



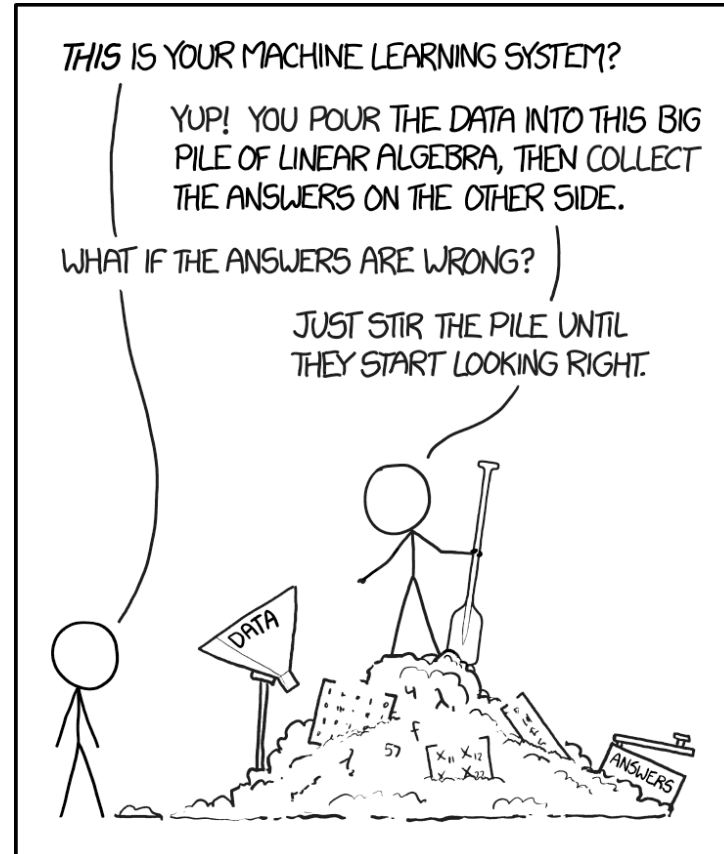
**ITMO UNIVERSITY**

Saint Petersburg, Russia

**Специализированные технологии машинного  
обучения /  
Advanced Machine learning Technologies**

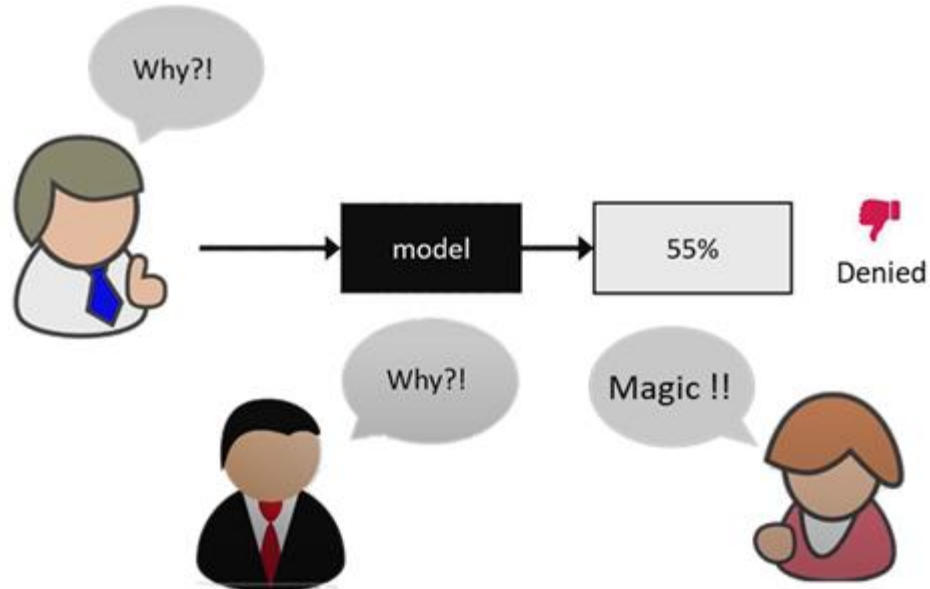
**Lecture 3 – Interpretable Machine Learning**

1. Interpretability of predictive models
2. Permutation Importance
3. Partial Dependence Plots
4. SHAP values
5. LIME



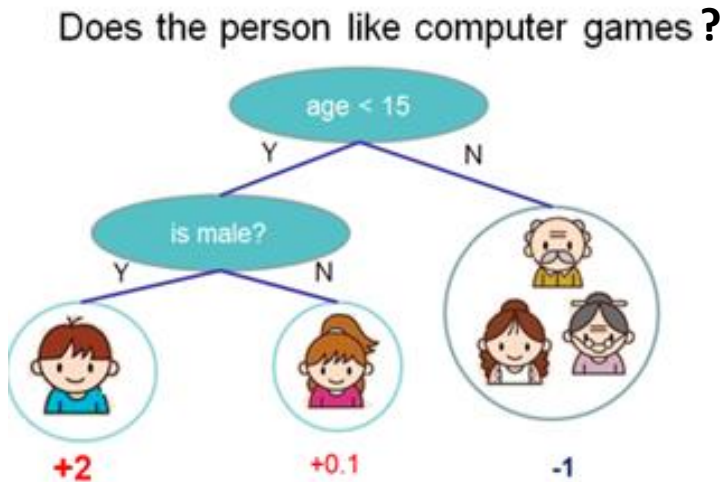
# Models interpretability.

- In expert systems, **interpretability** – is an important component of the explanation of the result.
- One need to find a trade-off between interpretability and accuracy: between **complex models** that handle diverse data and simple models that are easier to interpret but less accurate.

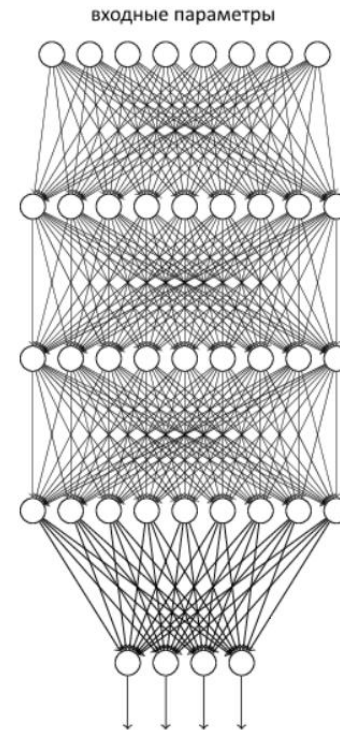


# Can you always explain how the model works?

Decision trees – very easy to interpret



Neural Networks – hard to interpret



# Black boxes

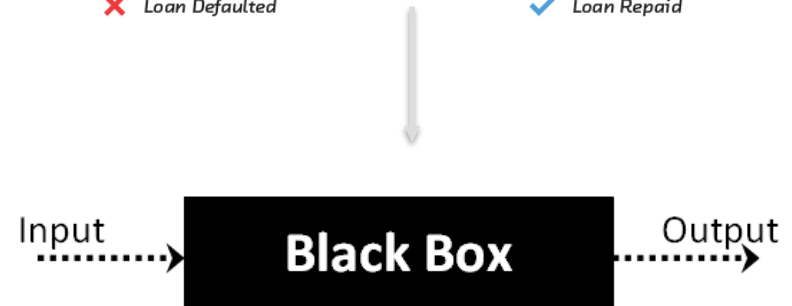
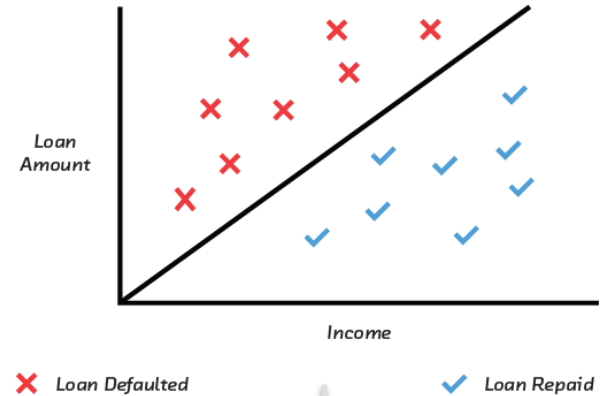
Dependencies between variables and prediction can be complex, non-monotonic, with many interdependencies.

Complex models are often referred to in machine learning as **black boxes**.

A set of variables is fed to the input of the model, on the basis of which it calculates its prediction, but how exactly this decision was made, what factors influenced it - the answers to these questions often remain hidden in the "**black box**" of the algorithm.

