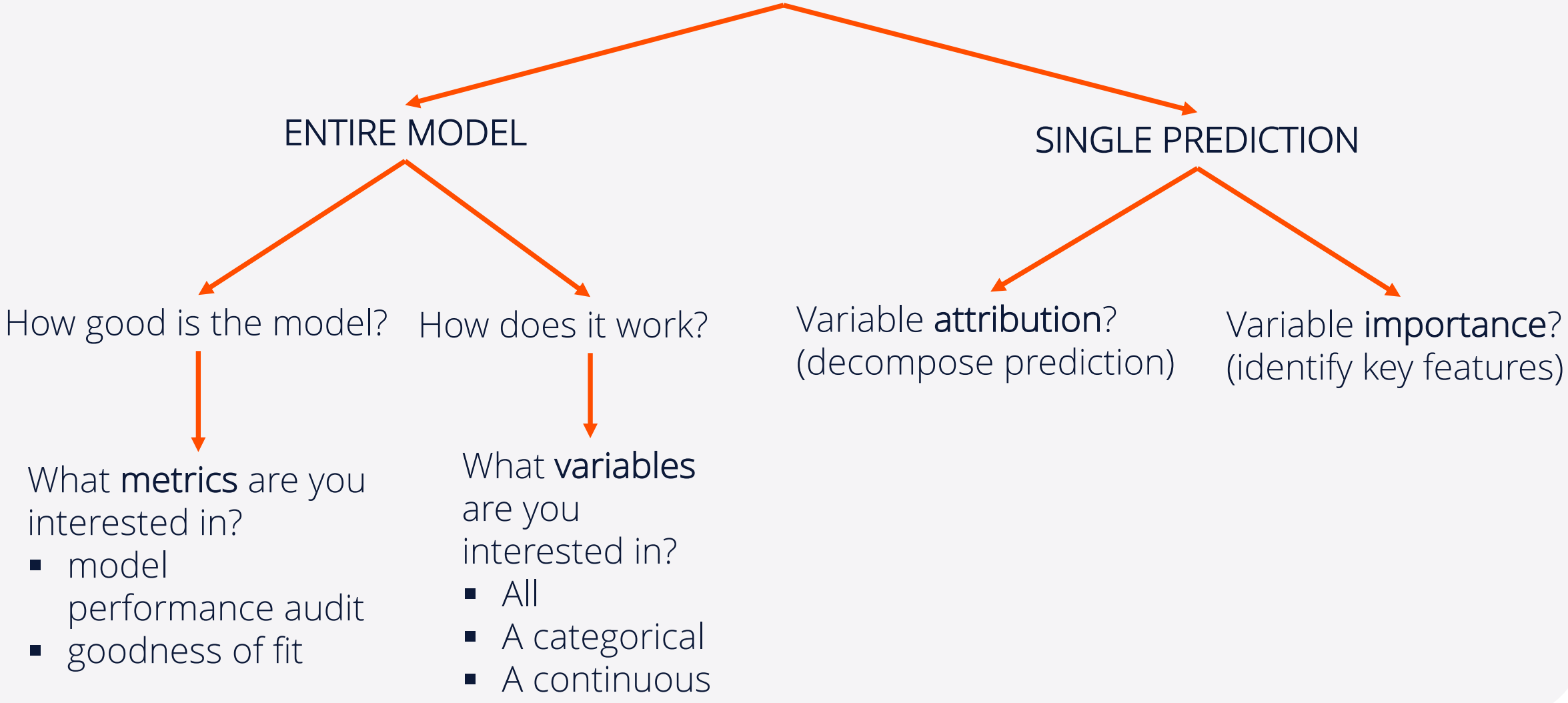
DALEX:
Descriptive mAchine Learning EXplanations

DALEX is a set of tools that help understand how complex models are working

Developed by Przemyslaw Biecek

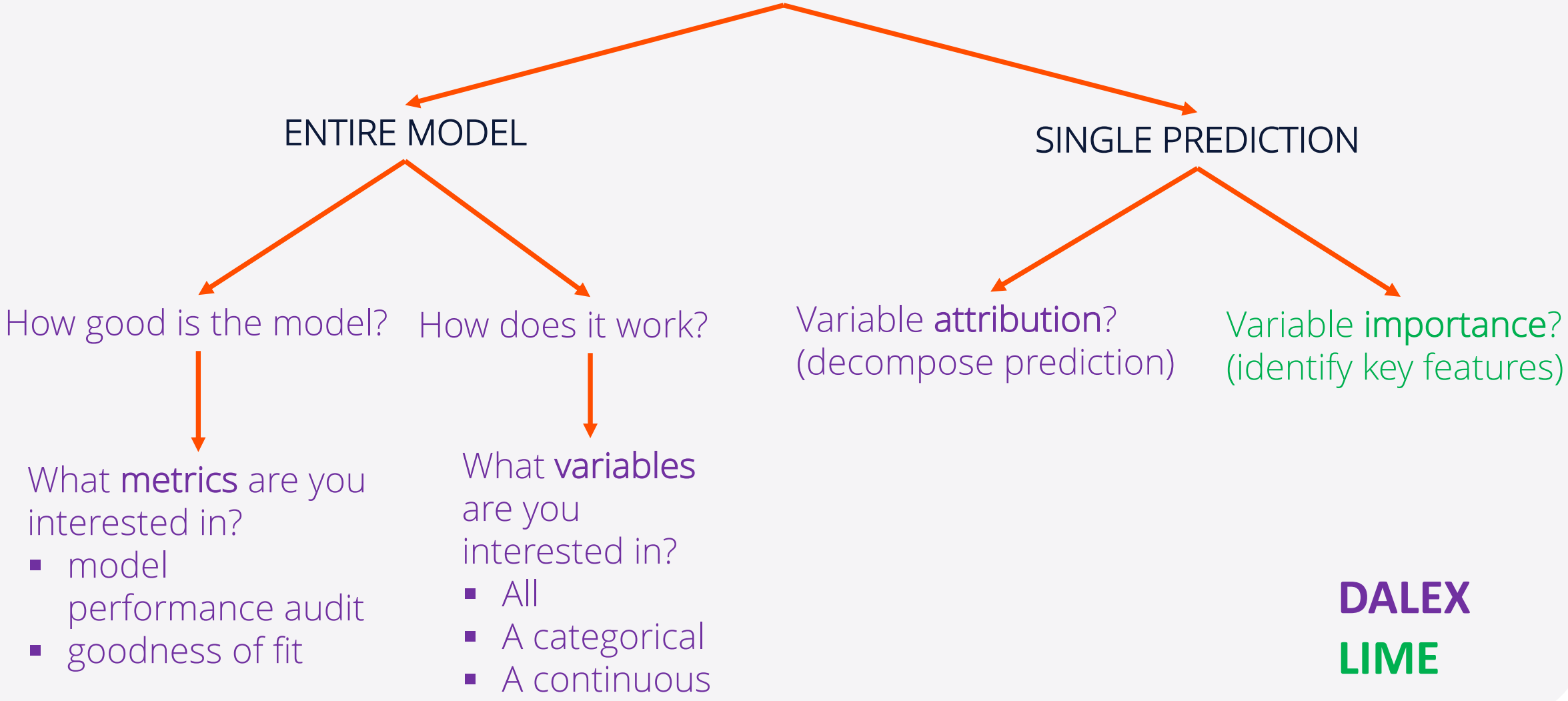
Github: https://github.com/pbiecek/DALEX

WHAT DO YOU WANT TO UNDERSTAND?



Based on https://raw.githubusercontent.com/pbiecek/DALEX_docs/master/images/Explain.png

WHAT DO YOU WANT TO UNDERSTAND?



Based on https://raw.githubusercontent.com/pbiecek/DALEX_docs/master/images/Explain.png

Before You Start

Regression problem – predict apartment prices in Warsaw, Poland

```
library(DALEX)
library(randomForest)

# train random forest and linear model
str(apartments)
set.seed(519)
apartments_rf_model <- randomForest::randomForest(m2.price ~ ., data = apartments)
predicted_rf <- predict(apartments_rf_model, apartmentsTest)

apartments_lm_model <-  lm(m2.price ~ ., data = apartments)
predicted_lm <- predict(apartments_lm_model, apartmentsTest)
```

Start with the Explainer

Compare model performance

```
# root mean square

sqrt(mean((predicted_rf -apartmentsTest$m2.price)^2))

sqrt(mean((predicted_lm - apartmentsTest$m2.price)^2))
```

Run DALEX explainer

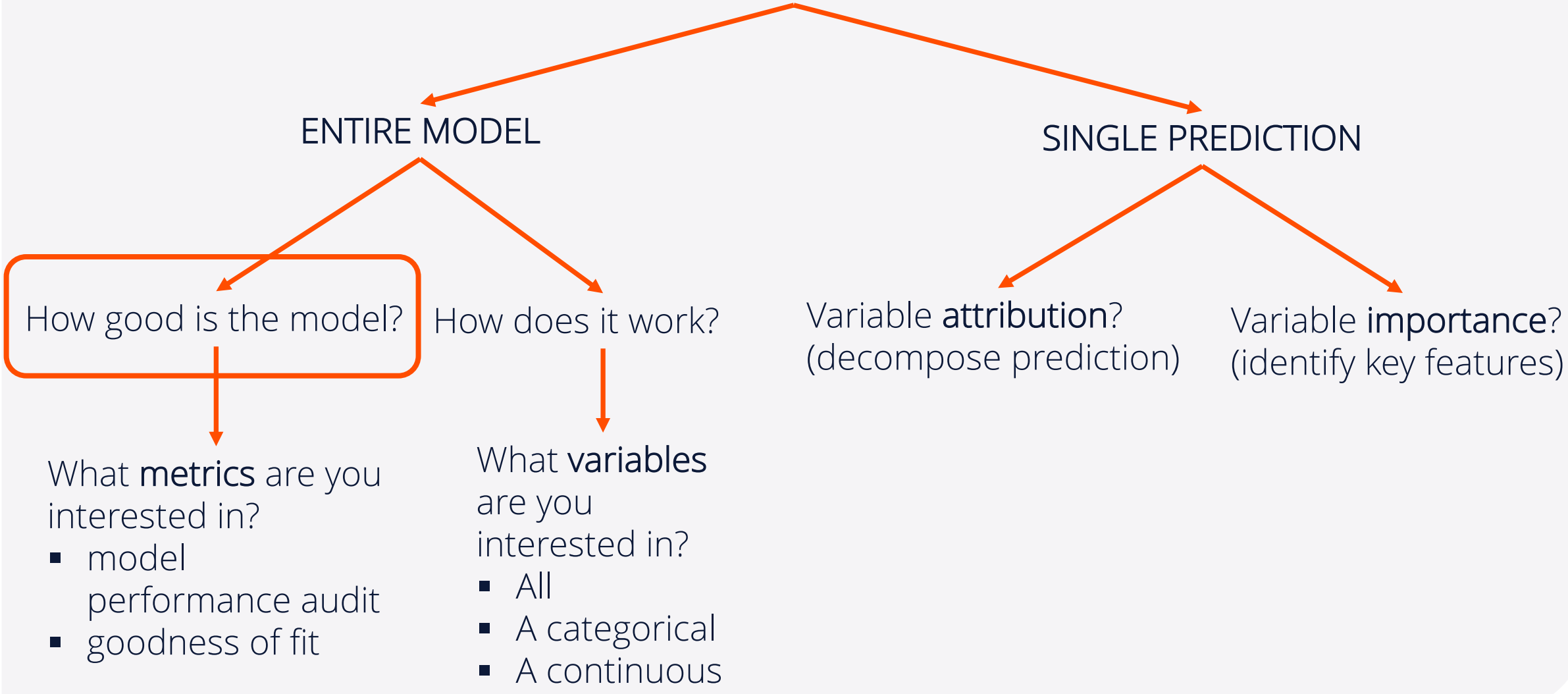
```
explainer_lm <- DALEX::explain(model = apartments_lm_model,
                               data = apartmentsTest[,2:6], y = apartmentsTest$m2.price)
explainer_rf <- DALEX::explain(model = apartments_rf_model,
                               data = apartmentsTest[,2:6], y = apartmentsTest$m2.price)
```

