Production: Feature Engineering

```
$ echo "Data Science Institute"
```

Agenda

4.1 Feature Engineering

- Common Operations
- Data Leakage
- Feature Importance
- Feature Generalization

4.2 Feature Engineering

- Transformation Pipelines
- Encapsulation and parametrization

Slides, Notebooks, and Code

- These notes are based on Chapter 5 of *Designing Machine Learning Systems*, by Chip Huyen.

Notebooks

- `./notebooks/production_4_transforms.ipynb`

Code

- `./05-src/credit_experiment_nb.py`
- `./05-src/credit_preproc_ingredient.py`

Our Reference Architecture

The Flock Reference Architecture

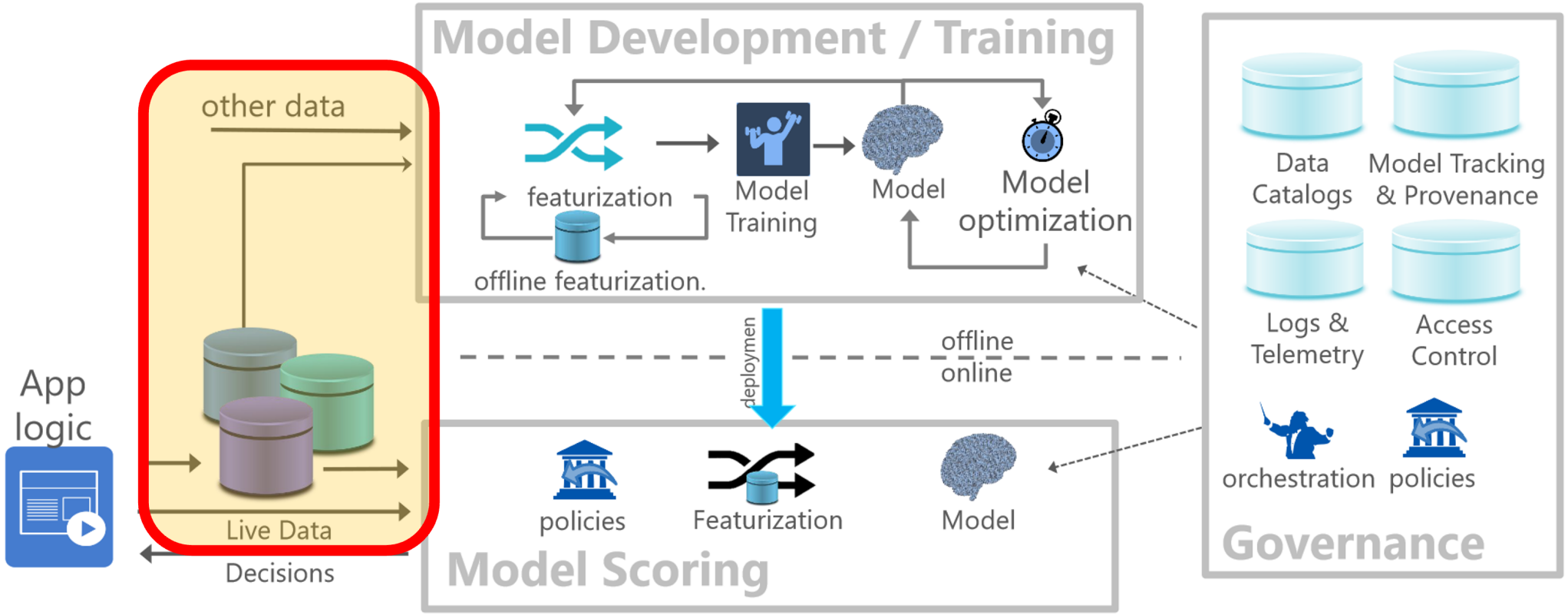


Figure 1: Flock reference architecture for a canonical data science lifecycle.

