

g-RIPS 2023 Fujitsu Group

# Design and Development of Explainable AI with Wide Learning

*Group Members:*

Lily Ge (Northwestern University)

Yuki Kimura (Musashino University)

Takeshi Nakashima (Musashino University)

Molly Noel (Rensselaer Polytechnic Institute)

*Academic Mentor:*

Jesica Bauer (Rensselaer Polytechnic Institute)

*Industry Mentor:*

Dr. Hiroyuki Higuchi (Fujitsu Limited)

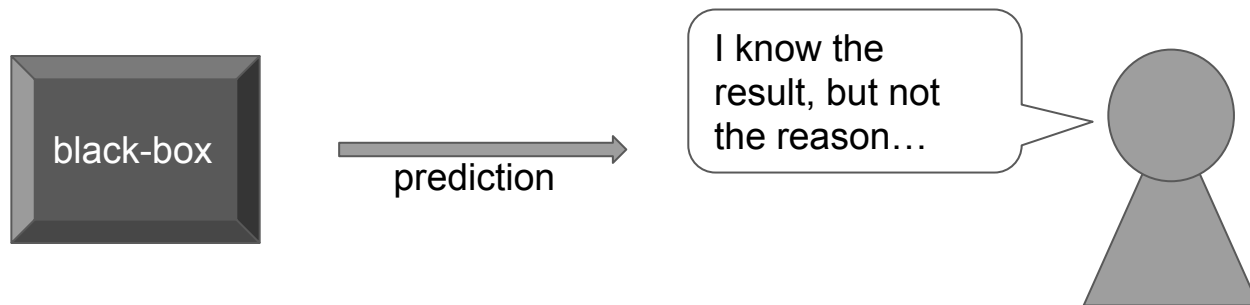
# Outline

- Motivation
  - What is Explainable AI?
  - Explainability Approaches
- Wide Learning
- Our Approach
  - User selection LASSO
  - Model selection
    - Rademacher complexity
      - result
  - Interface Specification and Development
    - result
  - Pilot Survey for New Interface
    - result
- Conclusion

# What is Explainable AI?

# Modern Model

- Many modern AI predictions are based on black-box models/systems and lack explanatory power and conviction.
- This lack has a significant impact on industries such as finance and medicine, which are driven by trust.



# Explanation Approaches

“I recommend you do this, and here is why...”

**Recommendation-driven**

A

“Based on your situation, here are some options...”

**Hypothesis-driven**

B

# “Hello, Wide Learning!” by Fujitsu Limited

- Wide Learning is a form of the hypothesis-driven approach
- We aim to enhance it and develop a more hypothesis-driven Explainable AI (XAI) tool

## Combination of important items

Oviparous\_No

Sized about the same as a cat?\_Yes

Breathes using lungs\_Yes  $\wedge$  Fins\_Has

Wings\_Does not have  $\wedge$  Eats meat\_No  $\wedge$  Spine\_Has

Teeth\_Has  $\wedge$  Breathes using lungs\_Yes

Flies\_Does not  $\wedge$  Aquatic\_No

# How Can We Improve Wide Learning?

# Pilot Survey

- Open-ended questions to elicit people's thoughts on the existing Wide Learning tool
- Anonymized and expected to take approximately 15 minutes to complete
- Distributed through the `#students-grips2023` Slack channel and received 10 responses
- Survey responses informed how we built the new user interface



# Existing Interface of “Hello, Wide Learning!”

Step 1

Step 2

Step 3

Step 4

Step 5

## Training

1. Upload training data
2. Get important feature combinations generated by Wide Learning
3. Assign weights to the combinations

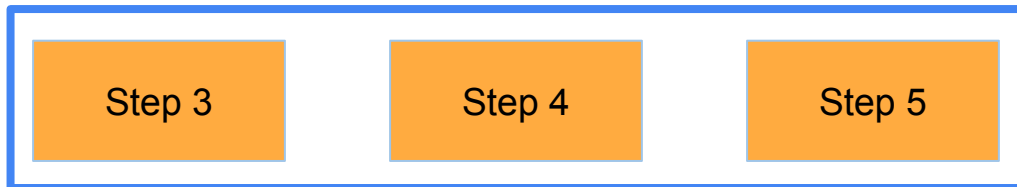
## Testing

4. Upload testing data
5. Make predictions for testing data

# Our focus

Step 1

Step 2



Step 3

Step 4

Step 5

- Wide Learning efficiently identifies important feature combinations at Step 2
- Weights are assigned to these important combinations in Step 3 and are used to make predictions in Step 5
- We focus on making improvements to Steps 3-5, by addressing areas of confusion from the survey responses and incorporating hypothesis-driven learning

# Improvement 1: Feature Selection in LASSO

# Logistic Lasso

- Logistic regression with L1 regularizer to assign weights to combinations from Step 2
- Lasso performs feature selection → some combinations are assigned weight 0
- Optimal weights are selected by solving the following problem:

$$\beta^* = \arg \min_{\beta} \mathcal{L}(\beta) := f(\hat{y}, y) + \rho \|\beta\|_1$$

$y_i \in \{0, 1\}$ , class label

$\hat{y}_i \in (0, 1)$ , prediction based on  $\beta$

# User Input

- In Wide Learning and the other models considered, important feature combinations are selected by the model
- We include user selection of important combinations in our Lasso model to increase trustworthiness based on this paper:

[Satoshi Hara and Takanori Maehara. Finding alternate features in lasso. arXiv preprint arXiv:1611.05940, 2016.](#)