Start with the Explainer

DALEX explainer attaches relevant meta data to the algorithms and unifies model interfacing

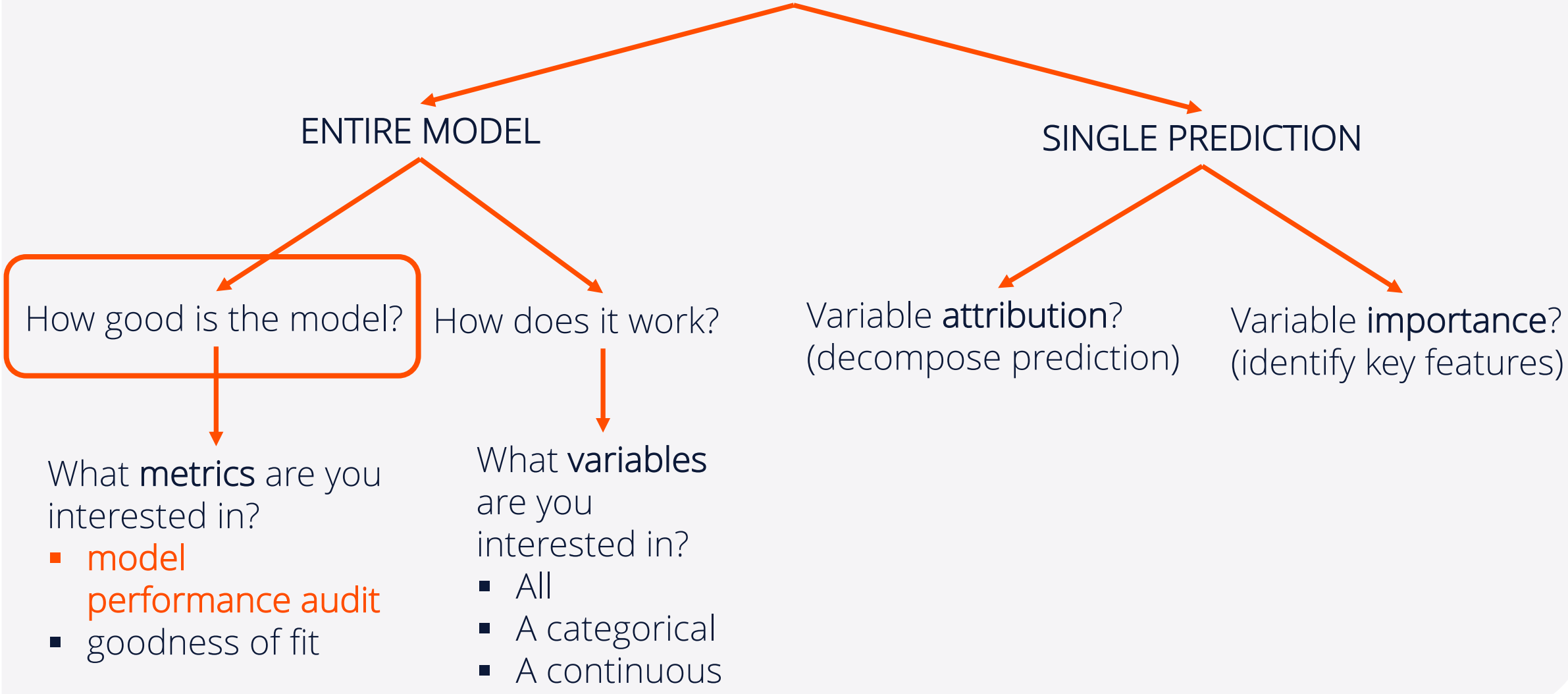
```
> explainer_lm

Model label:  train

Model class:  train

Data head  :

     construction.year surface floor no.rooms    district

1001              1976     131     3         5 Srodmiescie

1002              1978     112     9         4     Mokotow
```

WHAT DO YOU WANT TO UNDERSTAND?



Based on https://raw.githubusercontent.com/pbiecek/DALEX_docs/master/images/Explain.png

WHAT DO YOU WANT TO UNDERSTAND?



Based on https://raw.githubusercontent.com/pbiecek/DALEX_docs/master/images/Explain.png

How Good is the Model?

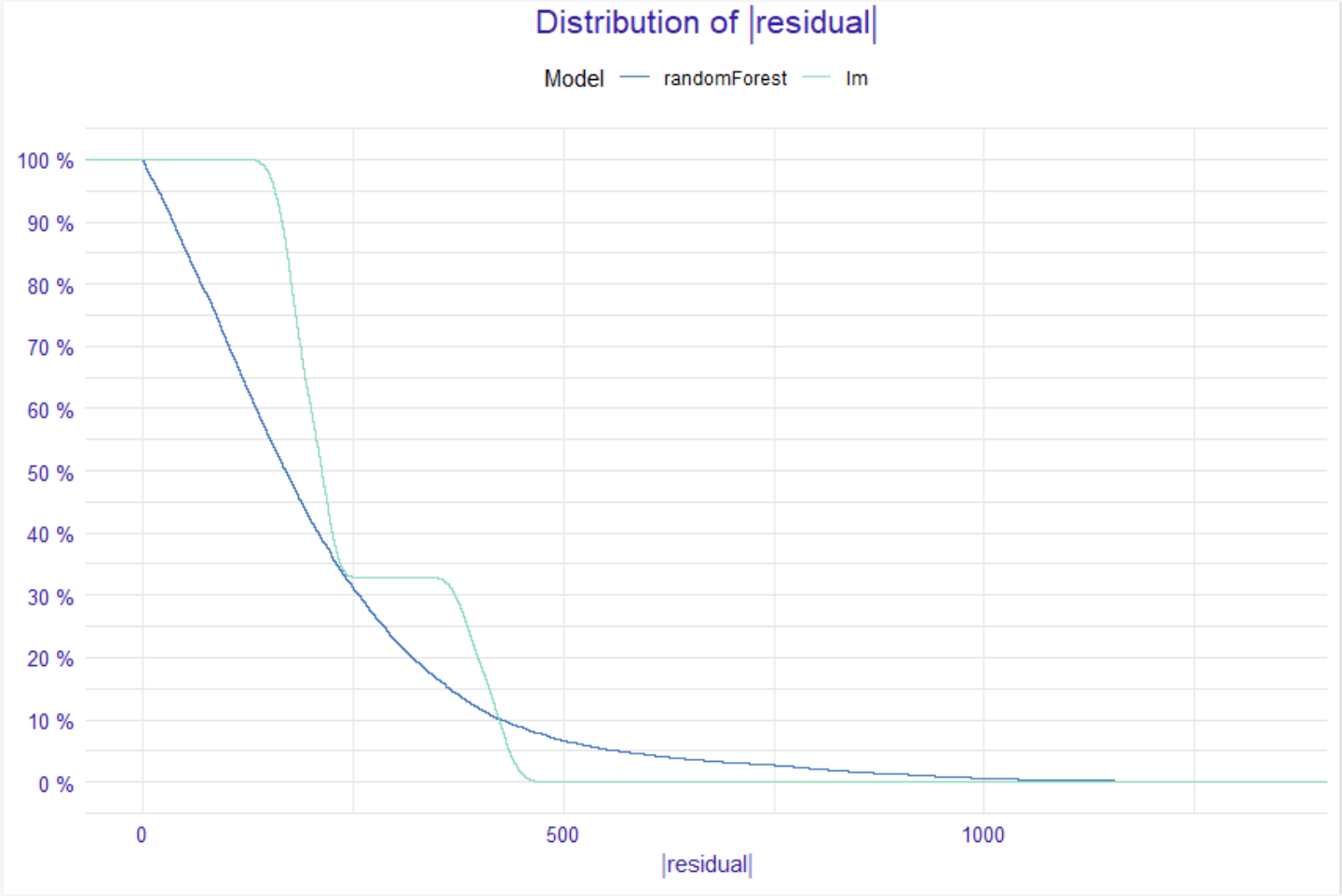
Explainer for model performance gives more information in a consistent form

The function `model_performance()` calculates predictions and residuals for validation data `apartments_test`

```
mp_lm <- model_performance(explainer_lm)

mp_rf <- model_performance(explainer_rf)
```

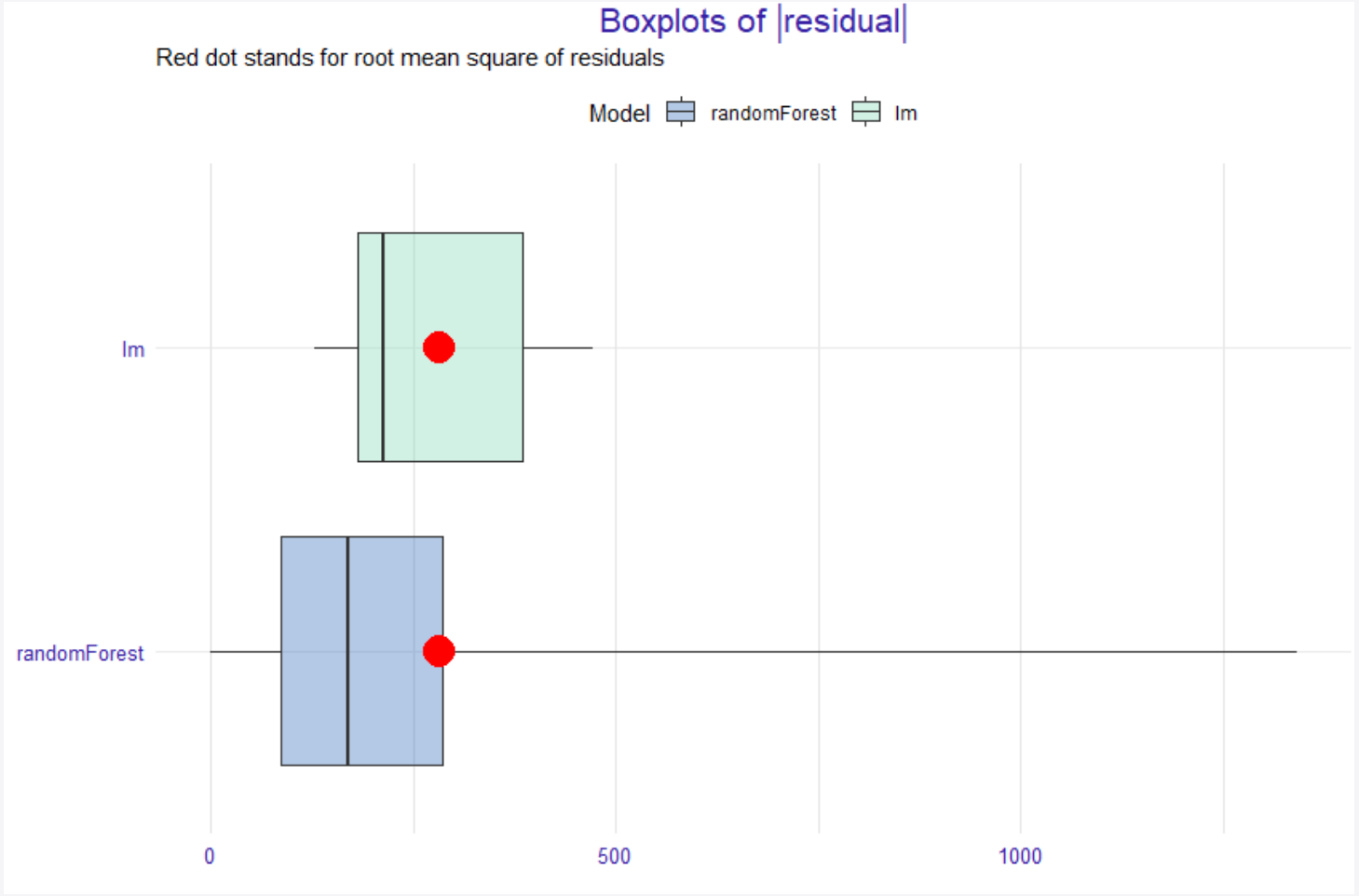
How Good is the Model?

```
# plot model performance

plot(mp_lm, mp_rf)
```