Learned Features Versus Engineered Features

- The promise of deep learning was that we no longer had to engineer features (feature or representation learning).
- Why do we need to engineer features?
- Some features can and will be automatically learned for certain use cases (vision, NLP, etc.)
- However, the majority of ML applications are not deep learning.

What is Feature Engineering?

- Feature engineering is the process of choosing what information to use and how to extract this information into a format usable by ML models.
- The purpose of feature engineering is to *transform and represent features so that their information content is best exposed* to the learning algorithm.

What is Feature Engineering?

Feature engineering can include:

- A transformation of a feature: standardization, scale, center, log, etc.
- An equivalent re-representation of a feature: dummy variables, one-hot-encoding, binning, etc.
- An interaction of two or more features such as a product or ratio: for example, calculate the ratio of a loan to the value of the collateral (or its inverse), as a new feature for default prediction.
- A functional relationship among features: Principal Components Analysis, LDA, etc. This may also include methods for imputing missing values.

Common Feature Engineering Operations

Handling Missing Values

- Missing values are a common occurrence in production data.
- Missing values can be of three types:
    - Missing Not At Random (MNAR): a value is missing because of the true value itself.
    - Missing At Random (MAR): a value is missing not due to the value itself but to another observed variable.
    - Missing Completely At Random (MCAR): there is no pattern in missing values.

Handling Missing Values: Deletion

- The simplest way to remove missing values is deletion.
- *Column deletion*
  - Remove variables with excessive missing values, but be cautious of potentially losing valuable information and compromising model performance.
- *Row deletion*
  - Remove samples with missing values, but only effective for MCAR with few missing values.
  - Drawbacks: Doesn't work for MNAR data and can create biases by removing rows.

Imputation

- Impute missing values using default values: missing strings, filled with "".
- Use a statistic like mean, median, or mode: fill the missing temperature with the mean temperature for the time of day within a specific window.
- Domain specific: if prices are liquid, use the last available price.

Imputation

- Model-based: if two variables are correlated and one of them has missing values, model the relationship and use model results for imputation.
- Flag imputed missing values.
- Avoid filling missing values with possible (fixed) values. Example: missing number of children should not be filled with 0, a possible value.

Scaling

- The Objective is to obtain values of similar magnitude.
- Scaling makes variables a "standard size". It benefits algorithms that are scale-sensitive and generally does not hurt algorithms that are scale-insensitive.
- There is little downside to scaling features, in general.
- Warning: scaling is a common source of data leakage.
- Scaling requires global statistics that may be expensive to calculate.

Scaling

- Min-Max scaling to obtain values in the range [0, 1]:

$$x' = \frac{x - min(x)}{max(x) - min(x)}$$

- Scaling to an arbitrary range [a, b]:

$$x' = a + \frac{(x - min(x))(b - a)}{max(x) - min(x)}$$

- If we believe that the variable is normally distributed, it may be helpful to use:

$$x' = \frac{(x - mean(x))}{std(x)}$$

Recoding Variables

- From the perspective of variables and values, we sometimes talk about *code* or, more frequently, how a variable is *encoded*.
- The *encoding of a variable* is the representation of the data.
- Some of the models we use will often benefit or require that data be *encoded* in a form that is better suited for the model's inputs.
- Encoding (or recoding) often involves changing a variable type: creating dummy variables implies converting information from a categorical variable to a numeric variable, for example.

Recoding Variables: Discretization

- Discretization, quantization, or binning is the process of turning a continuous feature into a discrete feature.
- Discretization is performed by creating buckets for the given values.
- A disadvantage of discretization is that categorization introduces discontinuities at the category boundaries.

Dummy Variables and One-Hot Encoding

- Dummy variables and One-Hot encodings are forms of encoding categorical data.
- Dummy variables are binary: they will take a value of 0 or 1.
- Dummy variables are numerical, not categorical, factor, boolean, etc.
- With dummy variables, if the original variable contained *C* levels, then we will get *C-1* levels by default.
    - For instance, our example had five levels (one per weekday), but the resulting dummy representation only has four.
    - We can back out the fifth value since we know that when all four values are 0, the fifth value should be 1.
    - This avoids a sometimes undesirable situation for some methods called colinearity (one variable can be obtained as a linear function of others). Colinearity is one form of observing information redundancy.