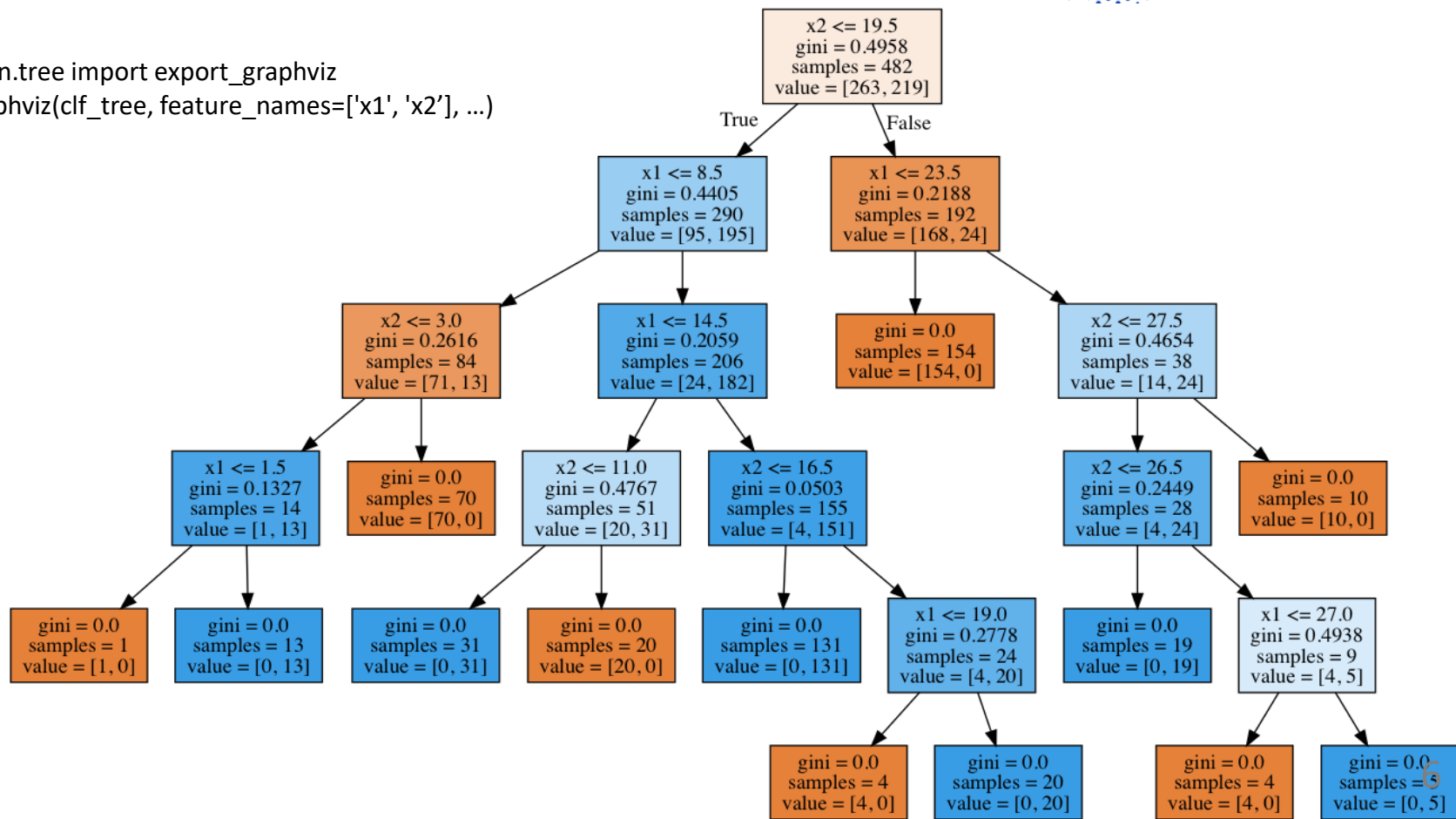
Even with **linear models**, things are not so simple. For example, if a model has several highly correlated variables (for example, age and the number of closed loans in a credit history), direct interpretation of the coefficients for these variables may not be such a trivial task.

The situation is even more complicated with **nonlinear models**, where the dependencies between variables and prediction can have a complex, non-monotonic form with many interdependencies.



# Decision trees

from sklearn.tree import export\_graphviz  
export\_graphviz(clf\_tree, feature\_names=['x1', 'x2'], ...)



**It is the degree to which a person can understand the reason for a particular decision.**

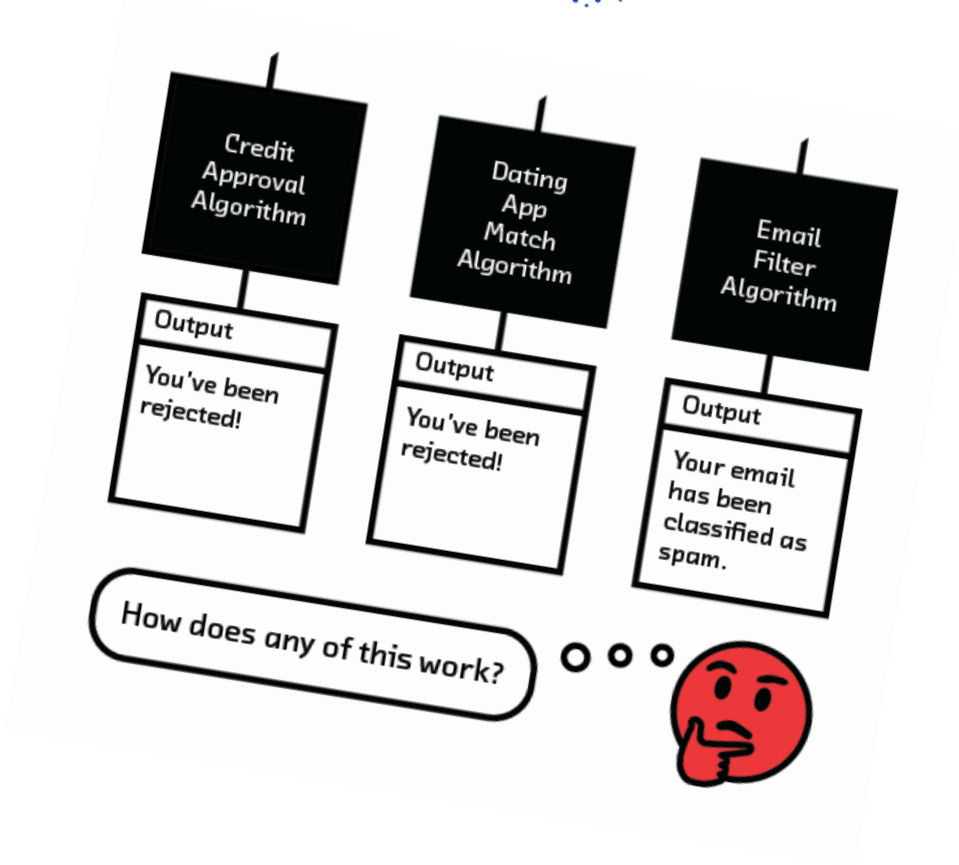
Who exactly is meant by “person” - an expert in a given subject area or an ordinary person “from the street”?

When interpreting the model, the expert explains

- how single features and model elements affect the target variable,
- what is their functional relationship,
- how changing model parameters affects accuracy and stability,
- how the individual parts of the sample are described (properties of the local model), etc.

## Interpretability (2/2)

This is any information in the form **accessible for perception**, which improves our understanding of **what factors** and **how exactly** influenced this particular prediction and the operation of the model as a whole. This information can take many forms, such as **graphics** or **textual explanations**.