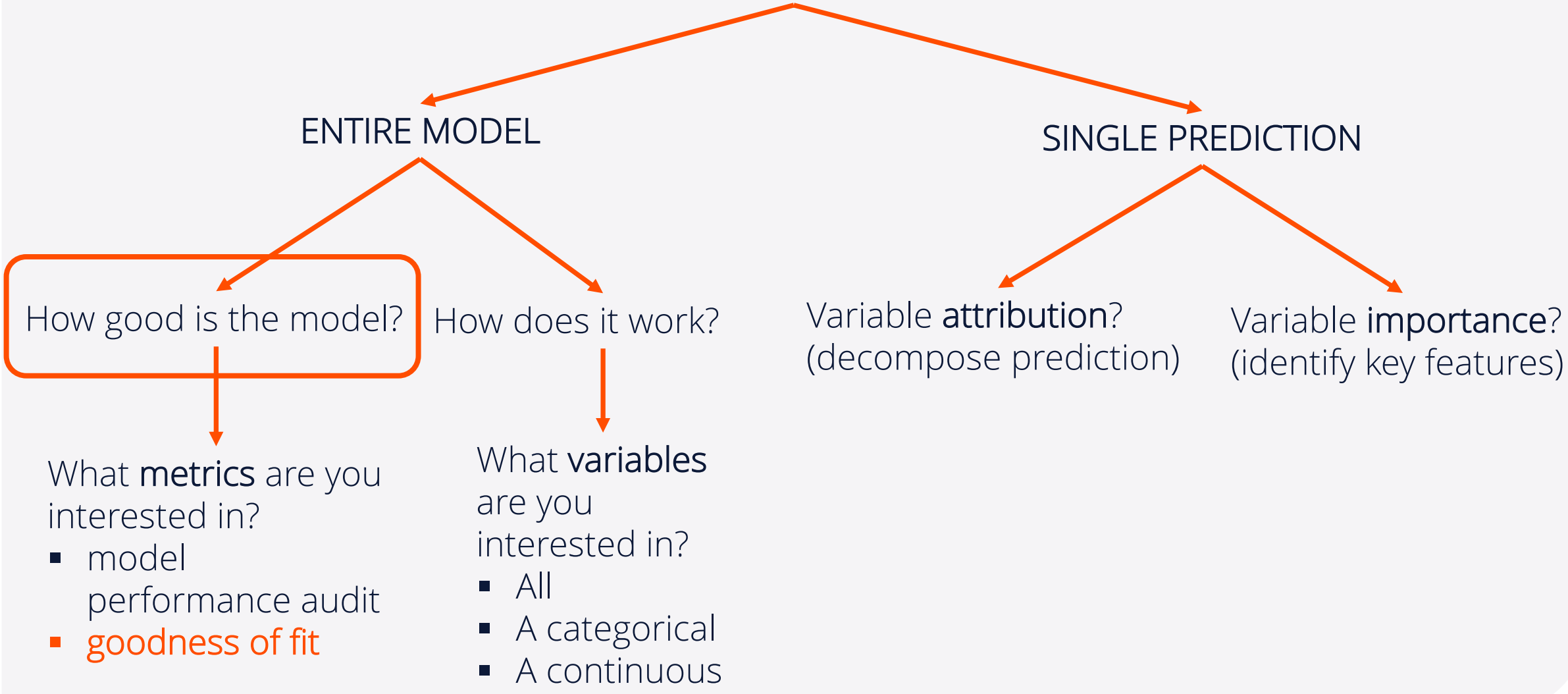


How Good is the Model?

```
# plot model performance

plot(mp_lm, mp_rf,

     geom = "boxplot")
```



WHAT DO YOU WANT TO UNDERSTAND?



Based on https://raw.githubusercontent.com/pbiecek/DALEX_docs/master/images/Explain.png

How Good is the Model?

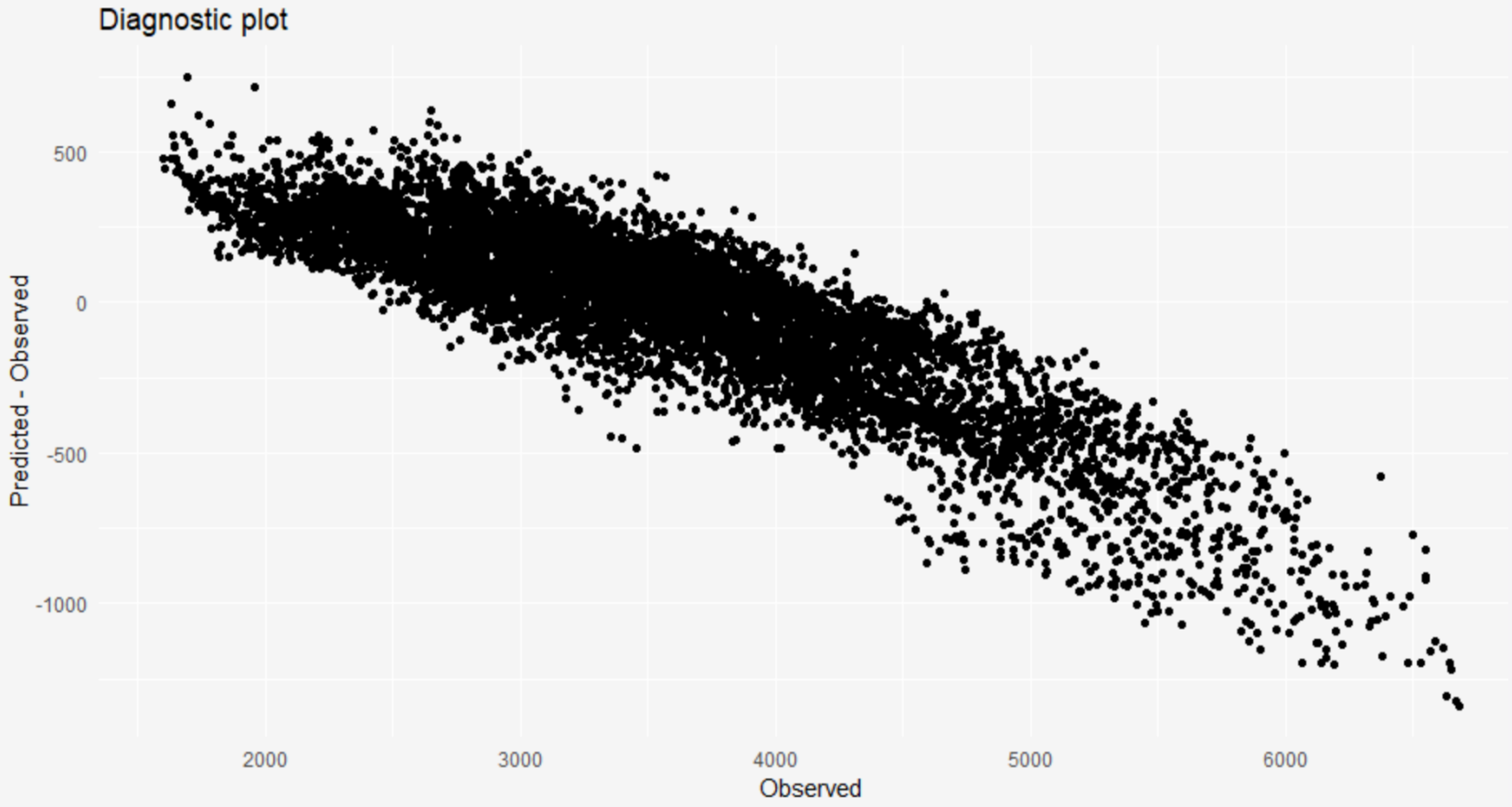
Explainer for model performance gives more information in a consistent form

The function `model_performance()` calculates predictions and residuals for validation data `apartments_test`

```
mp_rf <- model_performance(explainer_rf)

mp_rf$observed

mp_rf$predicted

mp_rf$diff # predicted - observed
```

How Good is the Model?

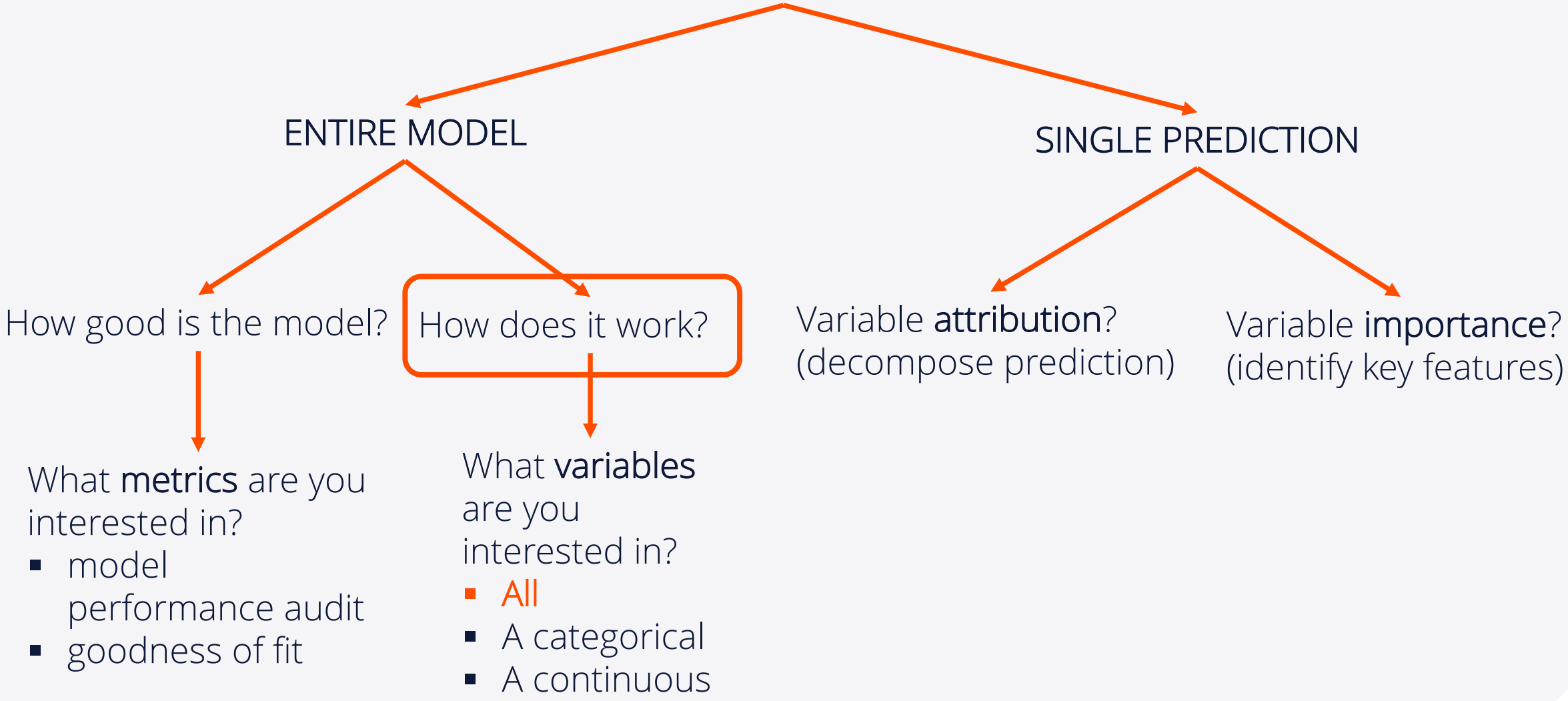
```
ggplot(mp_rf,
       aes(observed, diff)) +
  geom_point() +
  xlab("Observed") +
  ylab("Predicted - Observed") +
  ggtitle("Diagnostic plot") +
  theme_mi2()
```