Dummy Variables and One-Hot Encoding

- If the original value is missing, then all dummy variables are missing.
- If the data contains a novel value (a value that it hand not yet considered and encoded), then all values will be missing, unless explicitly considering other values.
- **One-hot encoding** is similar to dummy variables, but all values receive a column.

Encoding Categorical Features

- Categories are not static: categories change over time.
  - Categories were not represented in training data.
  - New categories may appear over time.
- It is generally a good idea to consider the category UNKNOWN.

Encoding Categorical Features

- In some cases, UNKNOWN labels may refer to samples that do not belong together: two new brands may not target the same market, new products, new IP addresses, new user accounts, etc.
- One solution is the hashing trick:
  - Use a hash function to generate a has for every category.
  - The hashed value will become the index of the category.
  - Some collisions may occur, but the overloading of the UNKNOWN category is reduced.

Feature Crossing

- Feature crossing is a technique to combine two or more features to generate new features.
- We may also benefit from establishing interaction terms. This type of transformation is primarily intended for numeric data. Still, it may be applied to categorical data after being transformed into dummy variables.

Feature Crossing

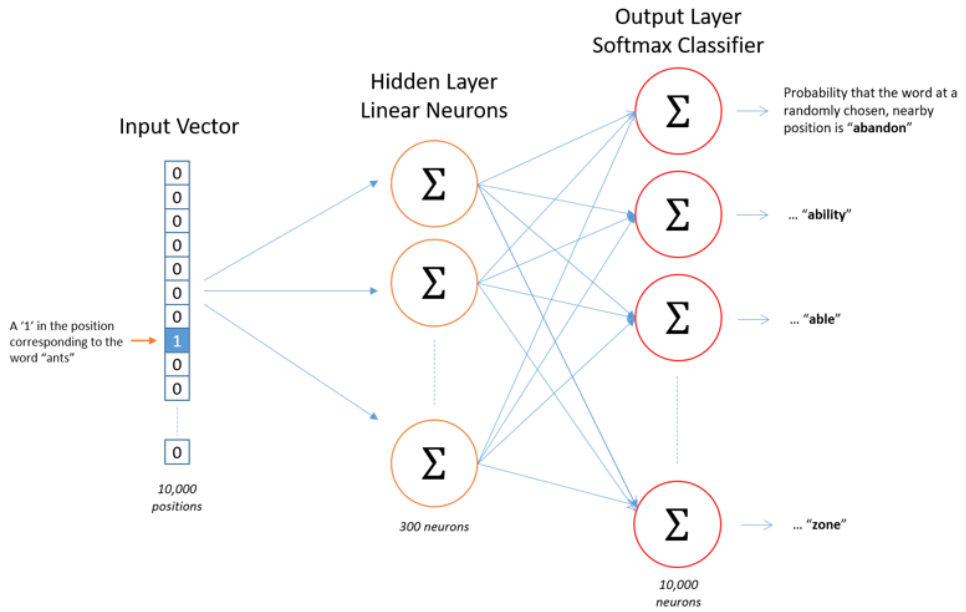
- Interaction variables typically capture the joint contribution to our predictive model of two or more variables after accounting for their contributions.
- A majority of cases will result in the model benefiting only marginally from these terms; however, they are fundamental in some contexts: for example, loan value and collateral value are typically included in default prediction models, together with their interaction term.

Multivariate Transformations

- Some transformations may include more complex formulations or the results of models that we use to pre-process the data.
- A couple of examples include:
- Principal Components Analysis:
    - Principal Components Analysis is a change of base such that orthogonal directions of maximum variation are used.
    - Compute PC Scores of a group of variables in the data and keep only the first n (up to a percent of variability explained).
    - Reduces redundant (highly correlated) information.