# User Input

- Lasso selects important feature combinations by assigning them a nonzero weight
- Feature combinations not included in the model are assigned weight zero
- User can select a feature combination with a zero weight to include in the model
- The optimal nonzero weight for this combination is found
- A feature combination included in the original model is assigned a zero weight to keep the same number of nonzero weights

# Finding Alternate Weights

Given the Lasso optimal solution  $\beta^*$ , we seek for whether there are any alternate feature  $x_j$  with  $\beta_j^* = 0$  that can be replaced with a feature  $x_i$  selected by the Lasso (i.e.,  $\beta_i^* \neq 0$ ). We solve this problem by optimizing  $\beta_j$  in (1) while fixing as  $\beta_i = 0$  and  $\beta_k = \beta_k^*$  ( $k \neq i, j$ ). The optimization problem can be expressed as

$$\beta_j^{(i)} = \underset{\beta_j}{\operatorname{argmin}} f(z^{(i)} + X_j \beta_j, y) + \rho |\beta_j|, \quad (2)$$

where  $X_j$  denotes the  $j$ -th column of  $X$  and  $z^{(i)} = \sum_{k \neq i} X_k \beta_k^*$ . If  $\beta_j^{(i)} \neq 0$ , the feature  $x_j$  can be an alternative of  $x_i$ . We note that the problem (2) is a univariate optimization problem, and can be solved easily, e.g., by using the proximal gradient method [5].

# User Selection Example

- Red betas are included in the model (nonzero), blue betas are not included in the model (zero)
- This example shows replacing feature combinations 2 and 3 with feature combination 1

$$\beta^* = \begin{bmatrix} \beta_1^* = 0 \\ \beta_2^* \neq 0 \\ \beta_3^* \neq 0 \\ \beta_4^* = 0 \end{bmatrix}$$

$\mathcal{L}(\beta^*)$

$$\beta^{(2 \rightarrow 1)} = \begin{bmatrix} \beta_1^{(2 \rightarrow 1)} = \beta_1^{(2)} \\ \beta_2^{(2 \rightarrow 1)} = 0 \\ \beta_3^{(2 \rightarrow 1)} = \beta_3^* \\ \beta_4^{(2 \rightarrow 1)} = \beta_4^* \end{bmatrix}$$

$\mathcal{L}(\beta^{(2 \rightarrow 1)})$

$$\beta^{(3 \rightarrow 1)} = \begin{bmatrix} \beta_1^{(3 \rightarrow 1)} = \beta_1^{(3)} \\ \beta_2^{(3 \rightarrow 1)} = \beta_2^* \\ \beta_3^{(3 \rightarrow 1)} = 0 \\ \beta_4^{(3 \rightarrow 1)} = \beta_4^* \end{bmatrix}$$

$\mathcal{L}(\beta^{(3 \rightarrow 1)})$



# User Selection Example

- Example with four important feature combinations
- Optimal betas from Lasso

$$\beta^* = \begin{bmatrix} \beta_1^* = 0 \\ \beta_2^* \neq 0 \\ \beta_3^* \neq 0 \\ \beta_4^* = 0 \end{bmatrix}$$

# User Selection Example

- $\beta_2$  and  $\beta_3$  have nonzero weights, so they are included in the model

$$\beta^* = \begin{bmatrix} \beta_1^* = 0 \\ \beta_2^* \neq 0 \\ \beta_3^* \neq 0 \\ \beta_4^* = 0 \end{bmatrix}$$

# User Selection Example

- $\beta_1$  and  $\beta_4$  have weights of 0, so they are not included in the model

$$\beta^* = \begin{bmatrix} \beta_1^* = 0 \\ \beta_2^* \neq 0 \\ \beta_3^* \neq 0 \\ \beta_4^* = 0 \end{bmatrix}$$

# User Selection Example

- Say the user wants to make sure  $\beta_1$  is included in the model
- Need to find a new optimal nonzero  $\beta_1$

$$\beta^* = \begin{bmatrix} \beta_1^* = 0 \\ \beta_2^* \neq 0 \\ \beta_3^* \neq 0 \\ \beta_4^* = 0 \end{bmatrix}$$

# Option 1 - Replace $\beta_1$ with $\beta_2$

- Find new optimal value of  $\beta_1$

$$\beta^{(2 \rightarrow 1)} = \begin{bmatrix} \beta_1^{(2)} \\ 0 \\ \beta_3^* \\ \beta_4^* \end{bmatrix}$$

# Option 1 - Replace $\beta_1$ with $\beta_2$

- Find new optimal value of  $\beta_1$
- Set  $\beta_2$  to 0

$$\beta^{(2 \rightarrow 1)} = \begin{bmatrix} \beta_1^{(2)} \\ 0 \\ \beta_3^* \\ \beta_4^* \end{bmatrix}$$

# Option 1 - Replace $\beta_1$ with $\beta_2$

- Find new optimal value of  $\beta_1$
- Set  $\beta_2$  to 0
- Keep  $\beta_3$  and  $\beta_4$  the same

$$\beta^{(2 \rightarrow 1)} = \begin{bmatrix} \beta_1^{(2)} \\ 0 \\ \beta_3^* \\ \beta_4^* \end{bmatrix}$$

## Option 2 - Replace $\beta_1$ with $\beta_3$

- Find new optimal value of  $\beta_1$
- Set  $\beta_3$  to 0
- Keep  $\beta_2$  and  $\beta_4$  the same

$$\beta^{(3 \rightarrow 1)} = \begin{bmatrix} \beta_1^{(3)} \\ \beta_2^* \\ 0 \\ \beta_4^* \end{bmatrix}$$