Exercise

- Load DALEX package and explore its in-built `dragons` and `dragons_test` dataset. The feature to predict is `life_length`.
- Create 2 explainers – one for the linear model and one for random forest .
- Compare models' performance in terms of distribution of residuals. Which one performs better?

WHAT DO YOU WANT TO UNDERSTAND?



Based on https://raw.githubusercontent.com/pbiecek/DALEX_docs/master/images/Explain.png

How Does the Model Work?

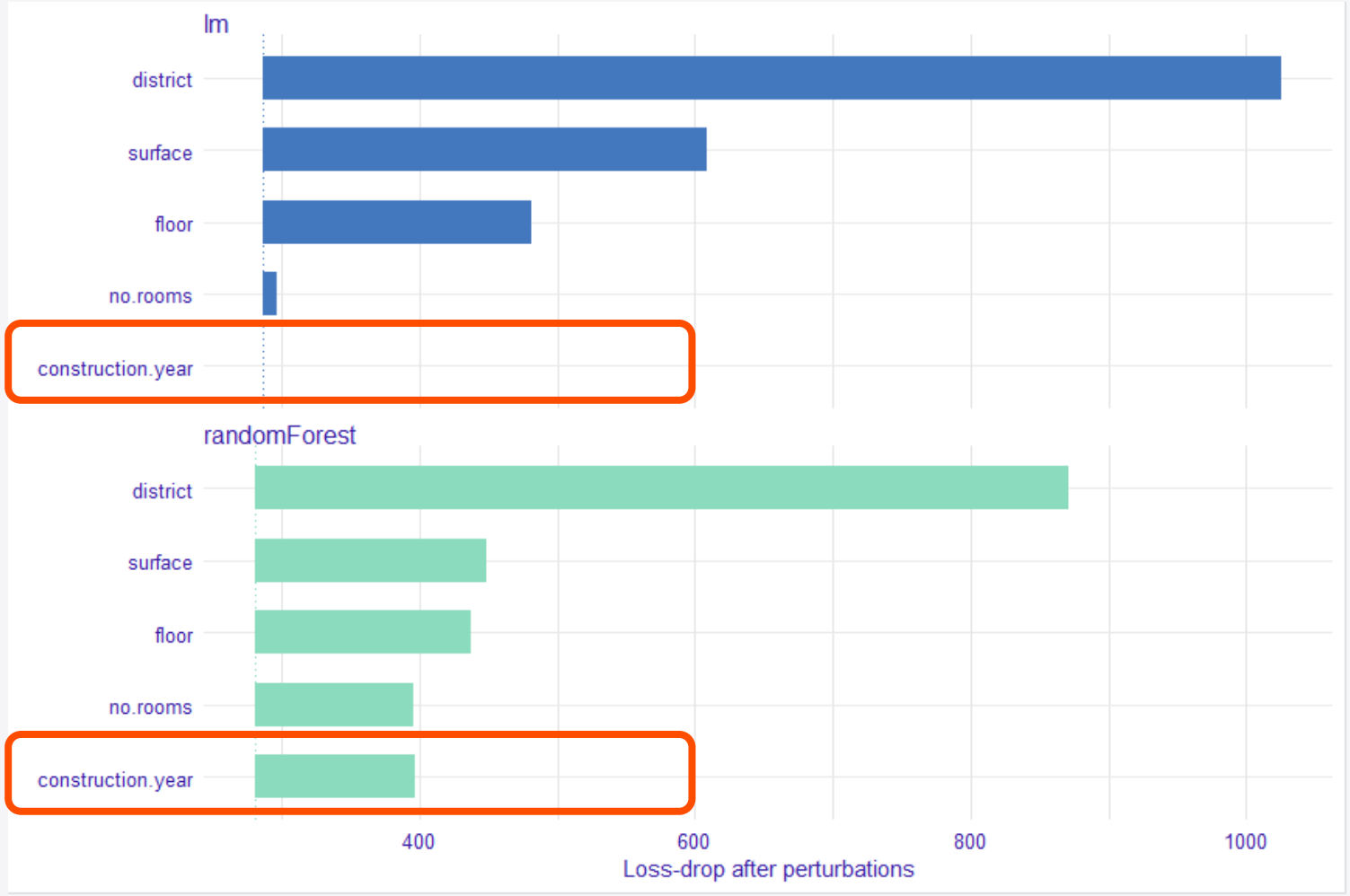
Variable Importance

- Variable importance helps us validate the model and increase our understanding of the domain.
- The function `variable_importance()` provides model agnostic variable importance (as opposed to model-specific).

```
vi_rf <- variable_importance(explainer_rf, loss_function = loss_root_mean_square)

vi_lm <- variable_importance(explainer_lm, loss_function = loss_root_mean_square)
```

How Does the Model Work?

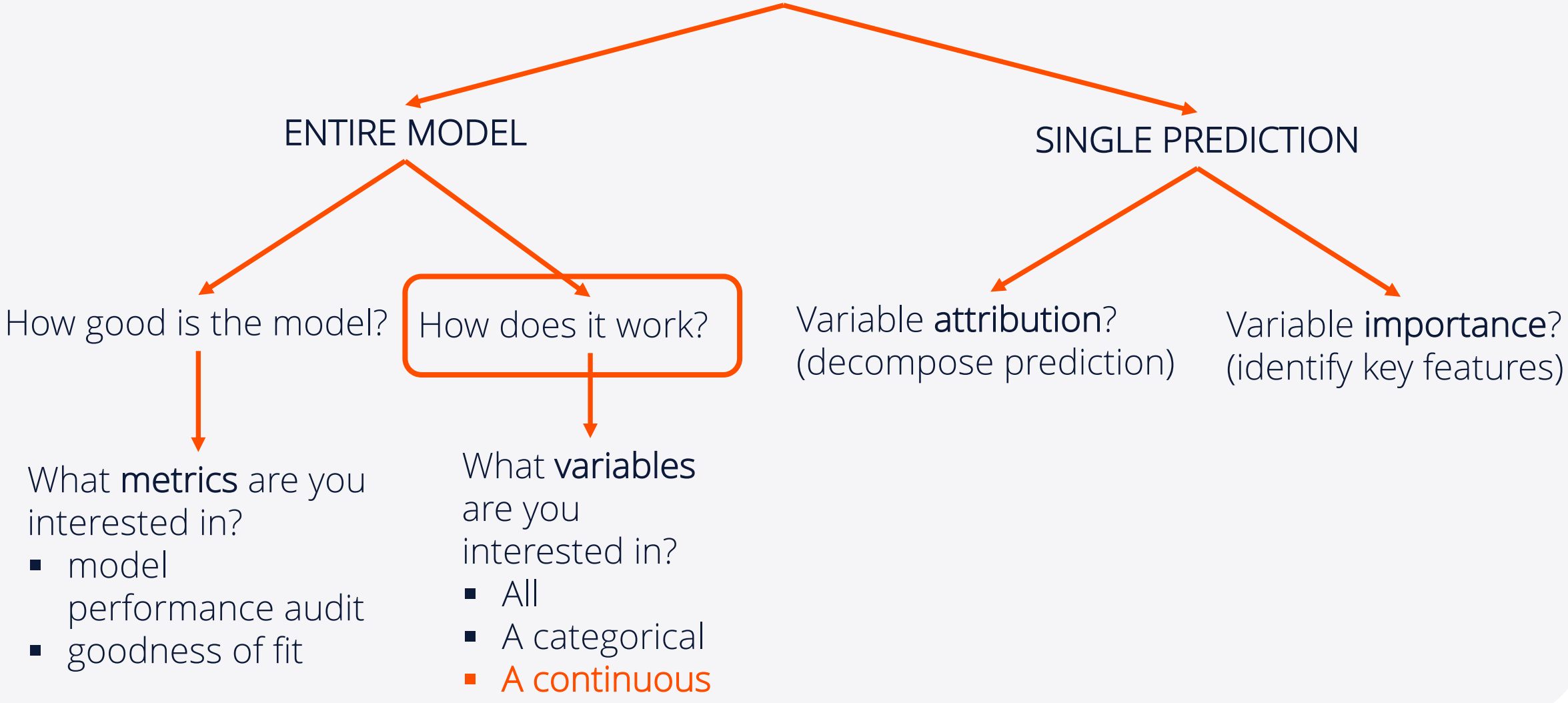
```
plot(vi_rf, vi_lm)
```



Exercise

- Load DALEX package and explore its in-built `dragons` and `dragons_test` dataset. The feature to predict is `life_length`.
- Create 2 explainers – one for the linear model and one for random forest .
- Compare models' performance in terms of distribution of residuals. Which one performs better?
- Extract and compare models' variable importance. How are they similar? How are they different?

WHAT DO YOU WANT TO UNDERSTAND?



Based on https://raw.githubusercontent.com/pbiecek/DALEX_docs/master/images/Explain.png

How Does the Model Work?

A single continuous variable

Partial Dependence Plots (PDP) show the expected output conditional on the selected variable.