# Why we need ML to be interpretable?

---

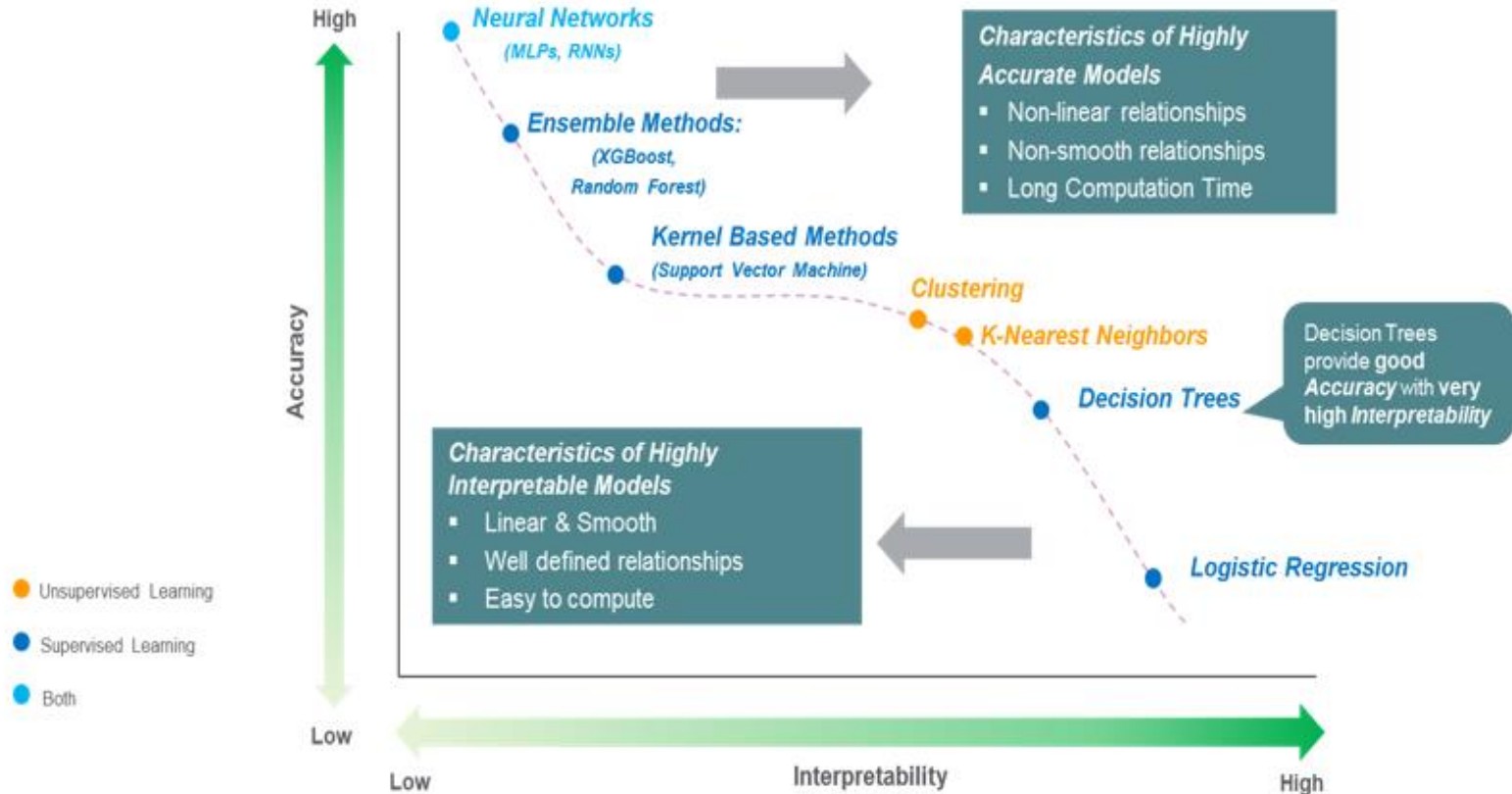
**Legal reasons.** On May 25, 2018, a new data protection regulation - General Data Protection Regulation (GDPR) came into force in the European Union. Among other things, article 13 of this regulation states that each data subject has a so-called "**right to explanation**", that is, to receive information about why an automated system made a particular decision, for example, a refusal to issue loan. Certain industries may have their own rules governing the operation of automated decision-making systems.

**Ethical reasons.** Understanding how a model works can help avoid ethically unacceptable situations such as discrimination against a specific group of clients.

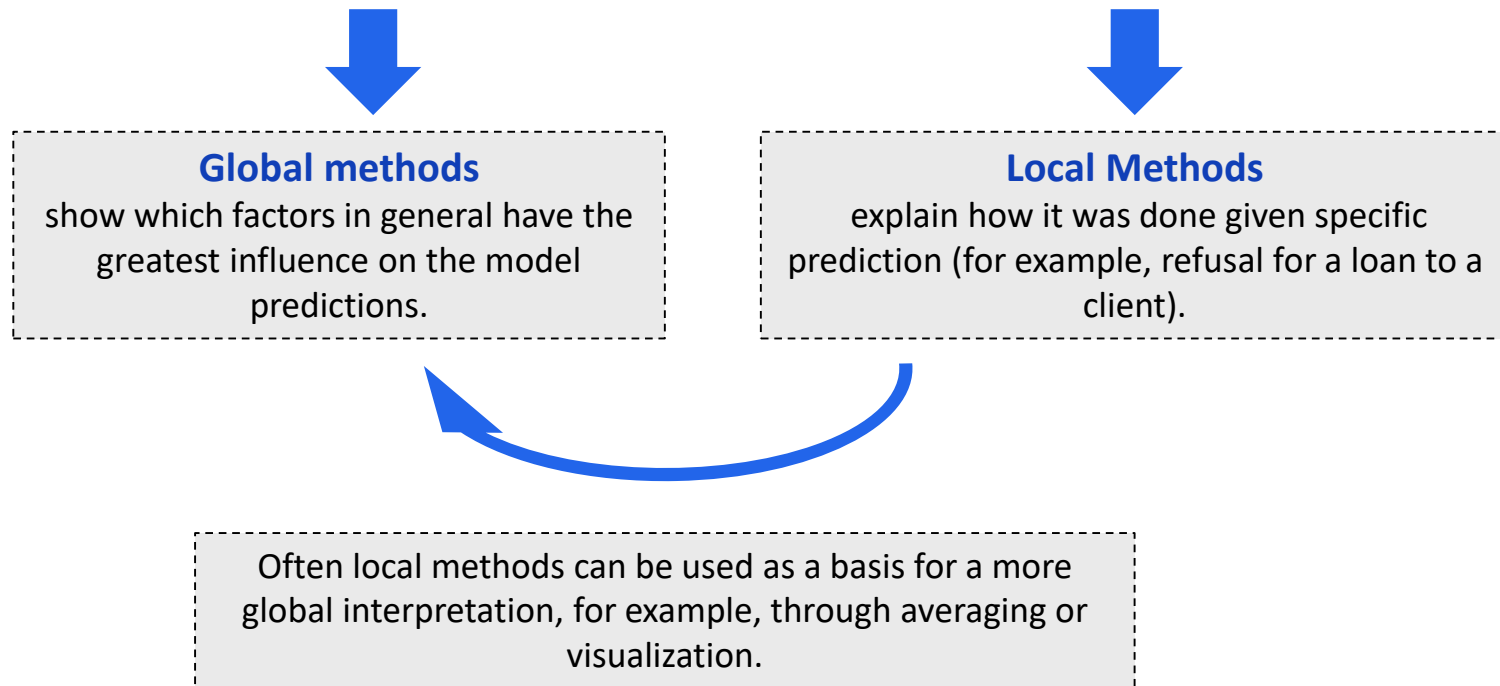
**Confidence.** Model end users will be more likely to trust them if they can obtain information about the influence of various factors on predictions.

**Testing and improving the model.** Understanding which variables and how affect model performance can help identify potential problems.

# Accuracy-interpretability tradeoff

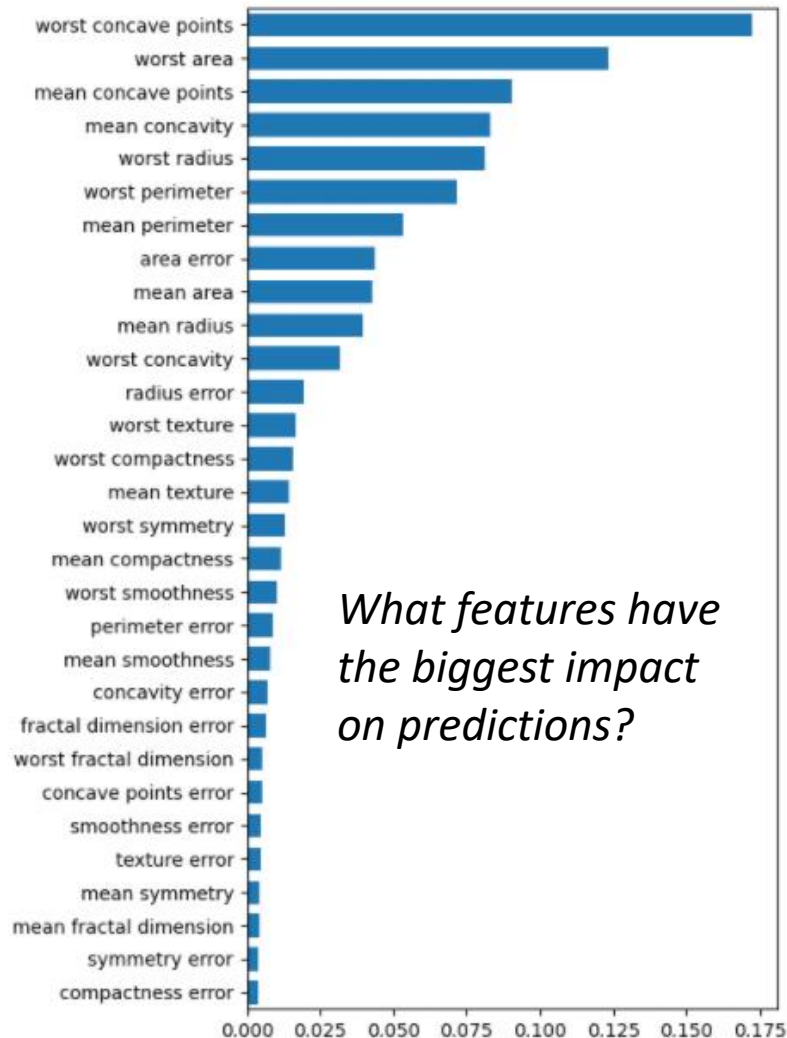


# How to interpret the models?



## Feature importance (1/3)

**Importance** is a kind of numerical indicator that shows how, **on average**, a given variable has on the predictions of a model, compared to other variables.