# Choosing the Best New Solution

- Compare the objective values of the new solutions to the original  $\beta^*$

$$\beta^* = \arg \min_{\beta} \mathcal{L}(\beta) := f(\hat{y}, y) + \rho \|\beta\|_1$$

$y_i \in \{0, 1\}$ , class label

$\hat{y}_i \in (0, 1)$ , prediction based on  $\beta$

$$\mathcal{L}(\beta^*)$$

$$\mathcal{L}(\beta^{(2 \rightarrow 1)})$$

$$\mathcal{L}(\beta^{(3 \rightarrow 1)})$$

# **Improvement 2: More Result Options**

# Additional Models to Select

We can improve the convincingness of our model results by providing other models to compare, such as:

- Logistic regression
- Perceptron
- Decision Trees
- Random Forest
- Support Vector Machines (SVM)
- Gaussian Naive Bayes

# Reasons for selecting six models

- Hello Wide Learning targets binary classification problems.
- Six models are well-known methods in the field of classification.
- Six models can be executed using sklearn.

# Logistic Regression

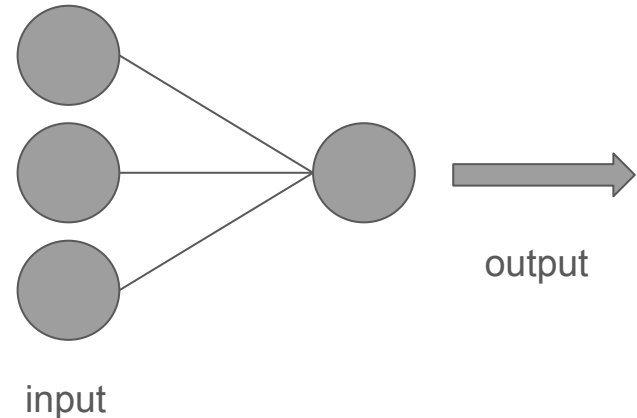
- A statistical model primarily used for binary classification
- it calculates using a linear prediction model
- The analysis is performed by passing the values through a sigmoid function

$$y = \sum_{i=1}^n w_i x_i + b \quad (i = 1, 2, \dots, n) : \text{linear prediction model}$$

$$\sigma(y) = \frac{1}{1 + \exp(-y)} \quad : \quad \text{sigmoid function}$$

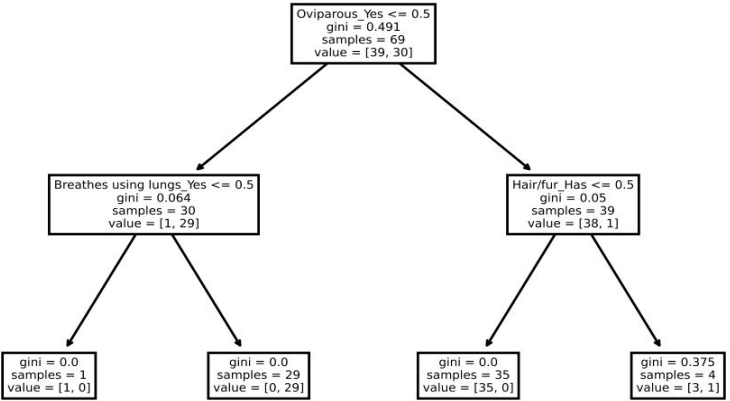
# Perceptron

- A simple linear classification algorithm
- calculates the weighted sum of input data features and classifies them into two categories



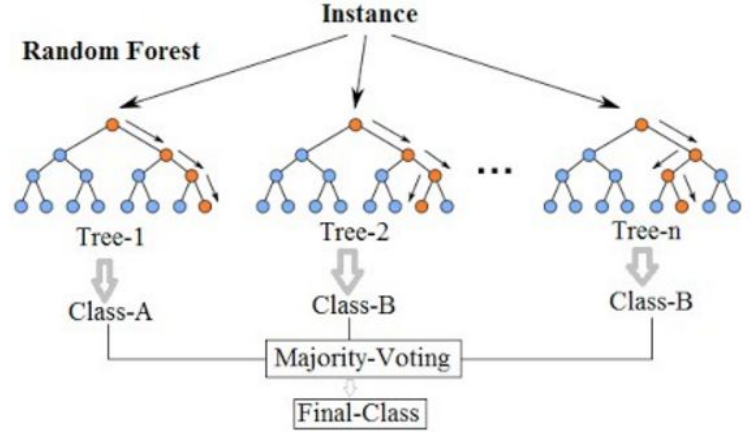
# Decision Tree

- An analysis that creates a tree diagram from data
- Classifies data based on “yes” or “no” information



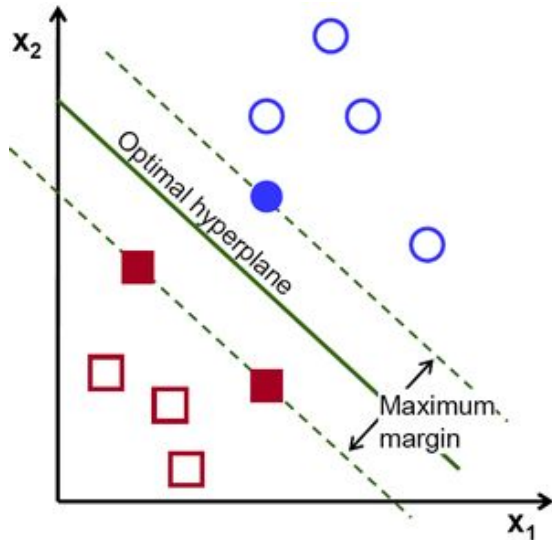
# Random Forest

- Combines the outputs of multiple decision trees to produce a single result
- Allows for highly accurate predictions by obtaining majority voting