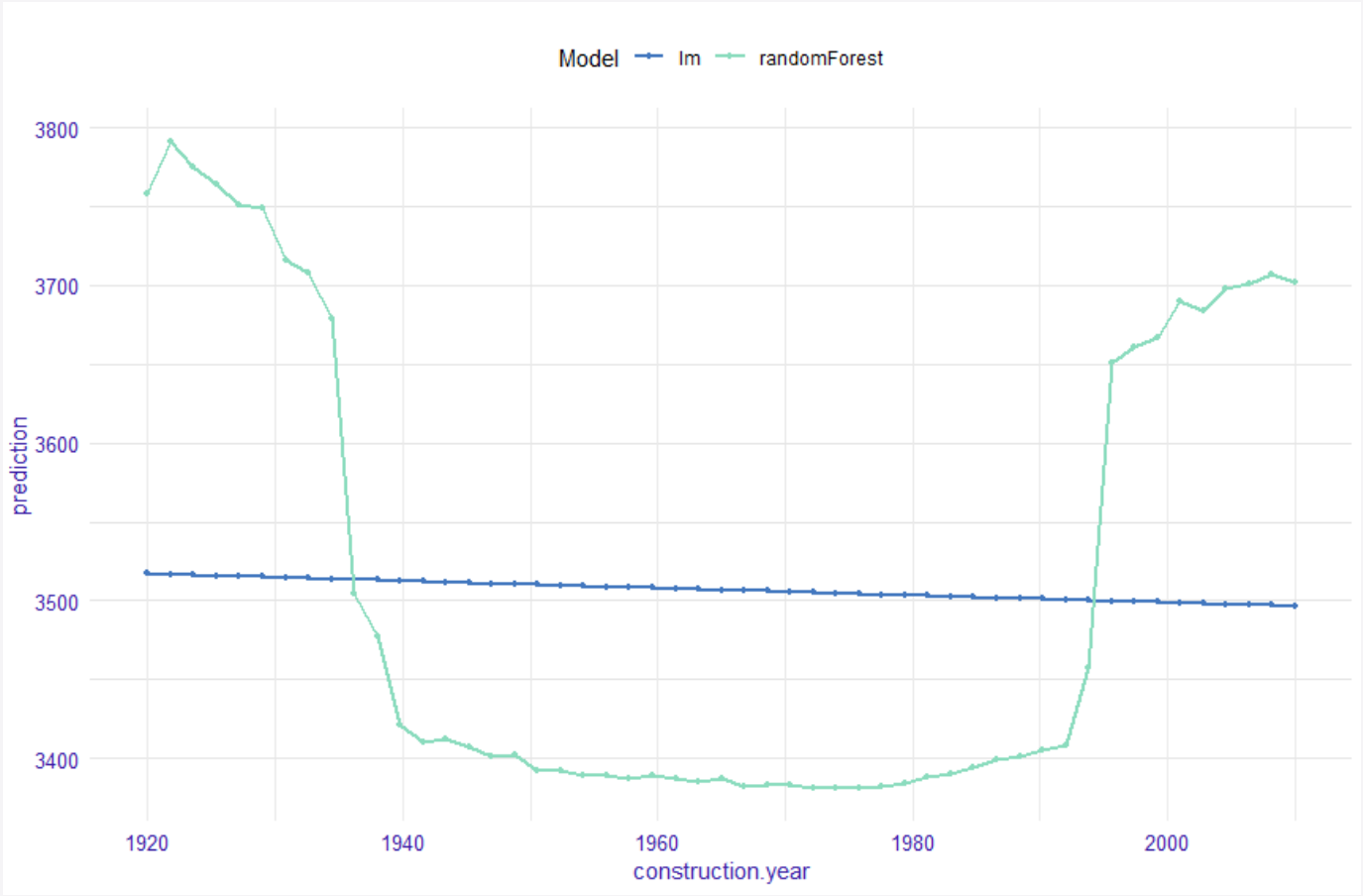
```
sv_rf  <- single_variable(explainer_rf, variable =  "construction.year", type = "pdp")

sv_lm  <- single_variable(explainer_lm, variable =  "construction.year", type = "pdp")
```

How Does the Model Work?

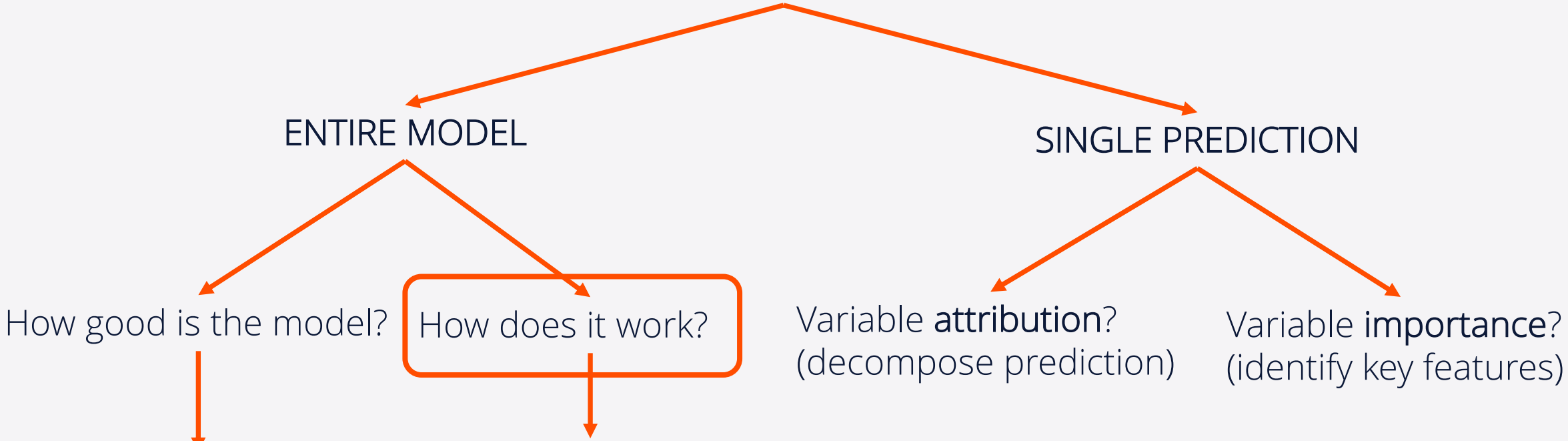
Partial Dependence Plots

```
plot(sv_rf, sv_lm)
```

The linear model is unable to capture a non-linear relationship between construction year and apartment price.



WHAT DO YOU WANT TO UNDERSTAND?



Based on https://raw.githubusercontent.com/pbiecek/DALEX_docs/master/images/Explain.png

How Does the Model Work?

A single categorical variable

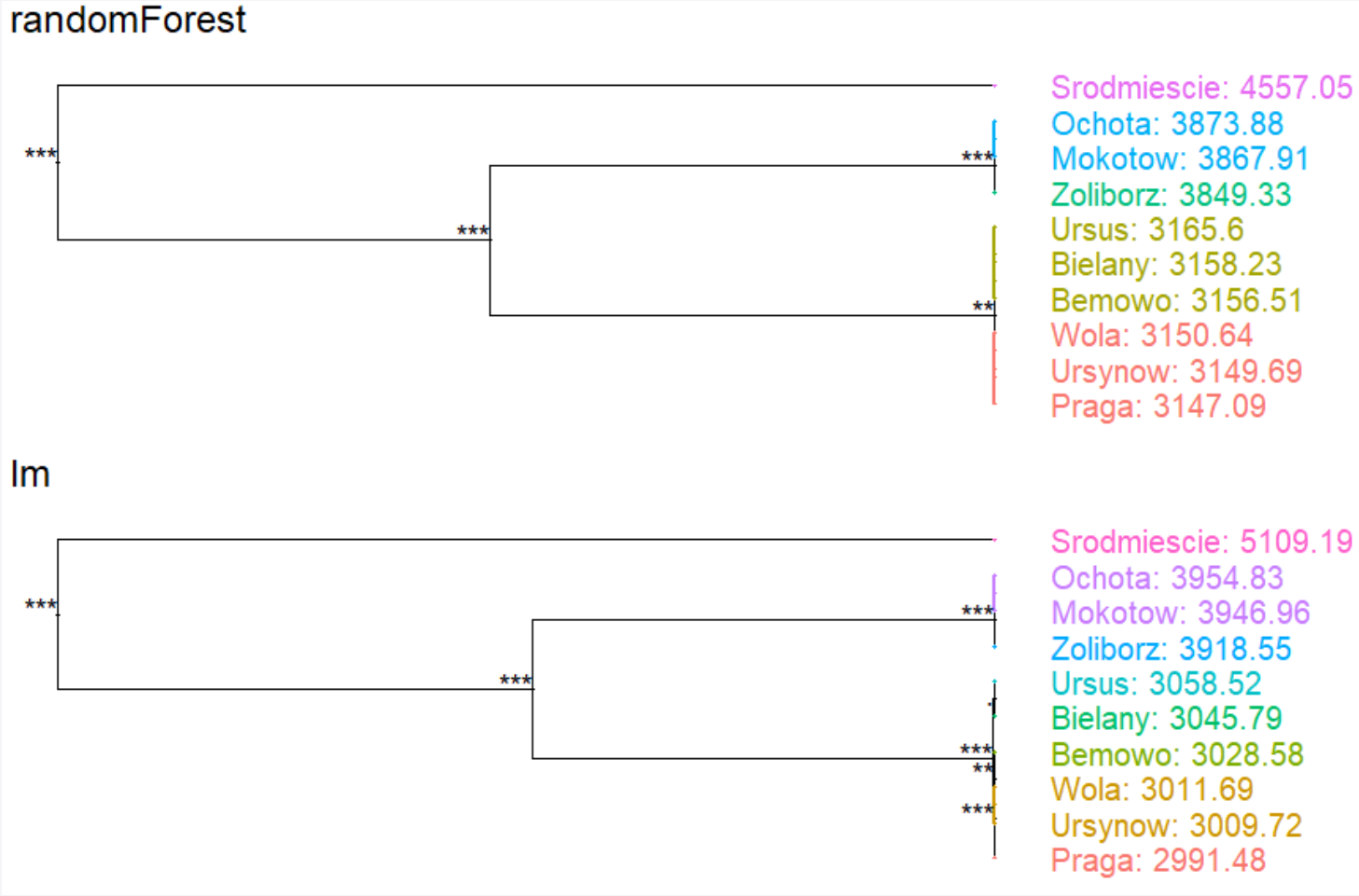
```
svd_rf  <- single_variable(explainer_rf, variable =  "district", type = "factor")

svd_lm  <- single_variable(explainer_lm, variable =  " district ", type = " factor")
```

How Does the Model Work?