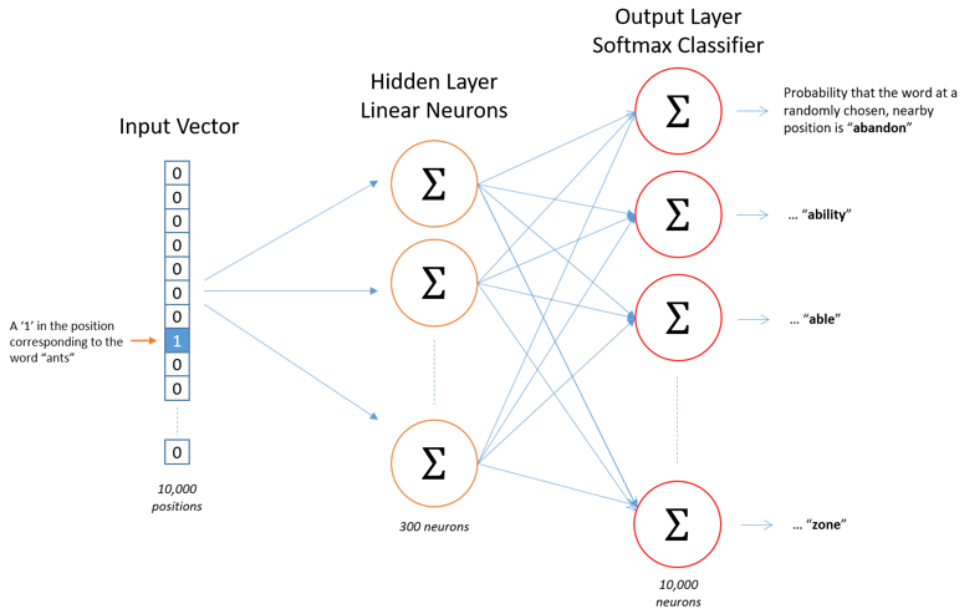
Multivariate Transformations

- Other transformations:
  - Discriminant Analysis Score: linear discriminant analysis produces a projection that maximizes linear separability.
  - Distance to cluster centroids.



Embeddings

- Training NN is computationally intensive and time-consuming.
- Assume that an NN has been trained:
  - Pre-trained NN are available.
  - Models can be trained on general language (news articles, Wikipedia, etc.) and specialized language (legal, medical, etc.) corpus.



Embeddings

- Word embeddings map words in vocabulary with an n-dimensional vector.
- A popular embedding is Word2Vec:
  - NN predicts a word from its context.
  - Similar to an autoencoder, obtained through a NN as a by-product.

Data Leakage

Data Leakage

- Data leakage refers to the situation when a form of the label "leaks" into the set of features used for making predictions, and this same information is not available during inference. Some common causes are discussed below.
- Splitting time-correlated data randomly instead of by time.
    - In many cases, we are dealing with time series data: the date-time in which data is generated affects its label distribution.
    - Ex: stock prices.
    - Solution: split data by time instead of random sampling whenever possible (ex., time-series cross-validation).

Data Leakage

- Scaling before splitting.
  - Scaling requires calculating statistics like mean, std, min, and max.
  - Data leakage occurs when stats are calculated on a data set that does not only contain training data.
  - Solution: split data before scaling; many frameworks do this automatically and ensure the appropriate splits.
- Filling in missing data with statistics from the test split.
  - Values used for imputation are calculated on complete data and not only the training portion.
  - Solution: split data before calculation and ensure only training data is used (use a framework).

Data Leakage

- Poor handling of data duplication.
  - Failure to remove duplicates or near-duplicates before splitting the data.
  - Data duplication can result from data collection or merging of different data sources.
  - As well, synthetic oversampling can induce duplication.
  - Solution: check for duplicates before splitting and also after splitting.

Data Leakage

- Group leakage.
  - Similar to duplication, where a group of examples have strongly correlated labels but are divided into different splits.
  - Example: in object detection, several pictures are taken a few seconds apart and almost identical.
- Leakage from the data generation process
  - The sampling mechanism may induce duplication.
  - Example: patient data in critical condition is duplicated because more tests are run.