# Support Vector Machine (SVM)

- Well-known algorithm in machine learning
- High generalizability with a small amount of supervised data



# Gaussian Naive Bayes

- Machine learning algorithm used for binary classification problems
- Assume that the characteristics of each class follow a Gaussian distribution
- Predicting new classes of data based on calculated probabilities by learning the mean and variance of each class from training data

# One of our approach

- Add “model selection”
  - we thought that this proposal would solve the “variety” problem required by Fujitsu.
- Furthermore, we added the complexity of the model.
  - Improving the interpretability of the models



# **Improvement 3: Explaining the Models and Calculating Complexity**

# Simple = Better? → Rademacher Complexity

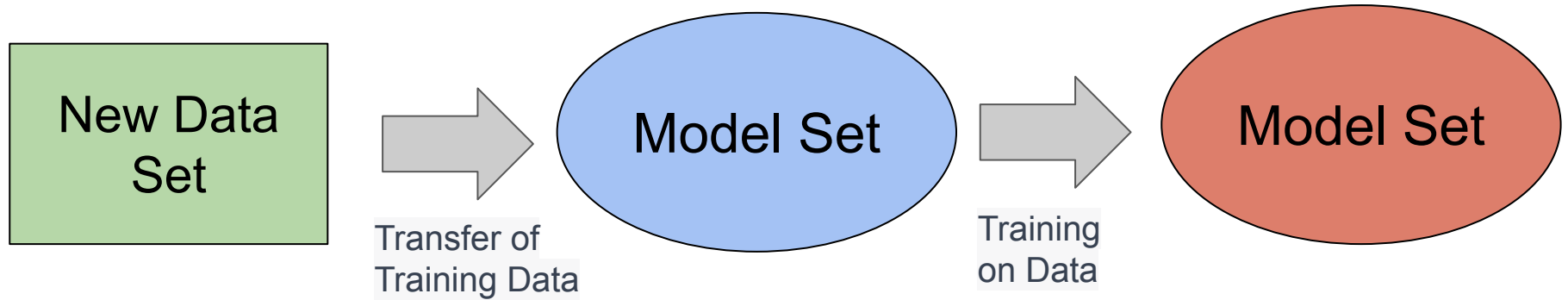
- Rademacher complexity is a measure that quantifies the complexity of a function set.
- Rademacher complexity can be measured for most predictive models such as Lasso, decision trees, and SVM.
- Our goal is to use Rademacher complexity to find models that are not complex. Simple = Better!

$$\hat{\mathfrak{R}}_S(\mathcal{G}) = \mathbb{E}_\sigma \left[ \sup_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^n \sigma_i g(x_i) \right] \quad S = \{x_1, \dots, x_n\} \in \mathcal{X}$$

	feature_1	feature_2	feature_3	feature_4
Name1	1	-1	1	-1
Name2	1	1	-1	1
Name3	1	-1	1	-1
Name4	-1	1	1	1

	feature_1	feature_2	feature_3	feature_4
Name1	1	-1	1	-1
Name2	1	1	-1	1
Name3	1	-1	1	-1
Name4	-1	1	1	1



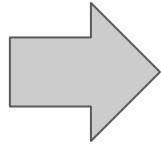


New dataset  
with **randomly**  
**re-labelled** labels  
and **original**  
**feature matrix**