Merging Path Plots

```
plot(svd_rf, svd_lm)
```

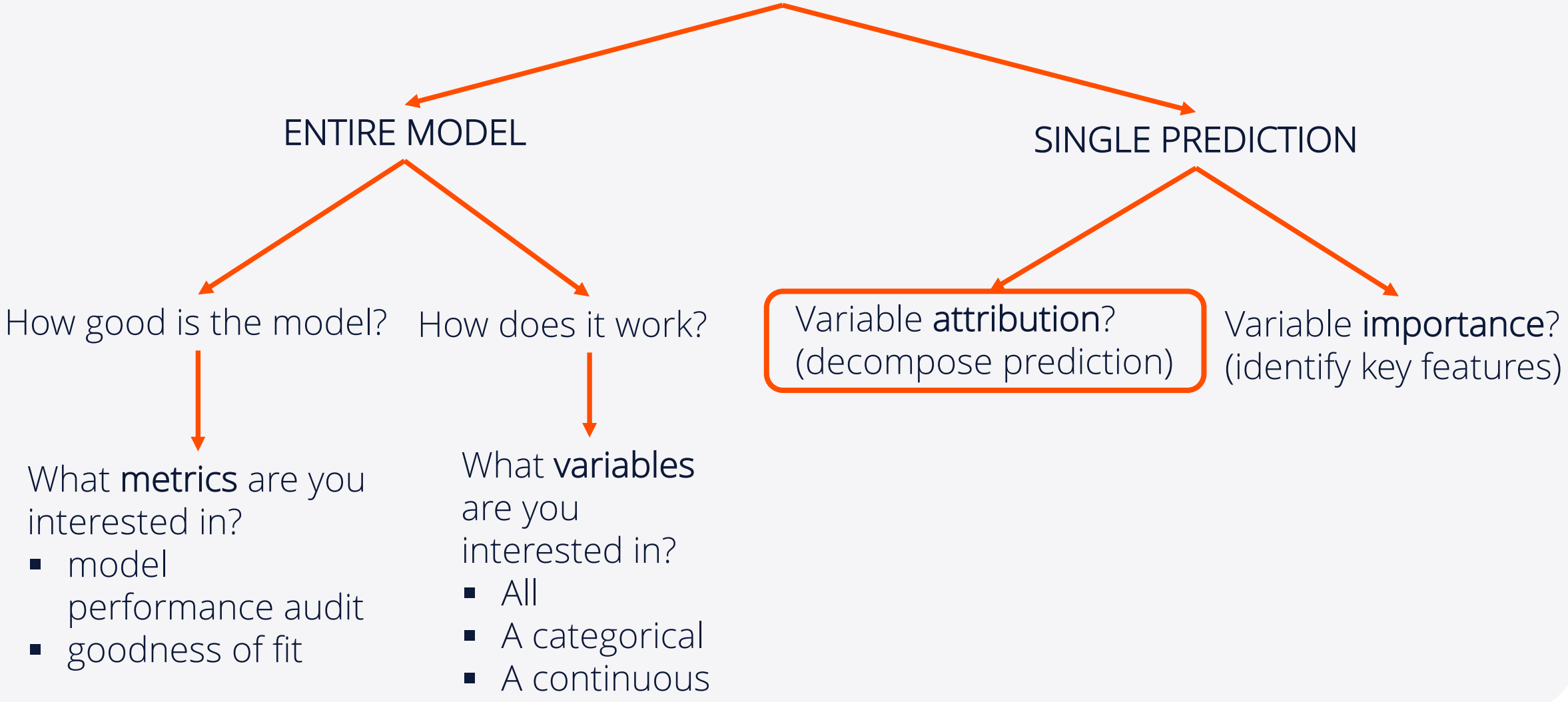
Can you identify the three distinct clusters?



Exercise

- Load DALEX package and explore its in-built `dragons` and `dragons_test` dataset. The feature to predict is `life_length`.
- Create 2 explainers – one for the linear model and one for random forest .
- Compare models' performance in terms of distribution of residuals. Which one performs better?
- Extract and compare models' variable importance. How are they similar? How are they different?
- Explore `height` in both models using Partial Dependency Plots. What are the differences and why?

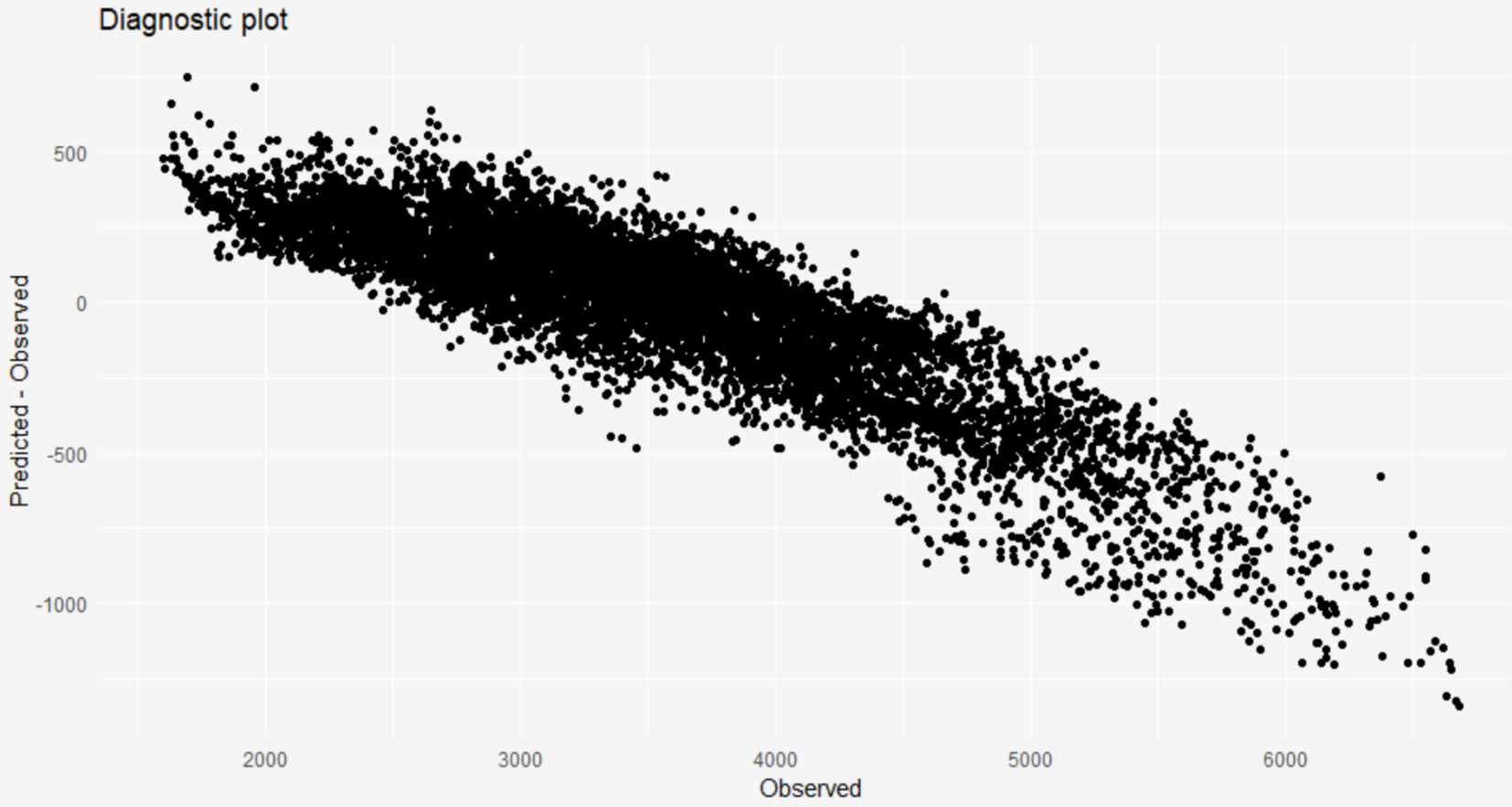
WHAT DO YOU WANT TO UNDERSTAND?



Based on https://raw.githubusercontent.com/pbiecek/DALEX_docs/master/images/Explain.png

Do You Remember `model_performance()`?

```
ggplot(mp_rf,
       aes(observed, diff)) +
  geom_point() +
  xlab("Observed") +
  ylab("Predicted - Observed") +
  ggtitle("Diagnostic plot") +
  theme_mi2()
```



Understand a Single Prediction

Variable attribution

- Identify the top outlier for RF model

```
max_diff_rf  <- which.max(abs(mp_rf$diff))
```

- See what the corresponding observed and predicted values are

```
mp_rf$predicted[max_diff_rf]

mp_rf$observed[max_diff_rf]
```

- Pick a single observation for which we want to gain more understanding

```
new_apartment <- apartments_test[max_diff_rf, ]

new_apartment
```

Understand a Single Prediction

Variable attribution

Breakdown plots and `single_prediction()` show the contribution of every variable present in the model

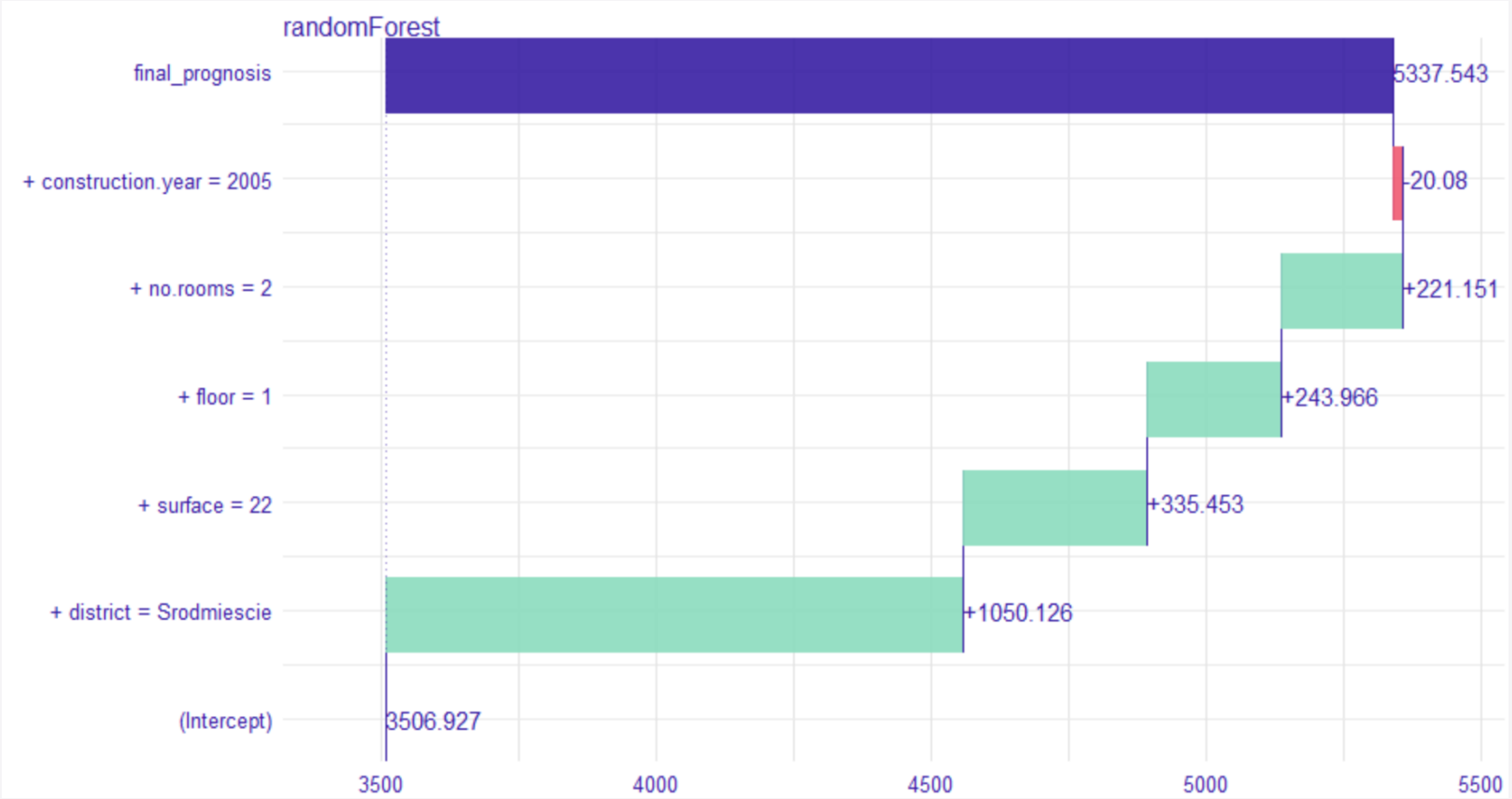
```
# Random Forest Model
new_apartment_rf <- single_prediction(explainer_rf, observation = new_apartment)
new_apartment_rf # one view
breakDown:::print.broken(new_apartment_rf) # more concise view


# Linear Model
new_apartment_lm <- single_prediction(explainer_lm, observation = new_apartment)
```

Understanding a Single Prediction

Variable attribution

```
plot(new_apartment_rf)
```

Breakdown plots show the contribution of every variable present in the model.