Detecting Data Leakage

- Measure the predictive power of each feature.
  - Investigate unusually high readings.
  - Investigate data generation and if we can derive a valid explanation.
- Perform ablation studies (remove one feature at a time) to measure how important a feature or set of features is to your model.
  - If removing a feature causes the model's performance to deteriorate significantly, investigate why that feature is so important.
- Pay attention to new features added to the model.

Engineering Good Features

Too Many Features

- Too many features can be bad during training and model service.
- More features, more opportunities for data leakage.
- Too many features cause overfitting.
- More features require more memory to serve a model, which can increase hardware requirements and cost.

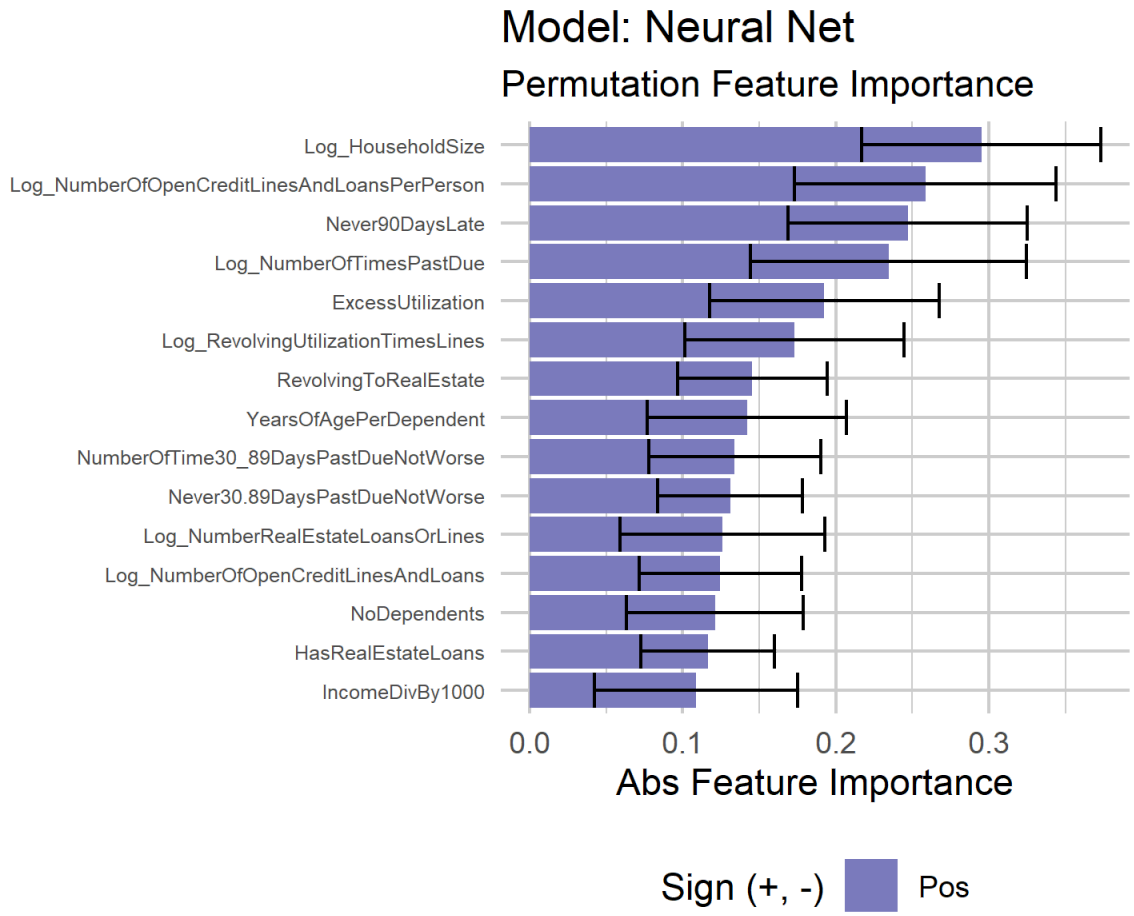
Too Many Features

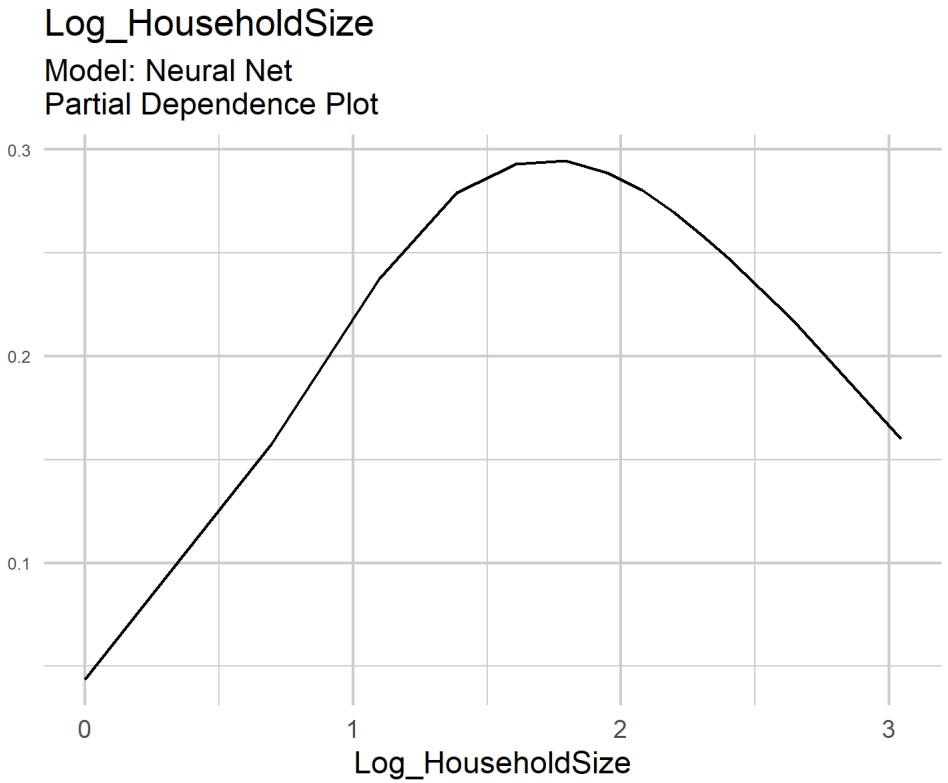
- Too many features can increase inference latency.
- Useless features become technicala debt.
- In principle, if a feature is not useful for prediction, regularization should get rid of it. In practice, it may be faster for the feature not to be available to the model in the first place.

Feature Importance

Permutation Feature Importance

- Measures the change in prediction error after a permutation of the values of a given feature.
- A feature is important if shuffling its values decreases the model performance (increases error).
- A feature is unimportant if shuffling its values does not change model performance.





Partial Dependence Plots

- Partial Dependence Plots (PDP) show the marginal effect of one or two features over the predicted outcome.
- Can show whether the relationship between feature and target is linear, monotonic, or more complex.
- For classification, PDP displays the probability of a particular class given the different feature values.