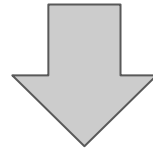
Set of Untrained  
Models

Make a Set of  
Trained Models

Original  
feature Matrix



Set of  
Trained  
Models

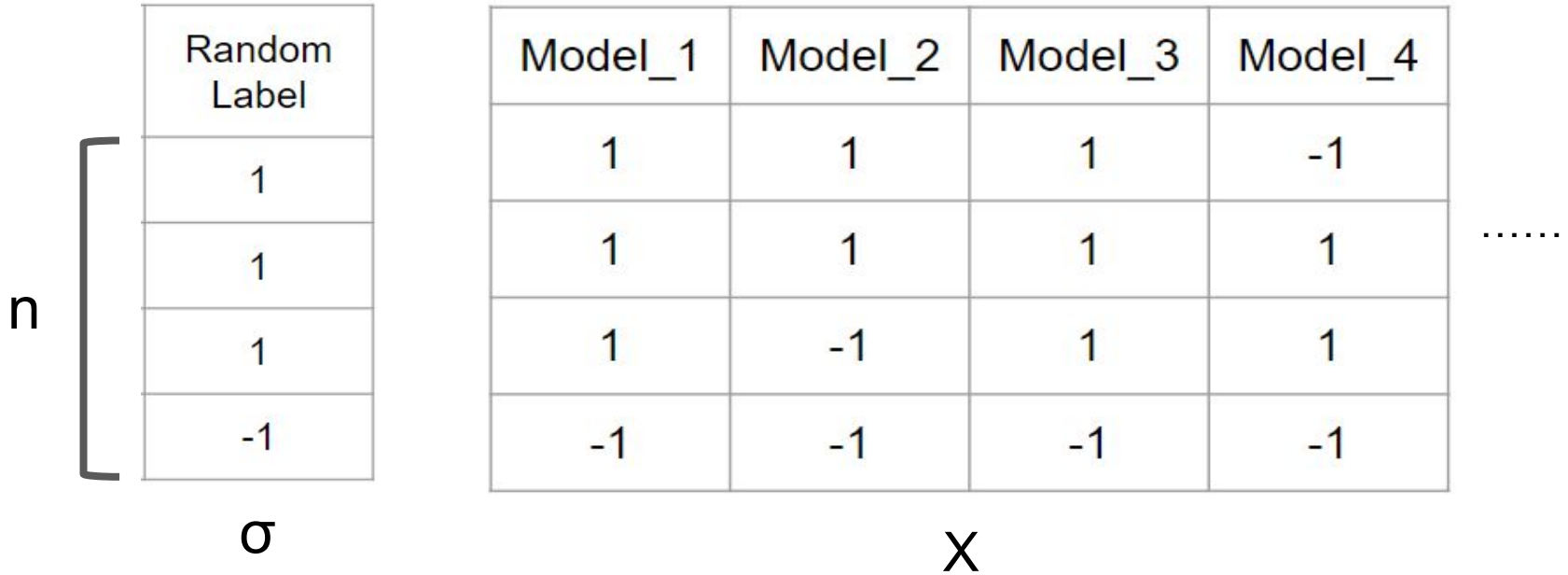


Predict random labels  
using the original feature  
matrix.

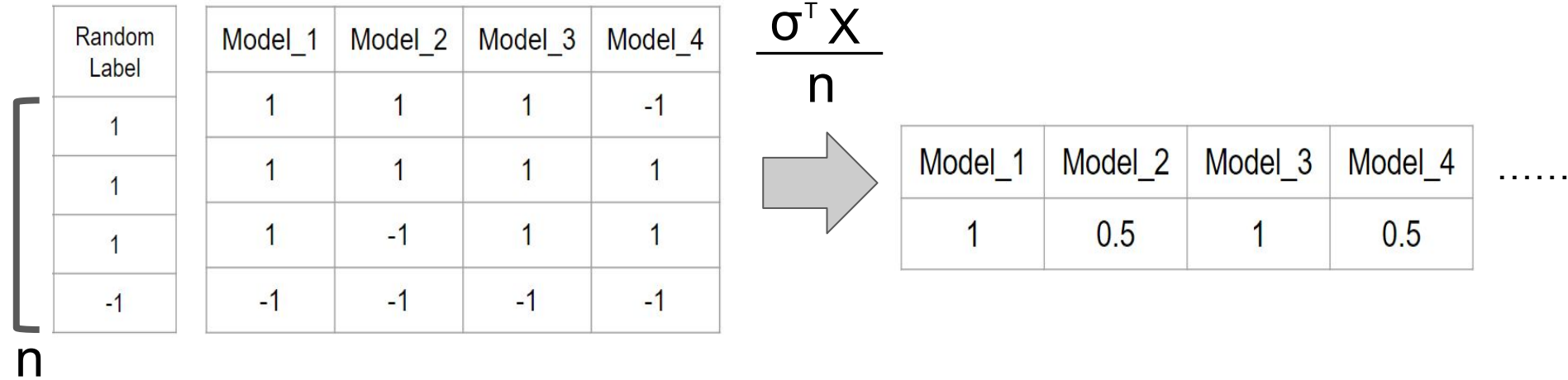
We obtained  
a matrix of  
predictions  
for each  
model

Model_1	Model_2	Model_3	Model_4
1	1	1	-1
1	1	1	1
1	-1	1	1
-1	-1	-1	-1

.....



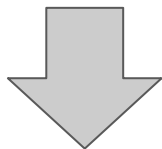
Let the random labels be the vector  $\sigma$ , the feature matrix of the model be the matrix  $X$ , and the number of rows be  $n$ .



Calculate the product of  $\sigma^T$  and  $X$  to determine the percentage of random labels that could be predicted.

	Model_1	Model_2	Model_3	Model_4
$\sigma^{(1)}$	1	0.5	1	0.5
$\sigma^{(2)}$	0.5	0.5	0.5	0
$\sigma^{(3)}$	1	1	1	0.5
$\sigma^{(4)}$	0.5	0	-0.5	1

⋮



average

Rademacher	0.72	0.65	0.32	0.44
------------	------	------	------	------

.....

Generate a random label  $\sigma$  many times and average a series of calculations.

It is called the empirical Rademacher complexity.

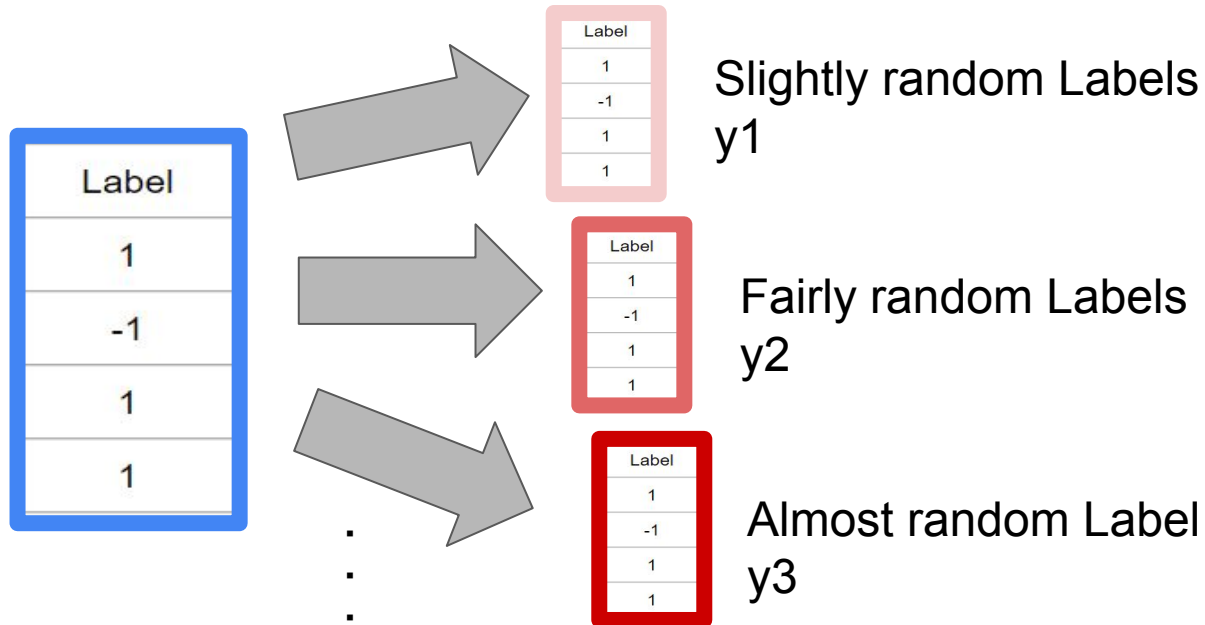
.....