IVERSITY

The exact way the importance is calculated depends on the algorithm. The sklearn library, xgboost, lightGBM packages have built-in methods for evaluating feature importance for tree models:

### Gain

This measure shows the relative contribution of each feature to the model. For the calculation, we go through each tree, look at each node of the tree which feature leads to the partition of the node and how much the uncertainty of the model is reduced according to the metric (Gini impurity, information gain). For each feature, its contribution is summarized for all trees.

### Cover

Shows the number of observations which have used the feature. For example, you have 4 features, 3 trees. Suppose feature 1 in the tree nodes was used for 10, 5 and 2 observations in trees 1, 2 and 3, respectively for the entire dataset. Then for this feature the importance will be 17 ( $10 + 5 + 2$ ).

### Frequency

Shows how often the feature is found in nodes (not depending on the dataset). That is, the total number of feature partitions into nodes for each feature in each tree.

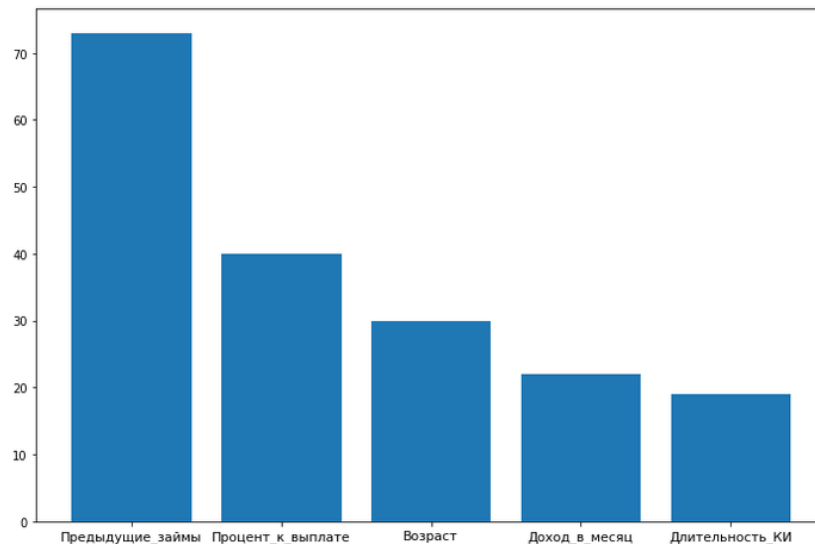
### Disadvantages:

**Does not provide any information about how exactly a particular variable affects the decision-making process.**

For example, looking at this graph, we cannot tell whether the increase or decrease in the probability of default increases or decreases in the number of previous loans.

**Does not solve the problem of correlated variables**

- if two variables are related, but variable 1 performs slightly better in the model than the other, then the importance of variable 2 is likely to be extremely low, which may not reflect the real state.



# Permutation Importance

Making the prediction on **permuted data** and evaluate the difference with baseline.

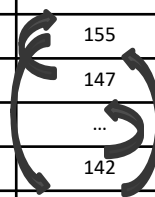
## Algorithm:

1. Train the model.
2. Shuffle values in one column;
3. Calculate predictions using the resulting dataset.
4. Calculate the forecast accuracy for the resulting dataset. The change in precision indicates the importance of the variable that was just shuffled.
5. Take the original dataset. Repeat step 2 with the next column in the dataset.

## Example:

Predict the height of a man in 20 years.

Height at 20	Height at 10	...	# of friends at 10
182	155	...	20
175	147	...	10
...	...	...	...
156	142	...	8
153	130	...	24



<https://www.kaggle.com/mathan/fifa-2018-match-statistics>

Predict which team player will receive a "Man of the Match" for each game.

Available Data:

- **Playing Team/Opponent Team**
- **Number of goals scored by teams**
- **Ball Possession %**
- **Number of attempts to score goal**
- **Number of opponent team's attempts blocked by the team**
- **Number of corner shots used**
- **Number of off-side events**
- **Number saves by the goal keeper**
- **Pass Accuracy %**Percentage of passes that reached the same team player as aimed
- **Total number of passes by the team**
- **Total distance covered by the team members in this game**
- **Number of Yellow & Red warning received**
- **Man of the Match - Did this team member win Man of the Match?**
- ...



```
import eli5
from eli5.sklearn import PermutationImportance

perm = PermutationImportance(my_model, random_state=1).fit(val_X, val_y)
eli5.show_weights(perm, feature_names = val_X.columns.tolist())
```

Weight	Feature
0.1750 ± 0.0848	Goal Scored
0.0500 ± 0.0637	Distance Covered (Kms)
0.0437 ± 0.0637	Yellow Card
0.0187 ± 0.0500	Off-Target
0.0187 ± 0.0637	Free Kicks
0.0187 ± 0.0637	Fouls Committed
0.0125 ± 0.0637	Pass Accuracy %
0.0125 ± 0.0306	Blocked
0.0063 ± 0.0612	Saves
0.0063 ± 0.0250	Ball Possession %
0 ± 0.0000	Red
0 ± 0.0000	Yellow & Red
0.0000 ± 0.0559	On-Target
-0.0063 ± 0.0729	Offsides
-0.0063 ± 0.0919	Corners
-0.0063 ± 0.0250	Goals in PSO
-0.0187 ± 0.0306	Attempts
-0.0500 ± 0.0637	Passes

- The values at the top are the most important features, while the values at the bottom are less important.
- The first number is how much the accuracy of the model has decreased due to random shuffling.
- The number after the  $\pm$  shows how the precision varies from one permutation to the other.
- In the case of negative values, the predictions on the permuted data turned out to be more accurate than the real data (feature is not important + the dataset is not big enough).

# Permutation Importance

```
import eli5
from eli5.sklearn import PermutationImportance

perm = PermutationImportance(my_model, random_state=1).fit(val_X, val_y)
eli5.show_weights(perm, feature_names = val_X.columns.tolist())
```

Weight	Feature
0.1750 ± 0.0848	Goal Scored
0.0500 ± 0.0637	Distance Covered (Kms)
0.0437 ± 0.0637	Yellow Card
0.0187 ± 0.0500	Off-Target
0.0187 ± 0.0637	Free Kicks
0.0187 ± 0.0637	Fouls Committed
0.0125 ± 0.0637	Pass Accuracy %
0.0125 ± 0.0306	Blocked
0.0063 ± 0.0612	Saves
0.0063 ± 0.0250	Ball Possession %
0 ± 0.0000	Red
0 ± 0.0000	Yellow & Red
0.0000 ± 0.0559	On-Target
-0.0063 ± 0.0729	Offsides
-0.0063 ± 0.0919	Corners
-0.0063 ± 0.0250	Goals in PSO
-0.0187 ± 0.0306	Attempts
-0.0500 ± 0.0637	Passes

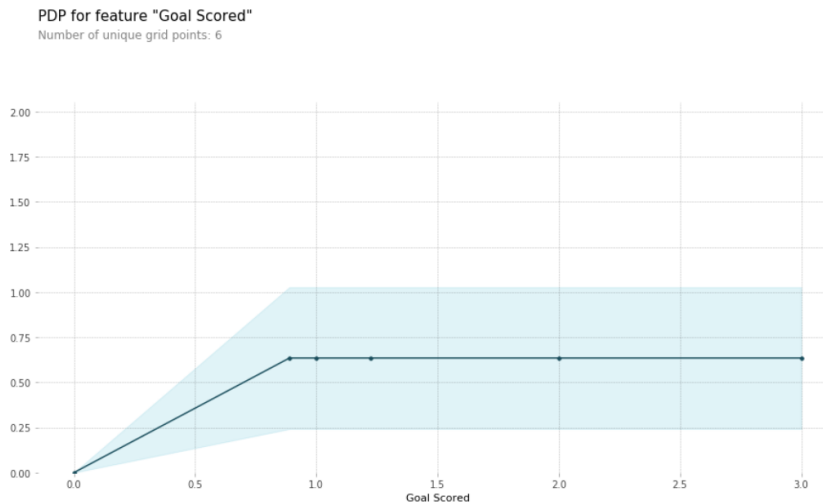
At the same time, it is not clear how the feature affects on the decision-making process If it has an average Permutation Importance.