Limitations of Post-Hoc Explainability Methods: Permutation Importance

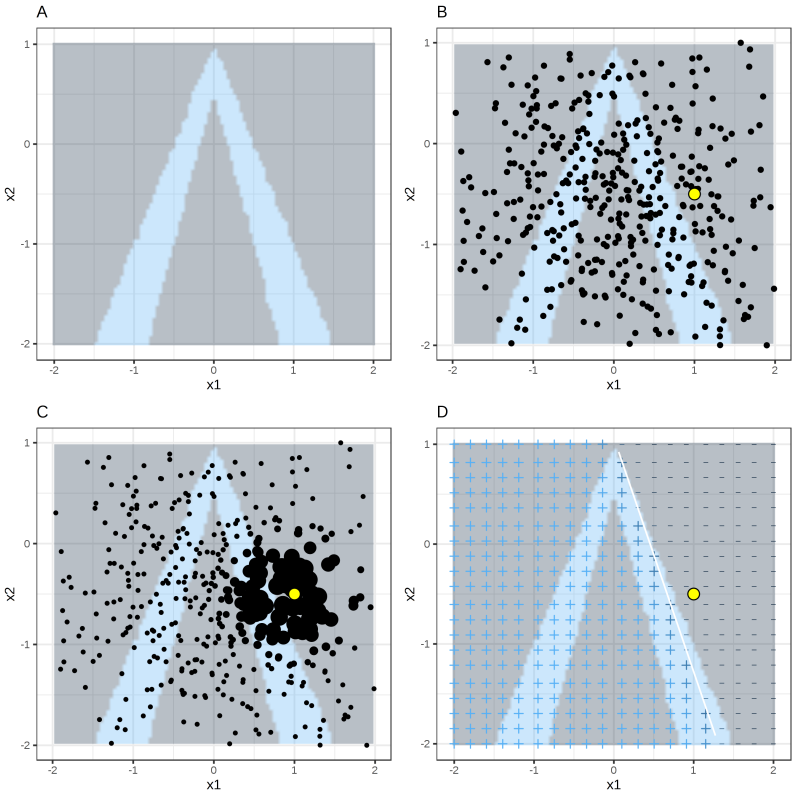
- Not clear if training or testing data should be used.
- Requires access to ground truth.
- Can be biased by unrealistic data instances.
- Adding a correlated feature can decrease the importance of the associated feature.

Limitations of Post-Hoc Explainability Methods: Partial Dependence Plots

- Maximum number of features per PDP is two.
- Some PDP do not show the feature distribution.
- Assumes independence between explanatory features: when features are correlated, we create new data points in areas of the feature distribution where the actual probability is very low.
- Heterogeneous effects might be hidden because PDP shows average marginal effects.

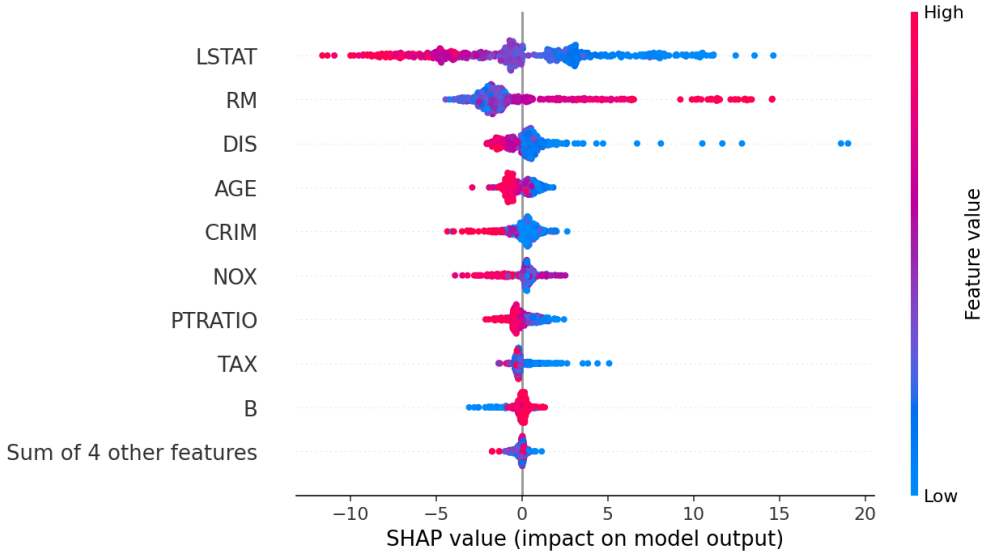
Local Surrogate Models

- Surrogate models are interpretable models trained to approximate the predictions of a black-box model.
- Local surrogate models are trained to explain individual predictions of black-box machine learning models.
- Local Interpretable Model-agnostic Explanation (LIME) is an approach for building a local surrogate model.