- By measuring the ability to learn to random labels, the potential complexity of the model could have been measured.
- However, we felt the need to evaluate the model in more detail and prepared a new metric.

Our metric shows how well the model predicts as the randomness of the labels increases.

This allows us to examine how much more random the model becomes less predictable.



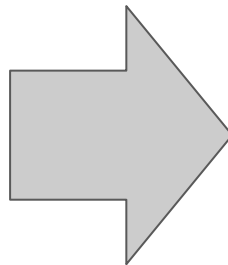
Random Label	Model_1	Model_2	Model_3	Model_4
1	1	1	1	-1
1	1	1	1	1
1	1	-1	1	1
-1	-1	-1	-1	-1

$$\frac{y_1^T X_1}{n}$$

Model_1	Model_2	Model_3	Model_4
1	0.5	1	0.5

Random Label	Model_1	Model_2	Model_3	Model_4
1	1	1	1	-1
1	1	1	1	1
1	1	-1	1	1
-1	-1	-1	-1	-1

$$\frac{y_2^T X_2}{n}$$



Model_1	Model_2	Model_3	Model_4
1	0.5	1	0.5

Random Label	Model_1	Model_2	Model_3	Model_4
1	1	1	1	-1
1	1	1	1	1
1	1	-1	1	1
-1	-1	-1	-1	-1

$$\frac{y_3^T X_3}{n}$$

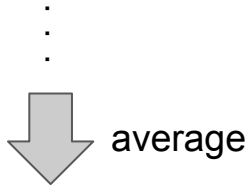
Model_1	Model_2	Model_3	Model_4
1	0.5	1	0.5

For each label, the same process is used as when the Rademacher complexity is calculated.

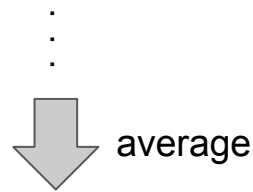
	Model_1	Model_2	Model_3	Model_4
$y1^{(1)}$	1	0.5	1	0.5
$y1^{(2)}$	0.5	0.5	0.5	0
$y1^{(3)}$	1	1	1	0.5
$y1^{(4)}$	0.5	0	-0.5	1

	Model_1	Model_2	Model_3	Model_4
$y2^{(1)}$	1	0.5	1	0.5
$y2^{(2)}$	0.5	0.5	0.5	0
$y2^{(3)}$	1	1	1	0.5
$y2^{(4)}$	0.5	0	-0.5	1

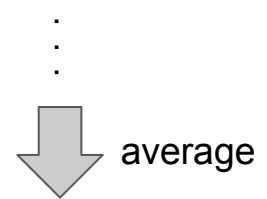
	Model_1	Model_2	Model_3	Model_4
$y3^{(1)}$	1	0.5	1	0.5
$y3^{(2)}$	0.5	0.5	0.5	0
$y3^{(3)}$	1	1	1	0.5
$y3^{(4)}$	0.5	0	-0.5	1