It may mean that it has:

- great effect for several predictions, but not in general
- average effect for all predictions.

# Partial Dependence Plots (PDP)

The importance of the function shows which features most influence the forecasts, PDP shows how a **particular feature values** affects the forecast.



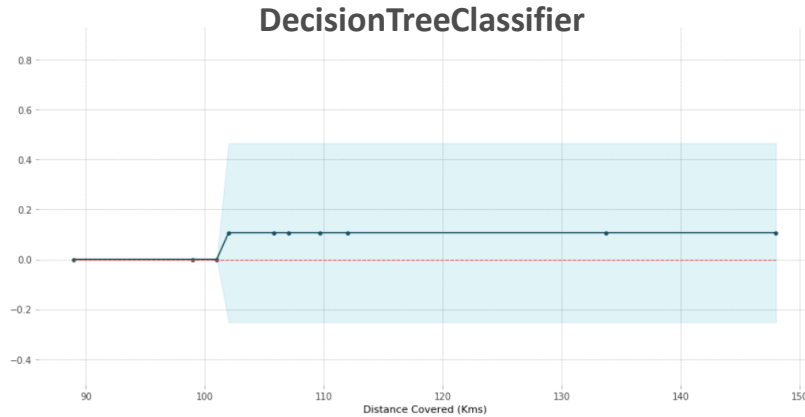
```
from pdpbox import pdp, get_dataset, info_plots
```

```
# Create the data that we will plot  
pdp_goals = pdp.pdp_isolate(model=tree_model,  
dataset=val_X, model_features=feature_names,  
feature='Goal Scored')
```

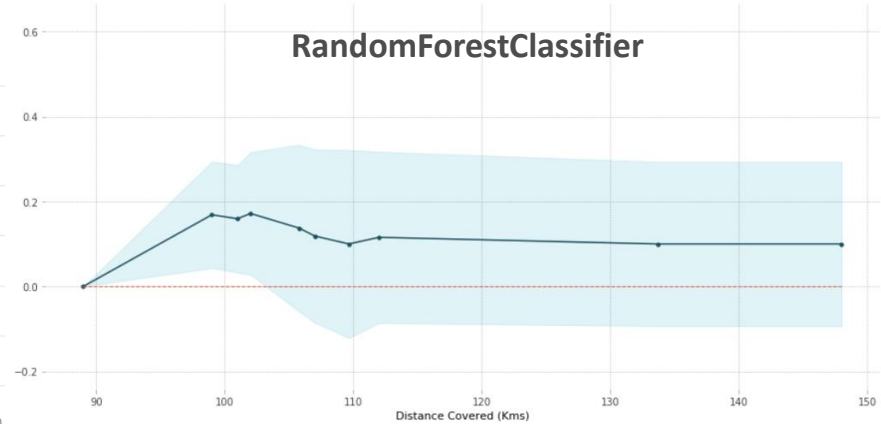
```
# plot it  
pdp.pdp_plot(pdp_goals, 'Goal Scored')  
plt.show()
```

# Partial Dependence Plots (PDP)

PDP for feature "Distance Covered (Kms)"  
Number of unique grid points: 10



PDP for feature "Distance Covered (Kms)"  
Number of unique grid points: 10



- Based on the first model, there is a better chance of winning “Man in a Match” if the players walk a total of 100 km during the game. Running a lot more causes lower forecasts on the second model, though.
- In general, the smooth shape of this curve seems more reasonable than the step function from the decision tree model.

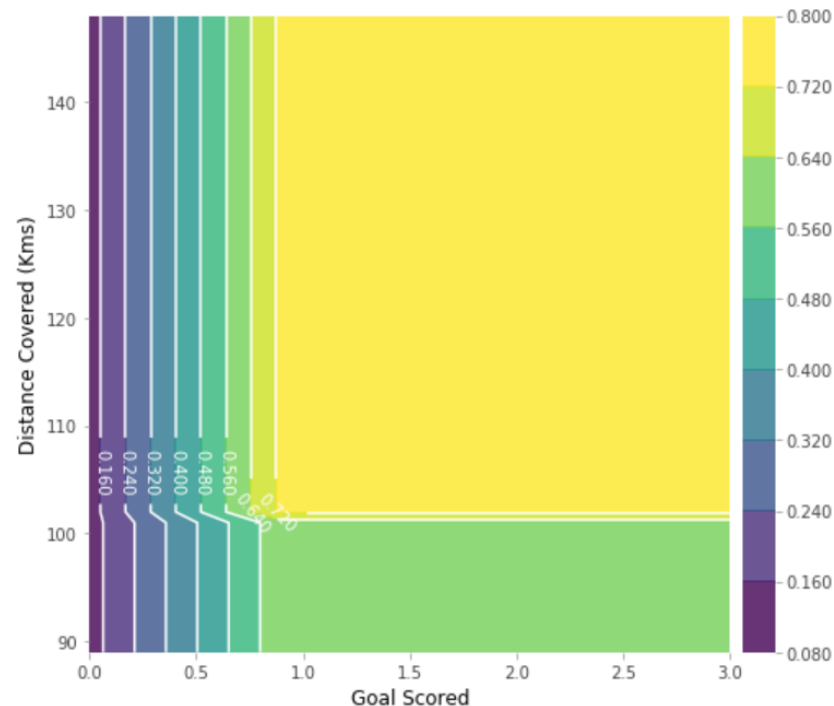
## 2D Partial Dependence Plots

Affecting of the **pair of features** on the forecast

```
features_to_plot = ['Goal Scored', 'Distance Covered (Kms)']  
inter1 = pdp.pdp_interact(model=tree_model, dataset=val_X,  
model_features=feature_names, features=features_to_plot)
```

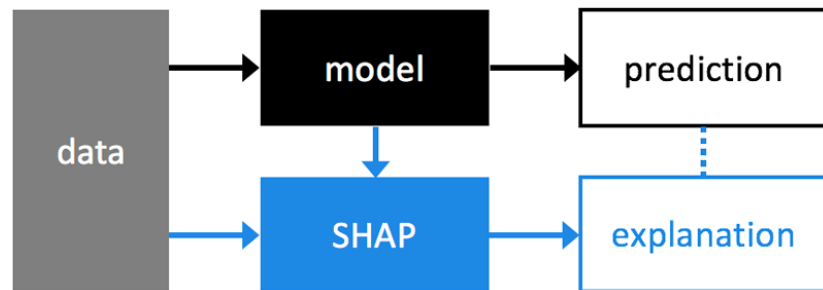
```
pdp.pdp_interact_plot(pdp_interact_out=inter1,  
feature_names=features_to_plot, plot_type='contour',  
plot_pdp=True)
```

The highest predictions are obtained when a team scores at least 1 goal and runs a distance of about 100 km. If they score 0 goals, the distance traveled doesn't matter.



## Main Idea:

to understand how the model worked for specific prediction



## Examples of the questions to the model, which can be answered via SHAP:

- Based on the model, the bank did not approved credit to someone, but the bank ought to explain the reason for the refusal.
- The healthcare professional wants to identify, which factors are causing each patient to get sick so that they can “eliminate” the risk factors through targeted medical interventions.
- To what extent was the forecast determined by the fact that **instead of** a certain baseline number of goals, the team scored 3 goals?

- The SHAP framework calculates Shapley values to assess the **importance of features**.
- To assess the importance of a feature, the predictions of the model **with and without** this feature are evaluated.
- Shapley's values come from game theory.

Consider a scenario: a group of people are playing cards. How to distribute the prize fund between them in accordance with their contribution? A number of assumptions are made:

- The amount of the reward for each player is equal to the total amount of the prize pool;
- If two players make an equal contribution to the game, they receive an equal reward;
- If the player has not made any contribution, he does not receive a reward
- If a player has played two games, then his total reward consists of the sum of rewards for each of the games

We represent the **features of the model** as players, and the prize fund as the **prediction** of the model.