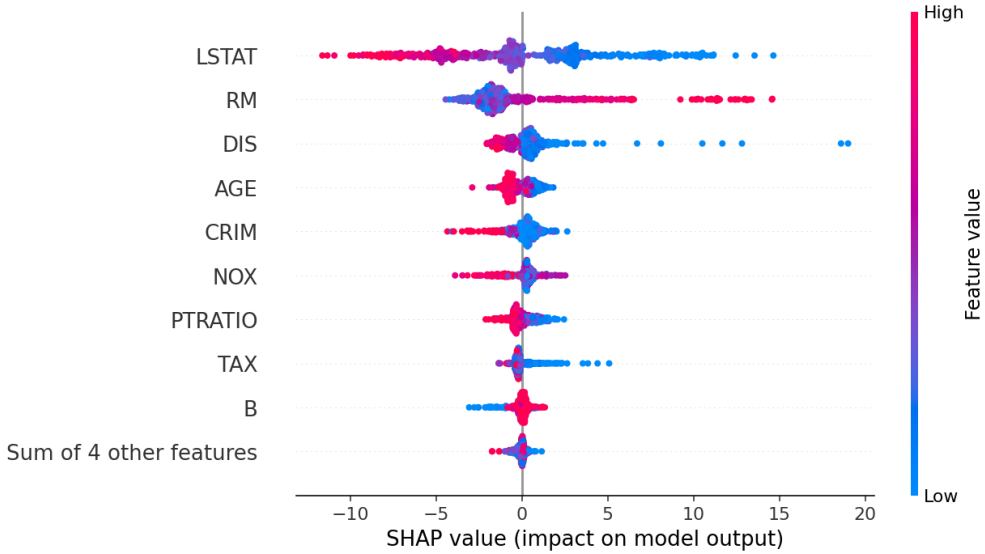
SHAP Values

- Objective: explain the prediction of an instance by computing the contribution of each feature to the prediction.
- Shapley value explanation is represented as an additive feature attribution method, a linear model.
- Local accuracy: each observation gets its own set of SHAP values.



SHAP Values

- Missingness: a missing feature gets an attribution of zero.
- Consistency: if a model changes so that the marginal contribution of a feature value increases or stays the same, the SHAP value also increases or stays the same.
- Linearity: as a consequence of consistency, SHAP is also linear.



Limitations of Post-Hoc Explainability Methods: Local Surrogate Models.

- The correct definition of the neighbourhood is an unsolved problem when using LIME for tabular data.
- Sampling can lead to unlikely data points.
- Fidelity-sparsity tradeoff: the complexity of the explanation must be defined in advance.
- Explanations can be unstable for nearby points.
- LIME explanations can be manipulated to hide biases.

Limitations of Post-Hoc Explainability Methods: SHAP Values

- Difficult to interpret:
  - Incorrect: the Shapley value of a feature value is the difference of the predicted value after removing the feature from the model training.
  - Correct: given the current set of feature values, the contribution of a feature value to the difference between the actual prediction and the mean prediction is the estimated Shapley value.
- Method (non-Tree implementation) is computationally expensive.
- SHAP (non-Tree) ignores feature dependence.

References

References

- Agrawal, A. et al. "Cloudy with high chance of DBMS: A 10-year prediction for Enterprise-Grade ML." arXiv preprint arXiv:1909.00084 (2019).
- Gilyadov, J (2017). Word2Vec Explained. URL
- Huyen, Chip. "Designing machine learning systems." O'Reilly Media, Inc.(2022).
- Lunderberg and Lee. A Unified Approach to INterpreting Model Predictions. Advances in Nueral INformation Processing Systems 30 (NIPS 2017). GitHub Repository
- Molnar, C. Interpretable Machine Learning. (2023) URL