Explainable Artificial Intelligence

Topics covered in PPTs:

1. Interpretability on Regression, Generalized additive models, Decision Trees
2. Neural Networks Interpretability – Part 1
3. Neural Networks Interpretability – Part 2
4. LIME, SHAP
5. PDP, Surrogate models

Referred from the Interpretable ML Book

VNY

1.Interpretability on Regression, Generalized Additive Models, and Decision Trees

What is a Linear Regression

A linear regression model predicts the target as a weighted sum of the feature input

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \epsilon$$

Where:
- Prediction is a weighted sum of features
- β_j = feature weights/coefficients
- β_0 = intercept (not multiplied by features)
- ϵ = error term

To find the optimal weights, we need to find coefficients that minimize squared differences between actual and predicted outcomes

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta_0,\ldots,\beta_p} \sum_{i=1}^{n} \left(y^{(i)} - \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_j^{(i)} \right) \right)^2$$

This formula helps find the line that's closest to all points by minimizing the total distance (squared) from each point to the line. The squared term part penalizes bigger errors more heavily.

Interpreting Feature Weights

Numerical Features:

- Increasing feature x_j by one unit changes prediction by β_j units (holding all else constant)
- Binary/Categorical Features:
- Changing from reference category to another category changes prediction by β_j) all else fixed)

Intercept:

Predicted outcome when all numerical features = 0 and categorical features at reference

Evaluating Model Quality (R-squared)

$$R^2 = 1 - \frac{SSE}{SST}$$

R-squared tells you what percentage of variation in outcomes your model explains. An R^2 of 0.8 means your model explains 80% of why outcomes vary.

Where

$$SSE = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2$$

Represents Error variance

$$SSE = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2$$

Represents Total variance

Feature Importance

The importance of a feature in a linear regression model can be measured by the absolute value of its t-statistic. The t-statistic is the estimated weight scaled with its standard error.

$$t_{\hat{\beta}_j} = \frac{\hat{\beta}_j}{SE(\hat{\beta}_j)}$$

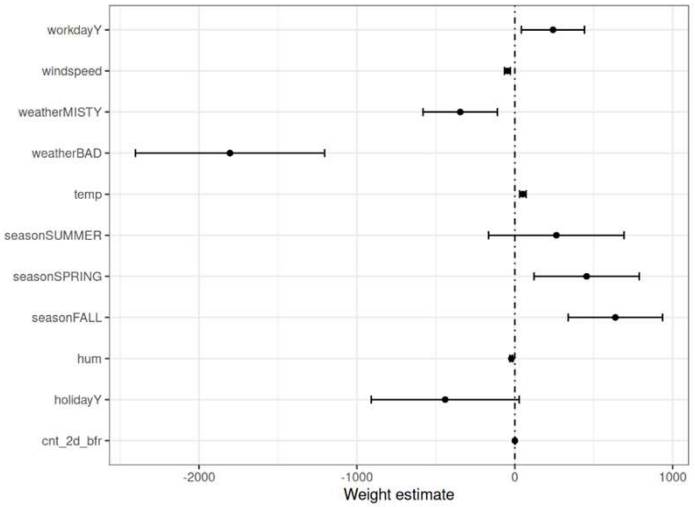
A large t-value means the feature's effect is both substantial and consistent

Visualization Methods

Weight Plots

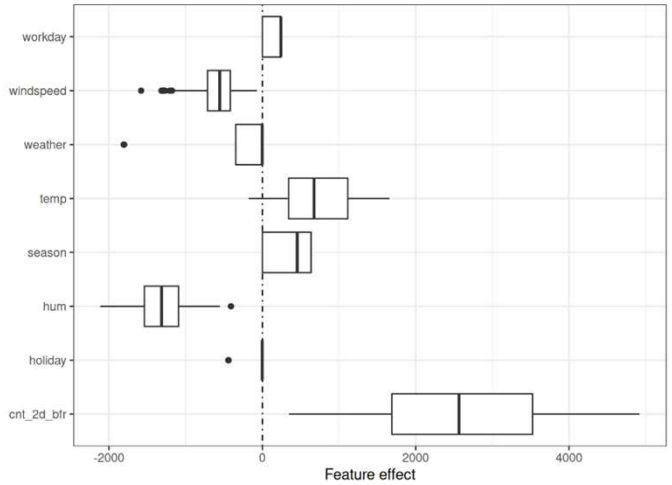
Shows coefficients with confidence intervals

Weight plots show you each feature's importance with error bars indicating uncertainty.