Equation for the Shapley value  
for the i-th feature:

$$\phi_i(p) = \sum_{S \subseteq N/\{i\}} \frac{|S|!(n - |S| - 1)!}{n!} (p(S \cup \{i\}) - p(S))$$

where:

$p(S \cup \{i\})$  — Model prediction with i-th feature

$p(S)$  — Model prediction without i-th feature

$n$  — Number of features

$S$  — Random batch of the features without i-th feature

The Shapley value for the i-th feature is calculated for each data sample (for example, for each client in the sample) on all possible combinations of features (including the absence of all features), then the obtained values are summed up and the final importance of the i-th feature is obtained.

*These calculations are extremely expensive, so various algorithms are used to optimize the calculations under the hood.*



# Shapley values

We want to assess the importance of features for predicting “*whether a person likes computer games*”.

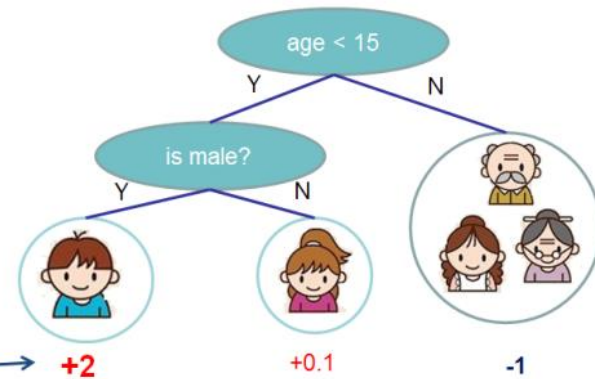
Let’s, for simplicity, apply only two features:

- age
- gender

Input: age, gender, occupation, ...



Does the person like computer games



prediction score in each leaf

Let’s Calculate Shapley value for the feature “age” for Bobby:

We have two possible batches of features without “age” S:

- $\{\}$  — no features
- $\{\text{gender}\}$  — one feature - “gender”.

# Shapley values

## 1. No features

In this case model simply average the predictions in each branch of the tree:

$$[(2+0.1)/2+(-1)]/2=0.025.$$

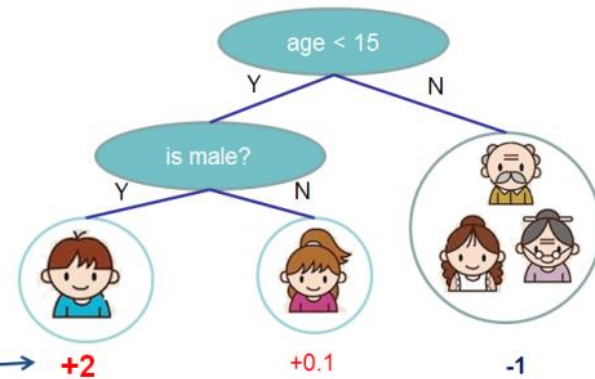
If we know the age, then the average of the model will be

$$(2+0.1)/2=1.05.$$

Input: age, gender, occupation, ...



Does the person like computer games



prediction score in each leaf

+2

+0.1

-1

For "No features" case Shapley value : 
$$\frac{|S|!(n - |S| - 1)!}{n!} (p(S \cup \{i\}) - p(S)) = \frac{1(2 - 0 - 1)!}{2!} (1.025) = 0.5125$$

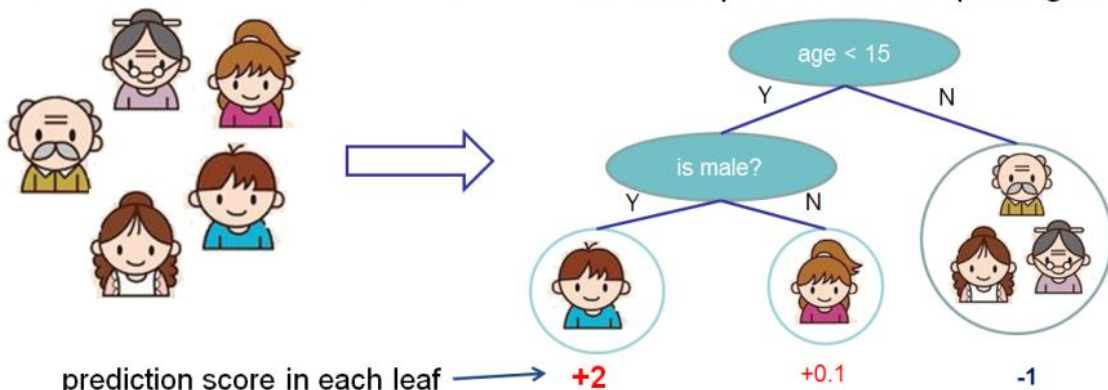
## 2. {gender}

For Bobby prediction without age  
is  $[(2+0.1)/2+(-1)]/2=0.025$ .

If we know gender and age, so that we  
have the most left branch, average is 2.

Input: age, gender, occupation, ...

Does the person like computer games



$$\frac{|S|!(n - |S| - 1)!}{n!} (p(S \cup \{i\}) - p(S)) = \frac{1(2 - 1 - 1)!}{2!} (1.975) = 0.9875$$

Finally, Shapley value for “age” feature for Bobby is:  $\phi_{AgeBobby} = 0.9875 + 0.5125 = 1.5$

## SHAP values. One more example

Imagine you have to recruit an 11-person soccer team from a large slate of candidates. Without a team, your chance of winning is 0%.

- You randomly choose the first player named Artem D. He is a good footballer, a well-known scorer, but alone he is not able to successfully oppose an opponent, and the **probability of victory rises to only 2%.**
- Then, you add Alexander K to the team. Alone, he could increase the probability of the team winning by 3%, but he interacts very well in conjunction with the already chosen Artem D., so the overall **probability of victory grows from 2% to 8%.**
- The next player on your list is goalkeeper Igor A. The goalkeeper is one of the most important elements of the team, and the **probability of winning increases dramatically - from 8% to 18%.**
- The next player is again the goalkeeper, Yuri L. If he was added to the team before Igor A., then he could increase the probability of winning by the same 10%. However, having two goalkeepers on the field does not make any sense - and the **probability of winning remains at 18%.**

The analogy with machine learning is easy to guess here, the team is the model, and the players are the variables. As you can see, **the influence of each variable** on the prediction (the probability of winning) **depends**, among other things, **on the presence and absence of other variables in the model.**

The main idea of the SHAP method is to calculate the contribution of each variable for all possible combinations of other variables in the model. The final contribution of each variable is calculated **as a weighted average of all possible contributions**.



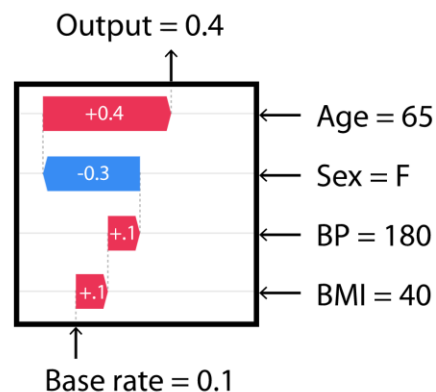
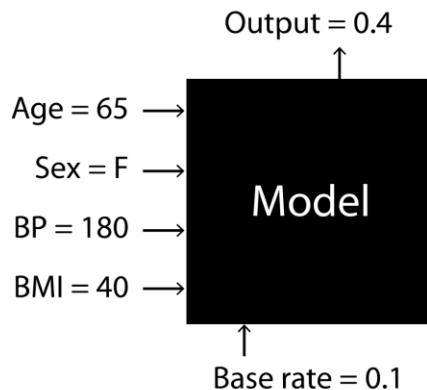
For example, if this value for Artem A. is 7%, it means that among all possible combinations of the team's composition, he, on average, increases the probability of winning by 7%. But in specific case this value can be extremely higher or lower.

# SHAP framework

```
import shap
explainer = shap.TreeExplainer(my_model)
shap_values = explainer.shap_values(data_for_prediction)
```



SHAP



```
import shap
explainer = shap.TreeExplainer(my_model)
shap_values = explainer.shap_values(data_for_prediction)
shap.initjs()
shap.force_plot(explainer.expected_value[1], shap_values[1], data_for_prediction)
```

SHAP values for all features are summed together to explain why the forecast differs from the baseline



- Red - an increase in the forecast, blue - a decrease.
- The horizontal size shows the magnitude of the feature's effect.
- Subtracting the length of the blue bars from the length of the red bars gives the distance from the baseline value to the specific forecast.

# SHAP summary plots

## Point characteristics:

- **Vertical position** is a feature name;
- **Color** - the magnitude of the influence of a feature for a particular prediction;
- **The horizontal position** indicates whether the impact of that value caused a higher or lower prediction.