	0.72	0.65	0.32	0.44
--	------	------	------	------

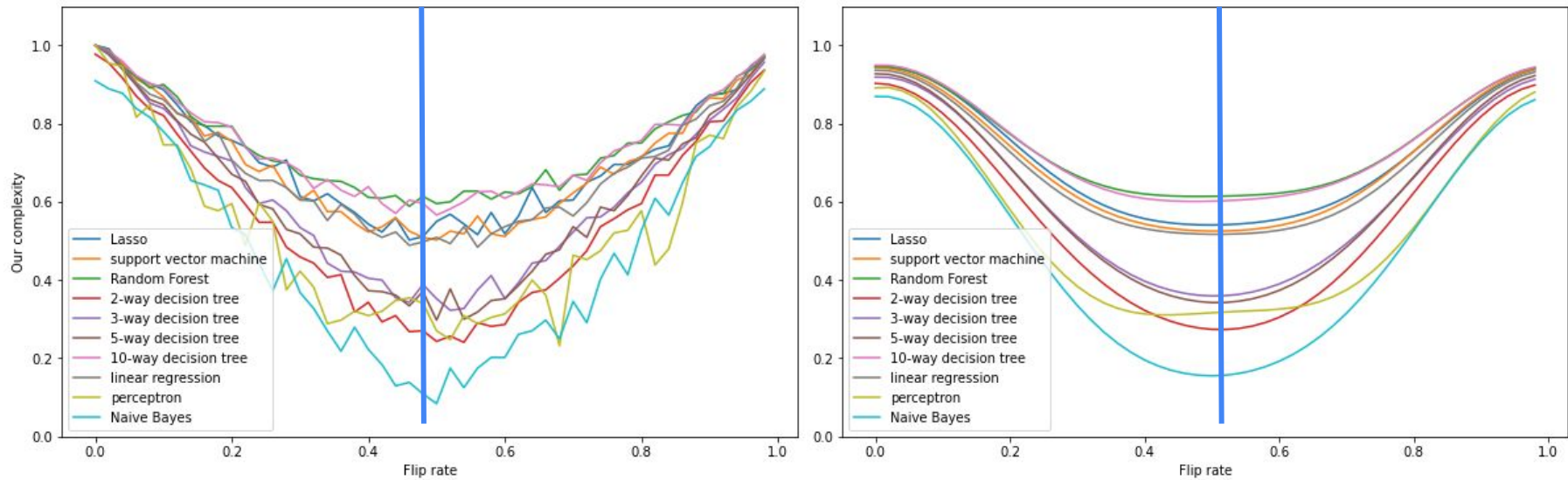


	0.72	0.65	0.32	0.44
--	------	------	------	------



	0.72	0.65	0.32	0.44
--	------	------	------	------

The process of determining the label for each certain clutter is repeated many times and the average is taken.



The axis is blue in the middle, the labels are completely random.

This graph shows how much the model loses predictive accuracy for random labels as the randomness of the labels increases.

# Complexity Summary

- The potential complexity of the model was evaluated by looking at the model's follow-up to random labels by Rademacher complexity.
- Our new method is a more detailed analysis of Rademacher complexity, which will lead to the discovery of a simpler model by referring to the graph.

# **“Hello, New Learning!”**

## **Current User Interface**

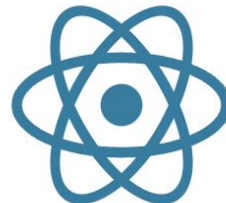
# New Interface with Our Algorithms



**Flask**

**Back-End**

Python Flask



**React**

**Front-End**

React JS



# Demo of New User Interface

Example Sheet 1002

Step 1: Upload Training Data    Step 2: Use Individual Contributions    Step 3: Select Weights    Step 4: Set of Possible Models    Step 5: Select Best One    Step 6: Weights with Final Selected Features    Step 7: Upload Training Data    Step 8: The Solution from the Training Model

Name	Gender	Weight	Empire	Star	Species	Estimated	True
Anakin	Male	Dark side force	No	Dark side	No	No	Yes
Ahsoka	Female	Dark side force	No	Dark side	No	No	Yes
Obi	Male	Dark side force	No	Dark side	No	No	Yes
Yoda	Female	Dark side force	No	Dark side	No	No	Yes
Truffula	Female	Dark side force	No	Dark side	No	No	Yes
Qui	Male	Dark side force	No	Dark side	No	No	Yes

The Training Data is uploaded to the web site and will be used to train various potential models, which will give weights to the different feature contributions. It will go to back to console and will do Step 2 above to proceed.

# Demo of New User Interface

Feature Selection

Step 1: Upload Training Data    Step 2: Set the Importance Constraints    Step 3: Learn the Weights    Step 4: Set of Feasible Models    Step 5: Select One Best Feature    Step 6: Weights with One Selected Feature    Step 7: Update Training Data    Step 8: Iteration from the Feasible Models

While learning generates all possible combinations of variables, the combination is iteratively set to the most important combination of features to determine whether or not it is useful to the model. The importance of a combination is measured by the length of the combination and the number of positive and negative responses for the models in the learning dataset that have that combination of features. The table below will only show those combinations with a maximum length of three features, regardless of how many combinations that are longer than three may exist.

Combination of Features	Combination Length	Number of POS	Number of NEG
"Does not have Wings" and "Has Beak" and "Lays Egg"	3	40	5
"Does not have Wings" and "Has Spine" and "Has Beak" and "Lays Egg"	4	40	5
"Has Tail" and "Has Beak" and "Lays Egg"	3	38	6
"Has Wings"	1	38	2
"Has Tail"	1	38	4
"Does not Fly" and "Has Beak" and "Lays Egg"	3	38	10
"Does not have Wings" and "Does not have Feet" and "Lays Egg"	3	33	20
"Does not have Wings" and "Has Spine" and "Does not have Feet"	3	31	9
"Has Tail" and "Does not have Feet"	2	30	9

# Demo of New User Interface

Learn Weights

Step 1: Define Training Data Step 2: On-Line Incremental Classification Step 3: Learn Weights Step 4: Set of Feature Models Step 5: Select One of the Feature Models Step 6: Weights with the Selected Feature Model Step 7: Online Training Data Step 8: The Effect of the Weights Models

The table shows the weights from feature (Step 4) to Decision Tree's model that decompose data into leaf and output prediction. It is a very easy-to-understand model because it is not produced by repeatedly adding 'yes' and 'no' questions.

Feature Combination	Weight
"Yes Drink" and "Yes Reservations" (yes)	0.99
"No Reservations"	0.99
"Superstar" and "Yes Eat"	0.99

# Demo of New User Interface

...navigation...

For the chart with more information about complexity for the models, click the button below.