Effect plots are more practical—they show the actual impact in your dataset by multiplying the weight by real feature values.

Reveals actual contribution in data (accounts for feature distribution)



Lasso Optimization in linear regression models

- Lasso (least absolute shrinkage and selection operator)
- Regular linear regression uses all features even if some are useless.
- Lasso adds a penalty for having too many non-zero weights, forcing the model to be selective.
- Lasso automatically drops unimportant features by setting their weights to exactly zero, giving a simpler, more interpretable model.
- The λ parameter controls how aggressive this pruning is.

$$\min_{\beta}\left(\frac{1}{n}\sum_{i=1}^{n}(y^{(i)}-\mathbf{x}^{(i)T}\beta)^2+\lambda||\beta||_1\right)$$

Where
n = number of data points.
$y(i)$ = true output for the i-th data point.
$x(i)$ = vector of features for the i-th data point.
β = vector of model parameters (coefficients) we're trying to learn.
$\|\beta\|1$= the L1-norm of the coefficients (sum of absolute values).
λ = regularization parameter (controls how much penalty we put on the size of coefficients).

Strengths of linear regression

- Transparent: Weighted sum makes prediction process clear
- Widely accepted: Extensive expertise and implementations
- Optimal solutions: Mathematical guarantees for finding best weights
- Statistical rigor: Confidence intervals and solid theory
-

Weakness of linear regression

- Linear only: Cannot capture nonlinearities without manual feature engineering
- Lower predictive performance: Oversimplifies complex reality
- Unintuitive weights: Interpretation depends on all other features
- Multicollinearity issues: Correlated features destabilize estimates

When Linear Regressions should be used:

- Linearity: Relationships must be linear combinations
- Normality: Target follows normal distribution given features
- Homoscedasticity: Constant error variance across feature space
- Independence: Each instance independent of others
- No multicollinearity: Features should not be strongly correlated

What is a logistic regression ?

Definition: Logistic regression is a statistical method used to model a binary outcome (target variable takes values 0 or 1) based on one or more predictor variables (features).

Use cases: Common for classification problems such as disease diagnosis, spam detection, and customer churn.

- Linear regression is not suited for classification problems because its predictions are not limited to probabilities between 0 and 1, and there's no natural threshold for classifying points.

- Outputs probability estimates for each class, not just discrete labels.

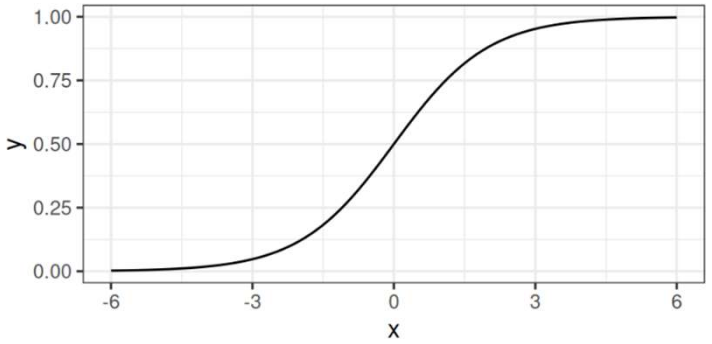
- Logistic regression models the probability of a binary outcome (class 1, e.g., success/failure) using a logistic (sigmoid) function.
- The output is always between 0 and 1 and can be interpreted as a probability.
- It is a type of generalized linear model (GLM).

$$\mathbb{P}(Y^{(i)} = 1) = \text{logistic}(\mathbf{x}^{(i)T}\boldsymbol{\beta}) = \frac{1}{1 + \exp(-(\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}))}$$

The Logistic (Sigmoid) function

$$\text{logistic}(\mathbf{z}) = \frac{1}{1 + \exp(-\mathbf{z})}$$

z is the linear combination of input features.
Maps any real number to a number between 0 and 1.



Log Odds and model linearity

In logistic regression, unlike linear regression, the response variable (probability) is not modeled directly as a linear function of the features. Instead, we model the *log-odds* (logit) as a linear function of features.

$$\ln\left(\frac{\mathbb{P}(Y=1)}{1-\mathbb{P}(Y=1)}\right) = \ln\left(\frac{\mathbb{P}(Y=1)}{\mathbb{P}(Y=0)}\right) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

The term inside the is called the odds (probability of event divided by probability of non-event), and the ln of odds is the log-odds.
Hence, logistic regression is a linear model for the log-odds.