## Examples of conclusions:

- The model ignored the red and yellow cards.
- Usually, a yellow card does not affect the forecast, but there is an edge case where a large number of yellow cards caused a much lower forecast.
- Goals scored impact on the positive forecast more



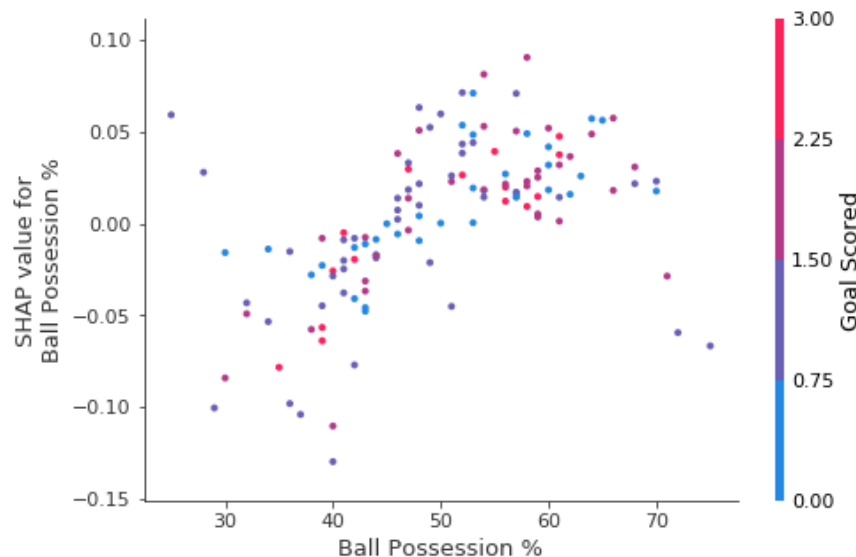


# SHAP Dependence Contribution Plots

SHAP plots for a specific feature are similar to PDPs, but add much more detail. Each point represents a data record.

- the horizontal position is the actual value from the dataset,
- the vertical position shows the effect of the value on the forecast.

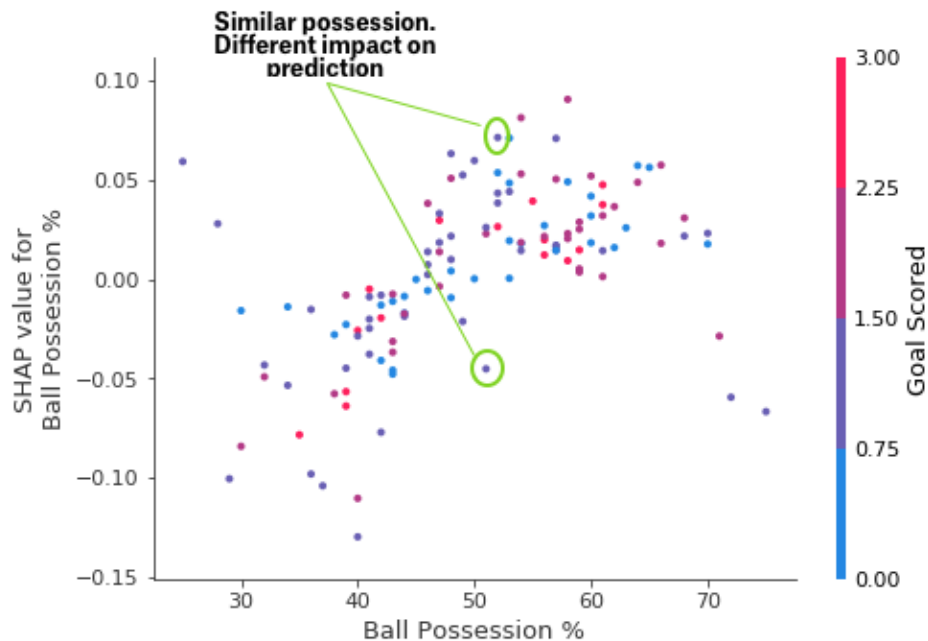
An uptrend indicates that the greater the proportion of time in possession, the higher the model's prediction for the probability of “Man of the Match” award.



# SHAP Dependence Contribution Plots

The location of the points indicates that other features interact with “Ball Possession” and affect on its Shapley Value.

For example, the graph shows two points with the same ball possession values. If the feature values are equal, the forecast value can be either increased or decreased.

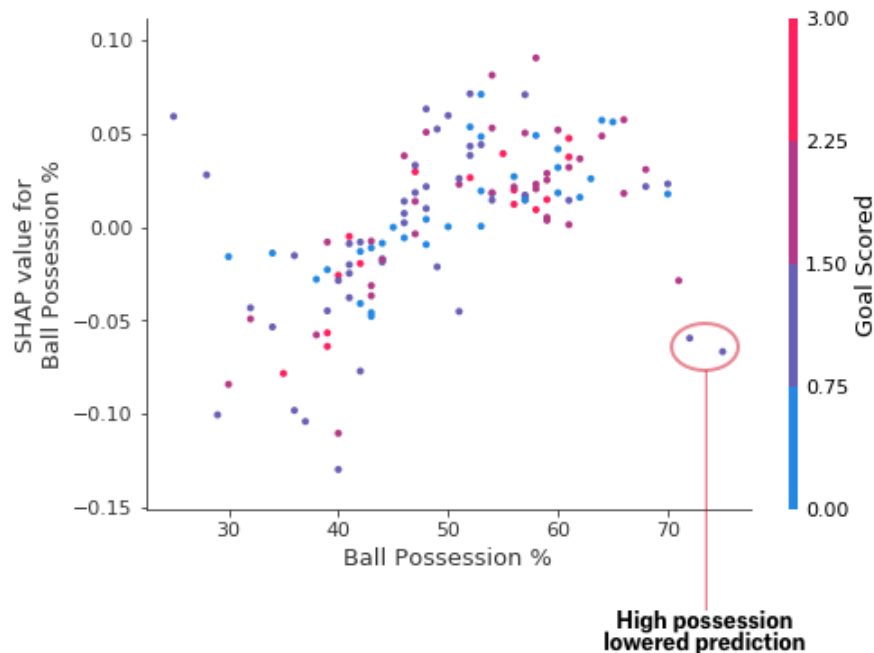


# SHAP Dependence Contribution Plots

The two highlighted points are far from the uptrend and have the same color (the team scored one goal).

You can interpret it like this:

- in general, having the ball increases a team's chance of a player winning an award.
- but if they score only one goal, this trend is reversed, and the referees involved in the awarding ceremony can “punish” them for possessing so much ball and scoring little.

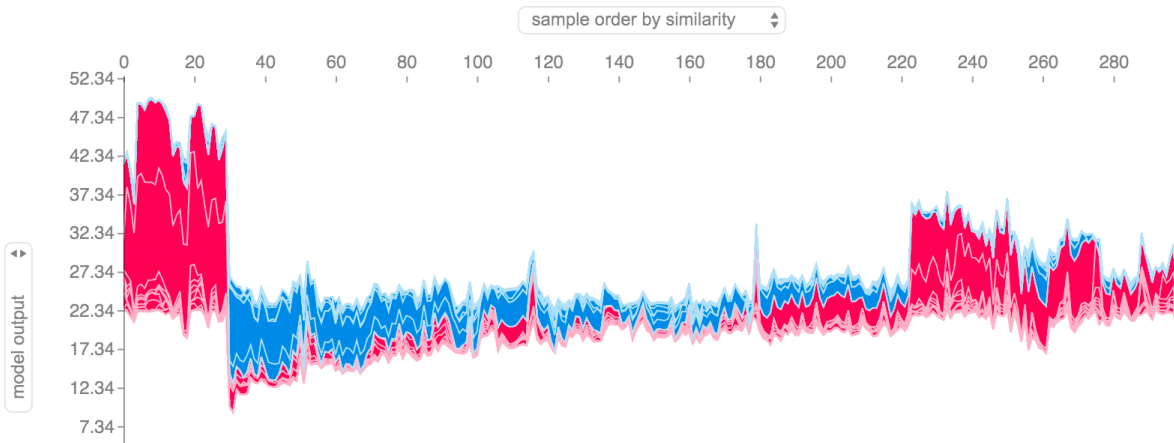
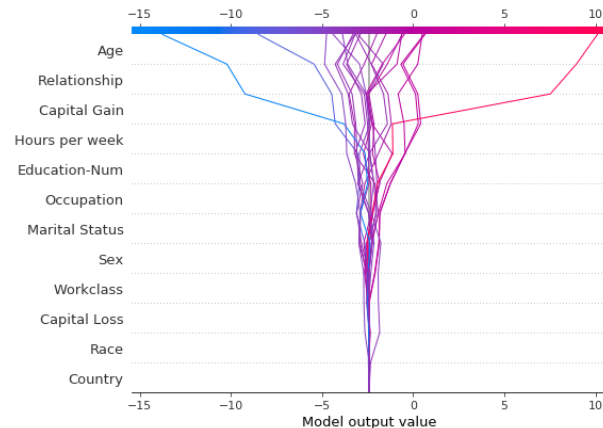
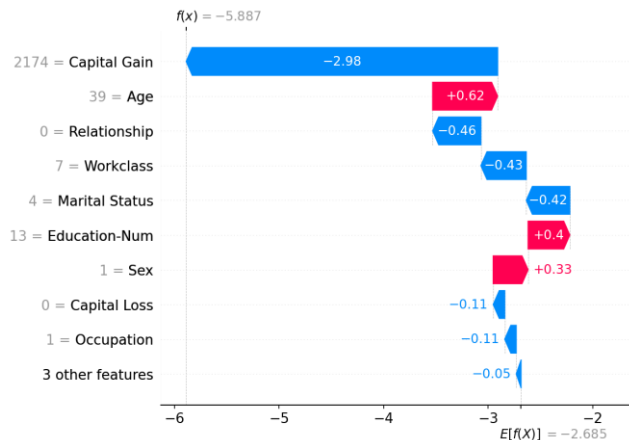