Understanding a Single Prediction

Variable attribution

```
plot(new_apartment_lm,
new_apartment_rf)
```

Breakdown plots show the contribution of every variable present in the model.

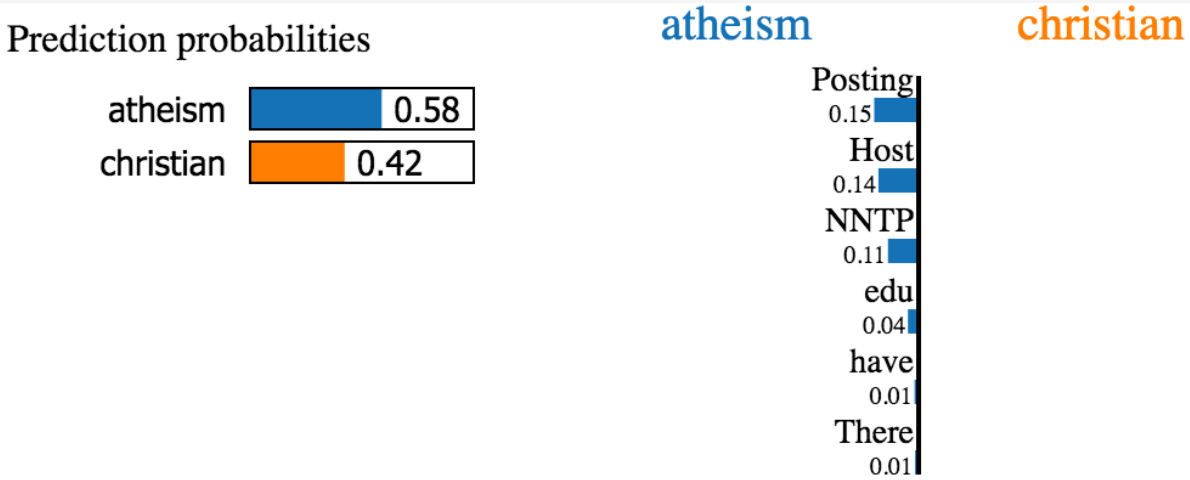
Exercise

- Load DALEX package and explore its in-built `dragons` and `dragons_test` dataset. The feature to predict is `life_length`.
- Create 2 explainers – one for the linear model and one for random forest .
- Compare models' performance in terms of distribution of residuals. Which one performs better?
- Extract and compare models' variable importance. How are they similar? How are they different?
- Explore `height` in both models using Partial Dependency Plots. What are the differences and why?
- Select the top outlier from the Random Forest model and understand the composition of its prediction in both linear and random forest models. Which algorithm performed better and why?

LIME

Example: Text Analysis

Predict whether an email is about atheism or Christianity



Text with highlighted words

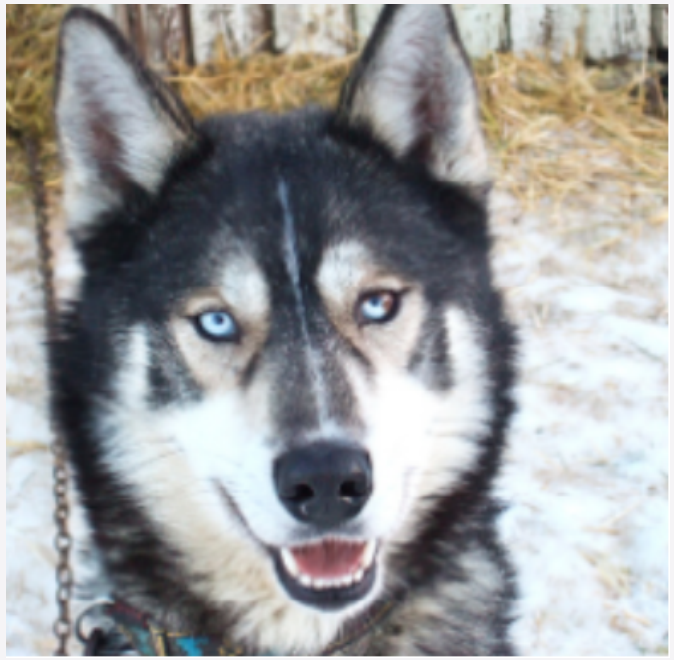
From: johnchad@triton.unm.edu (jchadwic)
Subject: Another request for Darwin Fish
Organization: University of New Mexico, Albuquerque
Lines: 11
NNTP-Posting-Host: triton.unm.edu

Hello Gang,

There have been some notes recently asking where to obtain the DARWIN fish.
This is the same question I have and I have not seen an answer on the
net. If anyone has a contact please post on the net or email me.

Example: Image Classification

Predict whether an image shows a wolf or a husky



Examples

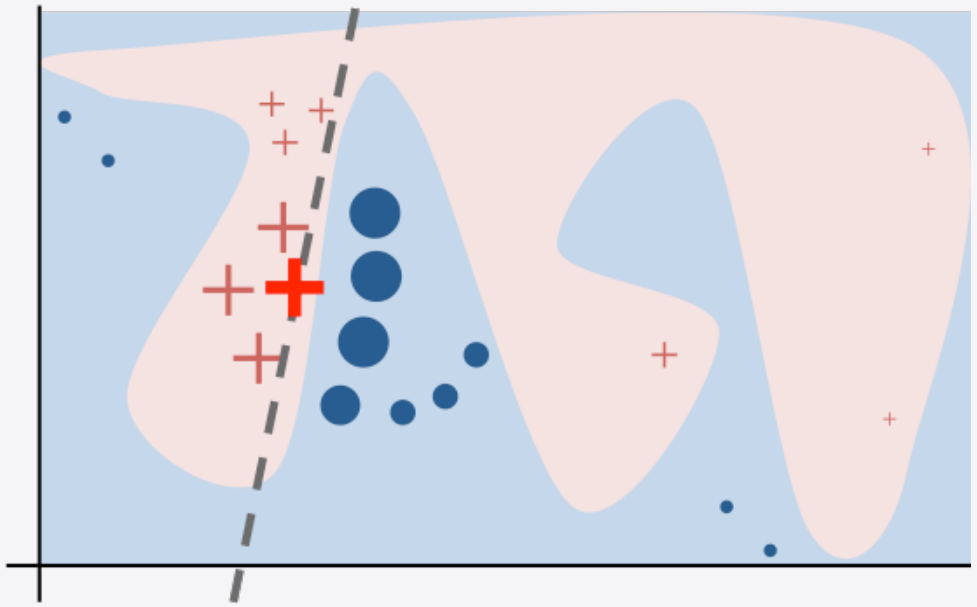
Note that

- The explanations are for one email or image each, i.e., they are local.
- The explanations involve a simplified version of the features for easier interpretability:
    - Regions (“superpixels”) of an image instead of individual pixels.
    - Presence/absence of a word for text instead of, e.g., word embeddings.

Theory: General Idea

Locally approximate the behaviour of a complex model by fitting a simple, interpretable model to sampled data close to the specific observation.

- Complex model: f
- Prediction to explain: $f(x)$ with $x \in \mathbb{R}^d$
- Perturbed sample: z
- Similarity between z and x: $\Pi_x(z)$
- Simple model: g



Riberio et al (2016)

Theory

What do we mean by ”simple” in the simple model?

- The model type should be interpretable, e.g., a linear model.
- The features of the simple model themselves should be interpretable. Interpretable representations of the features of the complex model can be generated, e.g.,
  - Image data: Superpixels instead of individual pixels.
  - Text data: Presence/Absence of words instead of word embeddings.
  - Tabular data: binary indicators for categorical variables, binned versions of continuous variables.
  - Notation: $x' \in \{0,1\}^{d'}$ and $z' \in \{0,1\}^{d'}$

Theory

How to pick the simple model?

The choice of g is based on two considerations: it should be

- Faithful to the complex model f → captured in the loss $\mathcal{L}$ between f and g, weighted by $\Pi_x(z)$
- as simple as possible → captured in the complexity Ω of g

The optimal model ξ of all possible models $g \in G$ is thus

$$\xi = \underset{g \in G}{\operatorname{argmin}} \mathcal{L}(f, g, \pi_x(z)) + \Omega(g)$$

Theory

Implementation choices

- Loss function: locally weighted quadratic loss
- Complexity measure: such that only models with up to K covariates are considered (by setting the complexity of model with more than K covariates to infinity)
- Consequence: optimisation problem becomes intractable
- Thus: approximate ξ in a two-step procedure

Theory

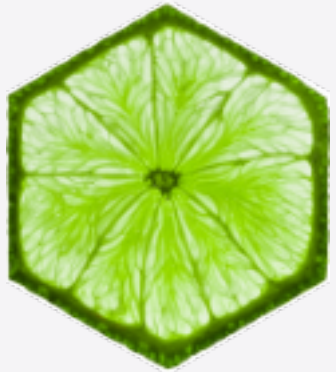
Approximation for ξ

- Select the best K covariates based on a procedure like forward selection or shrinkage via a LASSO model
- Fit a linear model with those K covariates

The lime package

The Python package lime is developed by the authors of the approach.

The R port is developed by Thomas Lin Pedersen.