Transition to probabilities: To get the probability itself, you apply the inverse of the logit (the logistic function):

$$\mathbb{P}(Y^{(i)}=1) = \text{logistic}(\mathbf{x}^{(i)T}\boldsymbol{\beta}) = \frac{1}{1+\exp(-(\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}))}$$

Increasing a feature by one unit increases the log-odds by β_j.
On the odds scale, this means odds are multiplied by exp(β_j)

Odds Ratios and Coefficient Interpretation

Coefficient interpretation (mathematical):

For feature x_j:

- Increase x_j by one unit: Odds change by a factor of
- Log-odds increase additively by β_j

Interpretation by feature type:

- Numerical feature: Increasing feature by one increases odds by
- Binary categorical feature: Switching from reference to category changes odds by
- Categorical >2 levels: Use one-hot encoding; each category (not reference) has its own odds ratio
- Intercept (β_0 :(Odds when all features zero/reference; often not interpretable directly

Odds Ratio Examples and Intuition

Odds ratio > 1: Feature increases likelihood of outcome
Odds ratio < 1: Feature decreases likelihood of outcome

- Odds ratio is the multiplier: 2 means "twice as likely," 0.5 means "half as likely."
- The actual effect on probability depends on the baseline odds—so interpreting odds ratios is easiest for direction and rough strength, rather than focusing on probability change.

- Odds ratio summarizes the change in relative likelihood—easy to communicate direction and strength of feature influence.

- Binary categorical features:
  - Reference level is usually the one coded as 0.
  - Increase to category value (usually 1) multiplies odds by odds ratio .
- More than two categories:
  - Typically represented by one-hot encoding.
  - Choose a reference category; each other category's coefficient shows odds ratio relative to the reference.
- Intercept:
  - Means odds when all continuous variables are 0 and categorical at reference (often not a realistic scenario).
  - Intercept is rarely interpretable in real applications, except when features are centered or standardized.

Logistic Regression Strengths

- Probability outputs: Logistic regression provides probabilities, not just yes/no labels.
- Interpretable: Coefficients (as odds ratios) clearly show feature impact, direction, and relative scale.
- Extensible: Can handle multiclass via extensions; widespread support in statistical software.

Limitations

- Nonlinear effects: Model only captures linear effects in log-odds. Interactions or nonlinearity must be engineered explicitly.
- Multiplicative interpretation: Effects are not additive for probability, but multiplicative for odds. This makes direct probability changes less obvious.
- Complete separation: If a feature perfectly predicts the outcome, coefficient estimation fails (infinite weights); use regularization to address this.

Problems of Pure Linear Models

Typical real-world issues:

- Non-Gaussian outcomes: Can't handle outcomes like counts, categories, or times-to-event.
- Feature interactions: The effect of one feature changes depending on another.
- Nonlinear relationships: "Effect per unit" isn't always constant—sometimes the curve bends.

Generalized Linear Models (GLMs)

Keep weighted sum of features, but:

Allow non-Gaussian distributions for the outcome.

Link linear predictor and outcome via a flexible function.

$$g(\mathbb{E}[Y|\mathbf{x}]) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p = \mathbf{x}^T \boldsymbol{\beta}$$

g = link function: e.g., log, logit, etc.

GLMs let you swap out the normal distribution and allow curves between prediction and outcome—fitting more types of problems.

These interactions multiply features or create combinations (extra columns) so linear models can "see" their effect together.

Generalized Additive Models

$$g[\mathbb{E}(Y|X=\mathbf{x})] = \beta_0 + f_1(x_1) + f_2(x_2) + \ldots + f_p(x_p)$$

GAMs let each feature “draw its own curve,” instead of forcing straight lines. The model adds all these curves together to make its prediction.

- Replace each linear $\beta_j x_j$ with a smooth function $f_j(x_j)$
- Still additive (no hidden interactions unless added explicitly)
- Functions are often fitted with splines for flexibility

- Each $f_j(x_j)$ is learned from data; commonly via splines (piecewise polynomials stuck smoothly together).
- Model’s design matrix is expanded: each smooth function is represented by several columns (basis functions).
- GAMs can include both linear and nonlinear terms as needed.