Model complexity chart

Model Name	accuracy	complexity
Linear	1.00	0.10
Support Vector Machine	1.00	0.10
Random Forest	1.00	0.50
Decision Tree (depth = 2)	1.00	0.30
Decision Tree (depth = 3)	1.00	0.40
Decision Tree (depth = 4)	1.00	0.20
Decision Tree (depth = 10)	1.00	0.10
Logistic Regression	0.99	0.20
Perceptron	0.98	0.20
Decision Naive Bayes	0.98	0.10

Linear Support Vector Machine Random Forest

Decision Tree (depth = 2) Decision Tree (depth = 3) Decision Tree (depth = 4)

Decision Tree (depth = 10) Logistic Regression Perceptron

# Demo of New User Interface

Exam (Exam 2011)

Step 1: Upload Training Data    Step 2: On-line Inspection    Step 3: Extract Weights    **Step 4: Set of Feature Models**    Step 5: Select One Class    Step 6: Weights with This Selected Feature    Step 7: Upload Training Data    Step 8: Weights from All Weights & Models

The table derives the weights from Exams. The Exam model assigns weights to each of the feature-combinations provided by With Learning. Exams automatically performs feature selection to generate some of the feature combinations a weight of 0, which we filter them from the final model.

[Click here to enter your text. \(Maximum 2000 characters\)](#)

Feature Combination	Weight
"Class not new Wings" and "The Brothers wing long" and "Lap of 2"	1.74
"Class not new Wings" and "The Spine" and "The Brothers wing long"	1.68
"The Spine"	1.24
"The Spine"	1.81
"Class not new Wings" and "The Spine" and "The Brothers wing long"	6.75
"The Spine and the spine is not"	1.18
"Class not new Wings"	1.77
...	...

A higher numerical value indicates a more important feature

# Demo of New User Interface

Examine (Exam).png

Step 1: Validated Training Data    Step 2: Data Inspection / Descriptions    Step 3: Labeled Weights    Step 4: Set of Feasible Models    **Step 5: Select Data Used Features**    Step 6: Weights with Feat. Selected Features    Step 7: Labeled Training Data    Step 8: Model Loss from the Feasible Models

The following is a list of some of the feature combinations from step 4 that were assigned a weight of 0 by the solver and were therefore not included in the final model. By selecting one of these feature combinations, it will be assigned a non-zero weight. One of the feature combinations with a nonzero weight from the original solver model is 4 (it is taken out of the model by assigning it a weight of 0).

["Age", "Sex", "Smoker", "NoInsulin"]	["Age", "Sex", "Smoker", "NoInsulin"]	["Age", "Sex", "Smoker", "NoInsulin"]
["Age", "Sex", "Smoker", "NoInsulin", "Diabetes"]	["Age", "Sex", "Smoker", "NoInsulin", "Diabetes"]	["Age", "Sex", "Smoker", "NoInsulin", "Diabetes"]
["Age", "Sex", "Smoker", "NoInsulin", "Diabetes", "BMI"]	["Age", "Sex", "Smoker", "NoInsulin", "Diabetes", "BMI"]	["Age", "Sex", "Smoker", "NoInsulin", "Diabetes", "BMI"]
["Age", "Sex", "Smoker", "NoInsulin", "Diabetes", "BMI", "HbA1c"]	["Age", "Sex", "Smoker", "NoInsulin", "Diabetes", "BMI", "HbA1c"]	["Age", "Sex", "Smoker", "NoInsulin", "Diabetes", "BMI", "HbA1c"]

# Demo of New User Interface

The screenshot displays a web application interface with a progress bar at the top and a data table below. The progress bar consists of eight steps, with the seventh step, 'Step 7: Upload Training Data', highlighted in a teal color. A red vertical bar is positioned above the progress bar, indicating the current step. The data table has eight columns: Name, Race Sex, Priage, O-Spectrum, Film, Age, Eye color, and Gender. The table contains five rows of data. To the right of the table, there is a text block that reads: 'The string data you uploaded is shown here and will be used to generate prediction results that are detailed on this screen in Step 4. Scroll to check the columns and data in Step 7 above to proceed.'

Name	Race Sex	Priage	O-Spectrum	Film	Age	Eye color	Gender
Violet	Black MA	Red	No	Dark	No	No	Black MA
Charles	Black MA	Dark MA	No	Dark MA	Yes	No	Red
Phyllis	Black MA	Red	No	Dark MA	No	No	Dark MA
Phyllis	Red	Dark MA	No	Dark MA	Yes	No	Dark MA
Walter	Dark MA	Dark MA	No	Dark MA	No	No	Dark MA

The string data you uploaded is shown here and will be used to generate prediction results that are detailed on this screen in Step 4. Scroll to check the columns and data in Step 7 above to proceed.

# Demo of New User Interface

The screenshot displays a software interface for a machine learning workflow. At the top, a navigation bar contains eight steps: Step 1: Upload Training Data, Step 2: Use Improved Coordinates, Step 3: Load Model, Step 4: Set of Features, Step 5: Select Data Class Feature, Step 6: Apply with Test Data/Features, Step 7: Upload Training Data, and Step 8: Use/Access from the Model & Model. A red vertical bar highlights Step 6. Below the navigation bar, a text prompt reads: "Show/Hide all the prediction scores generated by the Support Vector Machine model". A table is displayed with the following data:

Area	Prediction	Scores
Yellow	0	0.04
Orange	1	0.04
Green	0	0.04
Purple	1	0.04
Blue	0	0.04



# Conclusion

- Our version of Wide Learning considers many different models while making predictions
- The user can decide which model they prefer based on accuracy and complexity
- The user also has the option to select important feature combinations for the Lasso model
- The new user interface incorporates additional explanations to address feedback from the pilot survey
- A new survey has been sent out to evaluate the new workflow and interface we developed

**Thank You!**