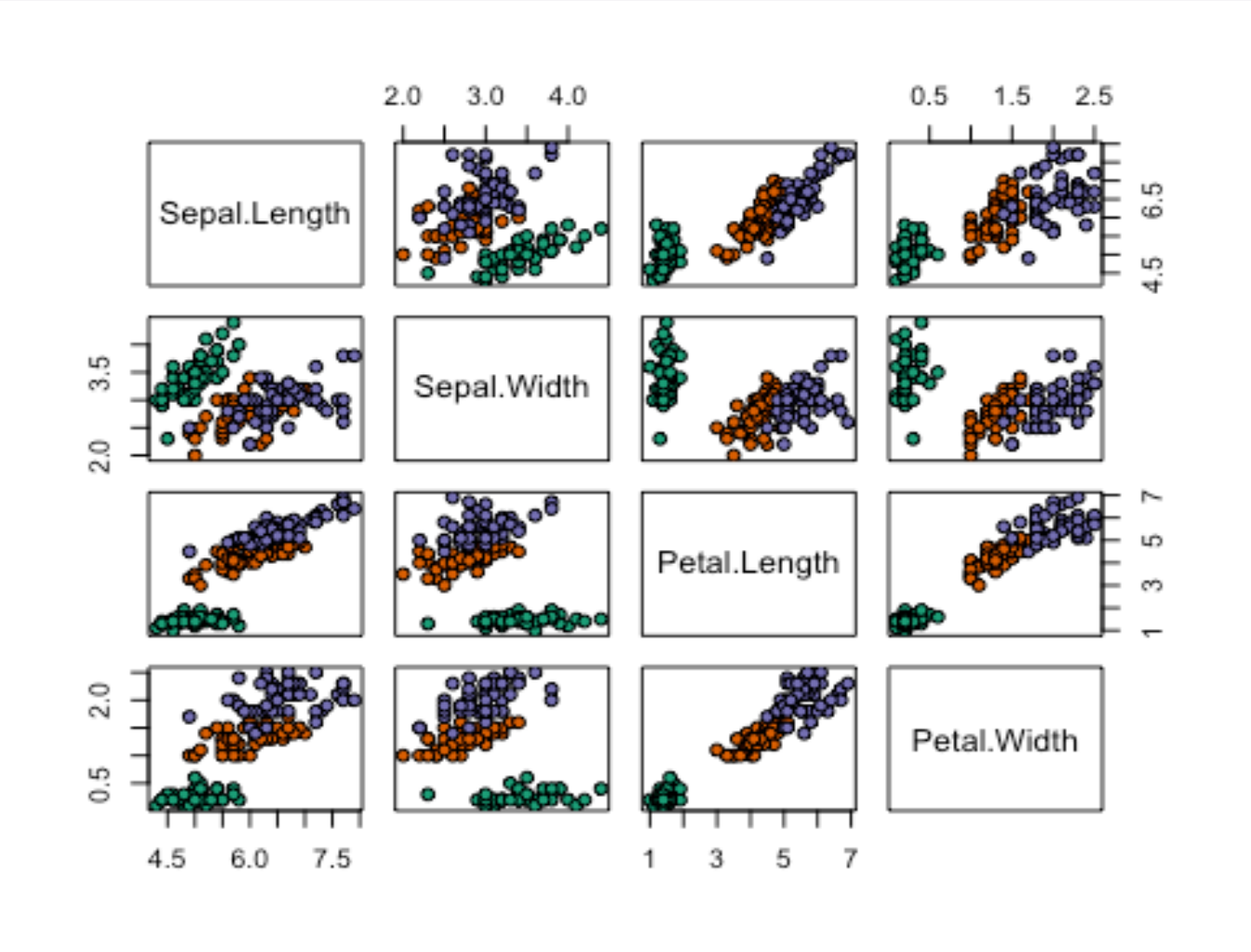


The main choices users get to make

- How to simplify the features to interpretable representations.
- How and how many of those interpretable representations are selected as covariates for the simple model.
- How to calculate the similarity between perturbed data points and original observation.

Iris Data

- 3 species
- 4 measurements
- 150 flowers



Classify Iris Data

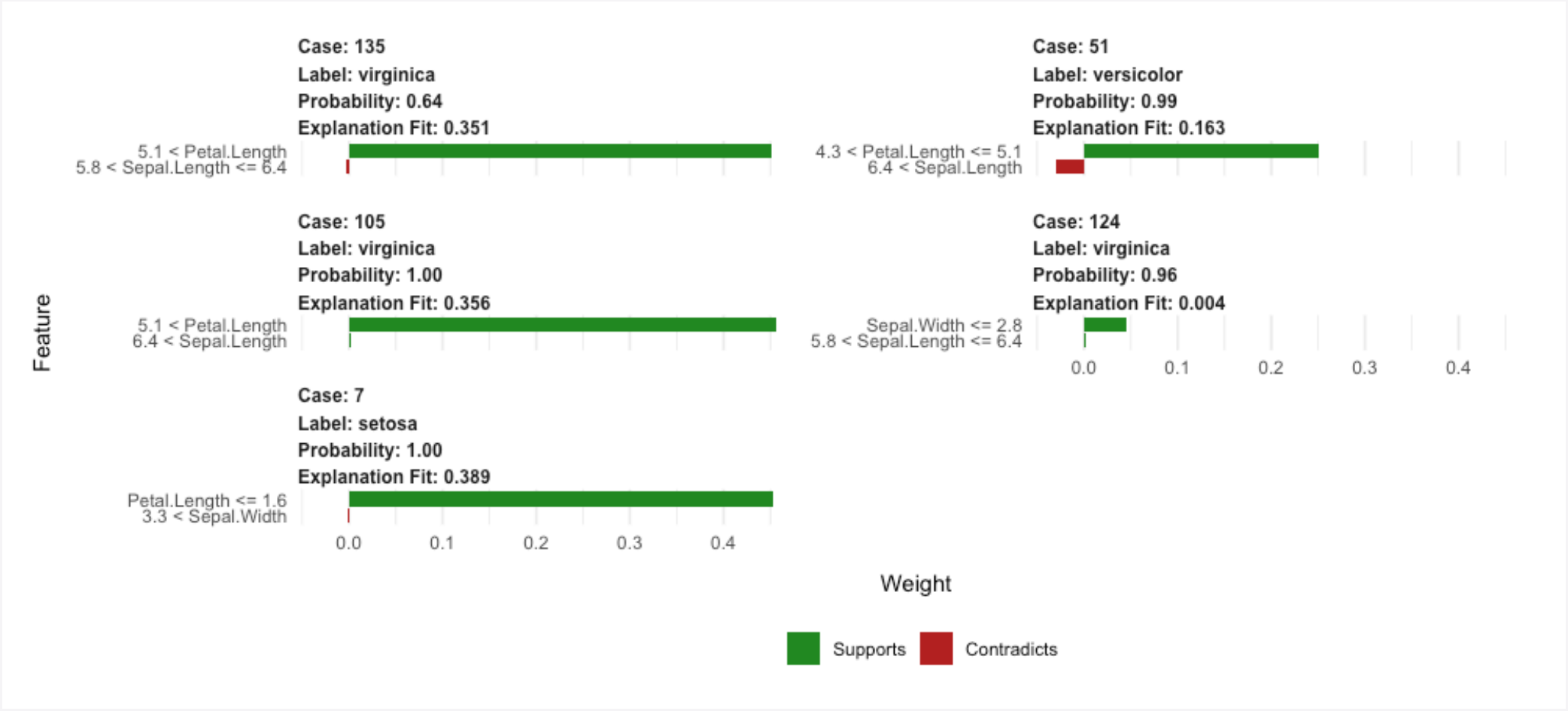
```
set.seed(304)

ind <- sample(1:nrow(iris), 5)

iris_explain <- iris[ind, 1:4]

iris_train <- iris[-ind, 1:4]

iris_lab <- iris[[5]][-ind]
```

```
library(caret)

model <- train(iris_train, iris_lab,
                    method = "rf")
```

Explain the Iris Classifications

```
library(lime)

explainer <- lime(iris_train, model,

                  bin_continuous = TRUE, n_bins = 4, quantile_bins = TRUE)


explanation <- explain(iris_explain, explainer,

                       n_labels = 1, n_features = 2,

                       n_permutations = 5000, feature_select = "auto")


plot_features(explanation)
```

Explain the Iris Classifications



Exercise

- Prepare the AdultUCI dataset from the arules package by omitting all rows with missing values, recoding education and income as unordered factors, and sampling 1000 rows.
- Set aside 5 rows as the observations we want to explain the predictions for.
- Fit a random forest on the remaining data with income as the response.
- Check the explanations for the predictions of the 5 observations set aside earlier. Do you trust this model? Explain your reasoning to your neighbour.

Further Concepts

Unifying Framework

Additive feature attribution methods

Framework by Lundberg and Lee (2017). Special cases:

- LIME
- DeepLIFT
- Layer-Wise Relevance Propagation
- Shapley values

Shapley Values

- Developed in game theory to distribute payoff of a game fairly amongst the players.
- Translated to ML: how to attribute a prediction to the features?
- Proven theoretical properties
    - Local accuracy
    - Missingness
    - Consistency

Shapley Values

Idea

- How much a player increases the final payoff when added to a team depends not only on their own skills but also on who else is in the team and their skills.
- Thus: Try all the combinations of players and average across all the marginal contributions.
- For ML: this requires refitting a lot of models for the different combinations of features present/absent in the model.

Shapley Values

Connection to LIME

- LIME works on binary features which can be translated to feature present/absent.
- Both methods are part of the same framework of Additive Feature Attribution Methods.
- We can choose the loss function, complexity measure, and similarity measure such that we can leverage the LIME procedure to approximate the Shapley values without refitting all the models.

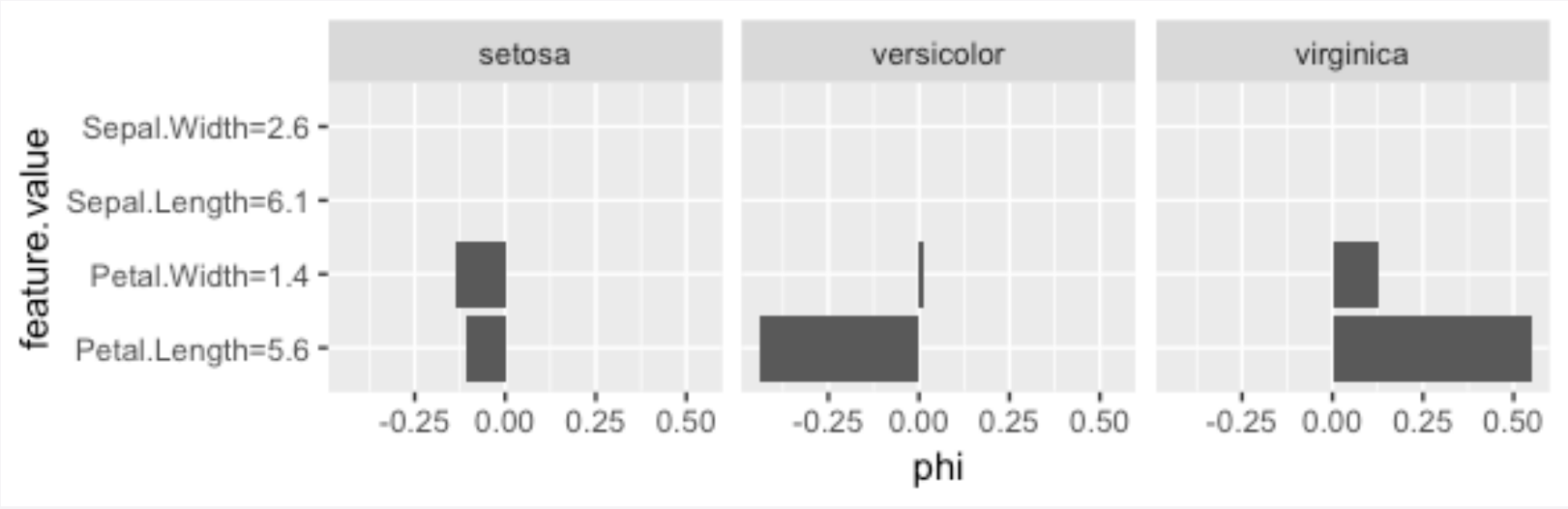
Shapley Values

Implementations in R

- **ShapleyR**: R implementation, not on CRAN (yet?)
- **iml**: R implmentation
- **shapper**: wrapper for Python library SHAP

Shapley Values

```
library(iml)

predictor <- Predictor$new(model, data = iris_train, y = iris_lab)

shapley <- Shapley$new(predictor, x.interest = iris_explain[1,])

shapley$plot()
```



Material / References

- Material at https://github.com/MangoTheCat/explainable-machine-learning-workshop
- DALEX: https://pbiecek.github.io/DALEX/
- LIME: Ribeiro et al. “Why Should I Trust You? Explaining the Predictions of Any Classifier” (ACM SIGKDD, 2016)
    - Python: https://github.com/marcotcr/lime
    - R: https://github.com/thomasp85/lime
- SHAP: Lundberg, Lee (2017). “A Unified Approach to Interpreting Model Predictions.” (NeurIPS, 2017)
    - ShapleyR: https://github.com/redichh/ShapleyR
    - iml: https://github.com/christophM/iml
    - shapper: https://github.com/ModelOriented/shapper