Interpreting GAMs

- The contribution of each feature is visualized as its own curve.
- Effect of a feature is the difference from the mean prediction at each point on its curve.
- Curves are often centered at zero (the overall mean prediction).
- For a given value x_j ,calculate $f_j(x_j)$, This gives the *numeric contribution* of feature x_j at that value.
- Plotting $f_j(x_j)$ across the range of x_j shows how its effect changes, The vertical position (height) at each point = the effect at that feature value.

Strengths of GAMs

- Flexible: Capture complex nonlinear patterns but retain additivity (still somewhat interpretable).
- Visualizable: Individual feature effects can be plotted for insight.
- Extensible: Can include interactions, other distributions (via GLMs), etc.

Limitations of GAMs

- Less interpretable than linear models: Instead of one simple weight, interpretation requires examining each feature's curve.
- Assumptions still matter: Overfitting is possible with too much flexibility; smoothness must be controlled.
- Not a cure-all: Some complex interactions or high-dimensional data are still better handled by tree ensembles or other machine learning models.

Decision Trees

- Tree-based models split data into subsets based on feature values, creating a tree structure.
- Leaf nodes (terminal nodes): Each instance falls into exactly one leaf, which provides the prediction.
- Decision trees can handle nonlinear relationships and interactions between features.

$$\hat{f}(\mathbf{x}) = \sum_{m=1}^{M} c_m I\{\mathbf{x} \in R_m\}$$

Each instance falls into exactly one leaf node (= subset R_m). $I_{\{\mathbf{x} \in R_m\}}$ is the indicator function that returns 1 if $\mathbf{x}$ is in the subset R_m and 0 otherwise. If an instance falls into a leaf node R_l, the predicted outcome is c_l, where c_l is the average of all training instances in leaf node R_l.

Working of decision trees

- Starts with all data in the root node.
- Splits dataset using a feature and cutoff point that maximize difference with respect to the target.
  - For regression: minimizes outcome variance.
  - For classification: minimizes Gini index (impurity).
- Recursively splits new subsets (nodes) until a stopping rule is met (e.g., minimum instances per leaf, maximum depth).

Splitting Criteria – Variance & Gini Index

Regression:

- Choose split that minimizes the variance of the outcome in the resulting nodes.
- Variance measures how spread out the values are within a node. Lower is better.

Classification:

- Choose split that minimizes the Gini index.
- Gini index quantifies how "mixed" the classes are. Zero = perfect purity (all same class).
- Simple explanation:
  Splits should create groups with target values as close together (regression), or class labels as pure as possible (classification).

Interpretation of Decision trees

Start at root node, follow edges determined by feature cut-offs (“AND” statements), until reaching a leaf.
All decisions are ANDed: If feature x_j is [smaller/bigger] than threshold c AND … then predict the mean or most common outcome in that leaf.

Feature Importance in Decision Trees

- Quantitative measure: For each feature, sum how much each split (using that feature) reduces variance/Gini index.
- Importance for all features is scaled to 100 (as percentages).
- Reflects share of influence of each feature in the tree.

Numeric Decomposition of Prediction Paths

Prediction for instance:

$$\hat{f}(\mathbf{x}) = \bar{y} + \sum_{d=1}^{D} \text{split.contrib}(d, \mathbf{x}) = \bar{y} + \sum_{j=1}^{p} \text{feat.contrib}(j, \mathbf{x})$$

The model starts at the average (y), then adds up how each question (split) shifts the prediction, showing what made your answer different.

Interpretability – Contrastive and Selective Explanations

- Contrastive: Compare prediction with “what if” scenarios—see what outcome would change if feature split boundaries were crossed.
- Selective: Shallow trees (few splits) require only a handful of features for each prediction, making explanations simple.
- Truthful: Simplicity and accuracy depend on model depth and data.

Strengths of Decision Trees

- Captures interactions and nonlinear relationships naturally.
- Groups data into clusters that are easy to understand.
- Natural visualization: Nodes and edges are intuitively explained.
- Works with both numerical and categorical features—no need for feature scaling or transformation.
- Invariant to monotonic transformations: Changing units or scaling doesn’t affect the tree structure.

Limitations of Decision Trees

- Approximate linear relationships with step functions—can be inefficient.
- Lacks smoothness: Small changes in input can lead to big jumps in prediction (not always desirable).
- Instability: Small data changes can produce very different trees.
- Decreased interpretability with depth: Deep trees are harder to follow.