# SHAP framework

[https://shap.readthedocs.io/en/latest/api\\_examples.html#plots](https://shap.readthedocs.io/en/latest/api_examples.html#plots)



Aggregating SHAP values for individual forecasts provides more detailed alternatives to Permutation Importance and Partial Dependence Plots.



## Advantages:

- visualization and interpretation
- reasoned calculation of the importance of features
- the ability to evaluate features for a specific subsample of data

SHAP framework have **Explainers** for different model types:

- **shap.TreeExplainer**
- **shap.DeepExplainer**
- **shap.KernelExplainer**
- ...

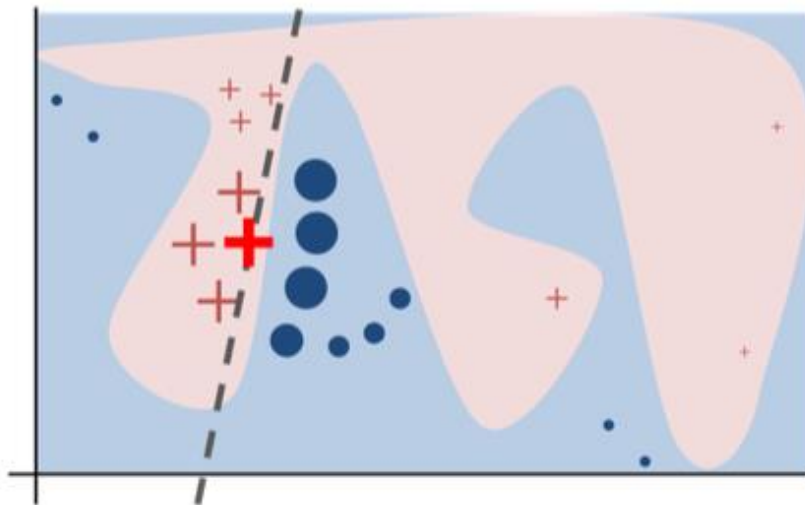
The authors of SHAP have proposed a very efficient implementation of the algorithm for different **ensembles of trees**, which demonstrate excellent prediction quality in many practical problems.

Mean assumption:

**each complex model is linear at a local scale.**

1. A simple model is trained to explain the observation.
2. The simple model is then used to locally explain the predictions of the more complex model.
3. Estimate values based on a set of samples rather than calculating them accurately

As a result: **less computations**

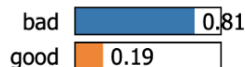


## Algorithm:

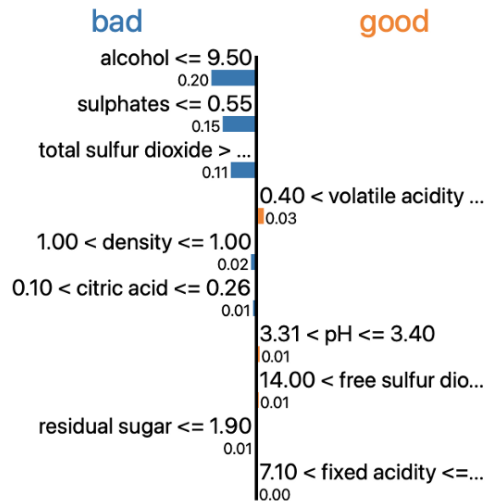
1. Given an observations, create new data with slight value modifications.
  2. Compute similarity distance measure between original observation and new observations.
  3. Apply selected machine learning model to predict on new data.
  4. Select **m** number of features which best describe predictions.
  5. Fit a simple model to the new data, explaining the complex model predictions with **m** features from the new data weighted by its similarity to the original observation .
  6. Use the resulting feature weights to explain local behaviour of the model (particular observation).
- 
- From the above, it is clear that there is a room for optimizations.
  - Quality of interpretation depends on
    - how permutations are created,
    - how the similarity of permutations is calculated,
    - what and how many features are selected, and
    - which model is used as a simple model.
  - Some of these options are predefined, others can be user-defined.

## Example of output:

Prediction probabilities



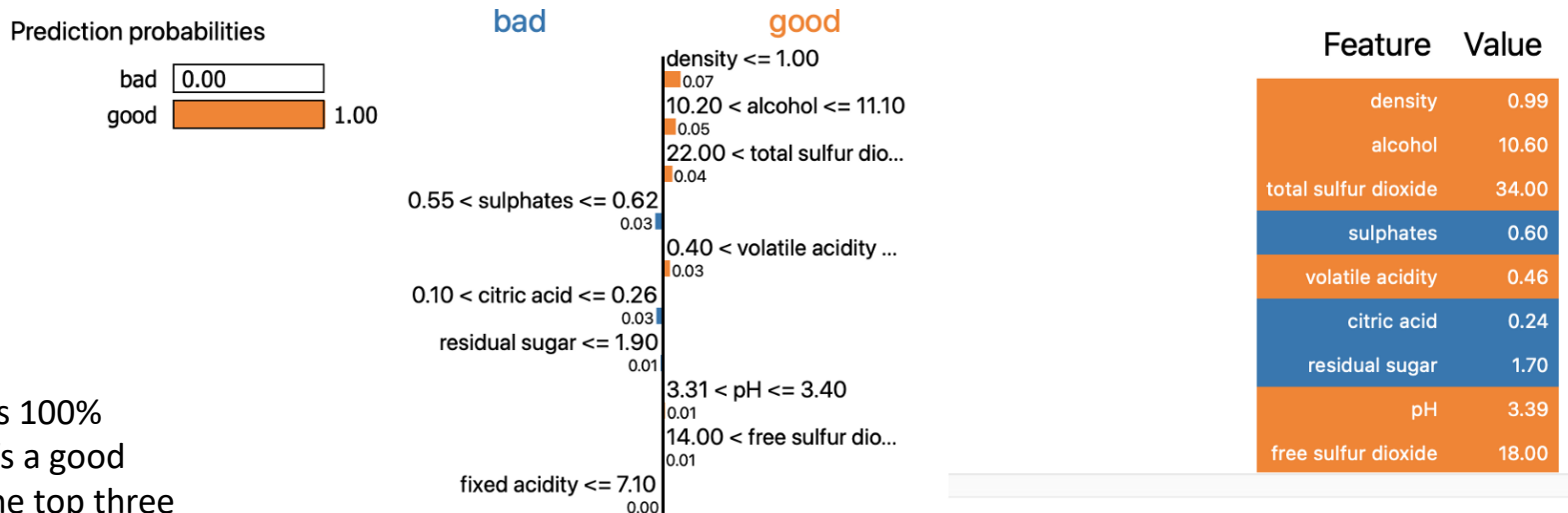
The model is 81% sure that this is a bad wine. The alcohol, sulfate and total sulfur dioxide content increases the likelihood that the wine will be classified as bad. Volatile acidity («летучая кислотность») is the only one that reduces it.



Feature	Value
alcohol	9.50
sulphates	0.48
total sulfur dioxide	102.00
volatile acidity	0.50
density	1.00
citric acid	0.17
pH	3.39
free sulfur dioxide	21.00
residual sugar	1.60



## Example of output:



The model is 100% confident it's a good wine, and the top three predictors show it.

Interpreting even the most complex ML models is not impossible.

Local and global interpretation techniques allow you to better understand the internal logic behind automated decision making.

