5 Model-agnostic methods

5.1 PDP

- How it works
  - the recipe for partial dependence plots is: 1) Select feature. 2) Define grid. 3) Per grid value: a) Replace feature with grid value and b) average predictions. 4) Draw curve.
- The math
  - $\hat{f}_{x_S,PDP}(x_s) = E_{X_C}[\hat{f}(x_s, X_C)] = \int_{x_C} \hat{f}(x_s, x_C)P(x_C)dx_C$ where x_s is the feature of interest at a specific value, and x_C are all the other features.
- Advantages
- Disadvantages
  - If features of a machine learning model are correlated, the partial dependence plot cannot be trusted. The computation of a partial dependence plot for a feature that is strongly correlated with other features involves averaging predictions of artificial data instances that are unlikely in reality.

5.2 Individual Conditional Expectation (ICE)

5.3 Accumulated Local Effects (ALE)

- ALE paper: [Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models]
- Comparison of PDP, M-Plots, and ALE: to summarize how each type of plot (PDP, M, ALE) calculates the effect of a feature at a certain grid value v:
  - **Partial Dependence Plots (averages over marginal distribution of all the other features)**: “Let me show you what the model predicts on average when each data instance has the value v for that feature. I ignore whether the value v makes sense for all data instances.”
  - **M-Plots (averages over conditional distributions of all the other features)**: “Let me show you what the model predicts on average for data instances that have values close to v for that feature. The effect could be due to that feature, but also due to correlated features.” The math for M-plots is as follows, $\hat{f}_{x_S,M}(x_S) = E_{X_C|X_S}[\hat{f}(X_S, X_C)|X_S = x_s] = \int_{x_C} \hat{f}(x_S, x_C)P(x_C|x_S)dx_C$. In practice, we just need to define a neighborhood of x_s to sample.
  - **ALE plots** average changes in the predictions and accumulate them over the grid. It calculates the prediction differences conditional on features S and integrates the derivative over features S to estimate the effect. “Let me show you

how the model predictions change in a small”window" of the feature around v for data instances in that window."
    - In practice, the ALE for feature j estimated at (around) z_j is computed by first calculating the average (among instances appear within the grid interval) differences in predictions when replacing the feature of interest with grid values z, and then accumulating the average effects across all previous intervals, finally, the effects will be centered so that the mean effect is zero.
        - The value of ALE can be interpreted as the main effect of the feature at a certain value compared to the average prediction of the data.
        - Intervals that define the grid are chosen using the quantiles from the distributions of each features.
        - **ALE vs PDP.** PDP always shows the total effect, ALE shows the first- or second- (in case of two-feature plot) effects
- Advantages of ALE pots
    - Unbiased and work with correlated data
    - Faster to compute and scale with O(n). Notice that each instance is only used once for plotting the ALE for a specific feature.
    - Better interpretability: Conditional on a given value, the relative effect of changing the feature on the prediction.
- Disadvantages.
    - Can become a bit shaky (small ups and downs) with a high number of intervals
    - No ICE curves accompanied
    - Second-order effect plots are hard to interpret
    - Implementation of ALE plots is more complex and less intuitive compared to PDPs.

5.4 Feature Interactions

- Friedman's H-Statistic is defined using partial dependence function to measure the interactions between two features. It can be interpreted as the share of variance that is explained by the interaction.

5.5 Feature Importance

- **Permutation feature importance** is calculated as the increase (absolute or relative) in model’s prediction error (or other performance measures) after permuting the feature.
- Using **Training Vs Test set** to calculate permutation importance
    - Case for using test set: measures on the training set usually don’t reflect true performances of the model
    - Case for using training set: Using the training set tells us how much the model relies on each features for making predictions (think about the PDP)
- Advantages

    - Nice interpretation: increase in model error when the feature's information is destroyed.
    - Take into account feature interactions as well when one feature is permuted.
- Disadvantages
    - Unclear to use training vs test set
    - Need true labels to calculate permutation importance
    - Similar to PDPs, If features are correlated, the permutation feature importance can be biased by unrealistic data instances.

5.6 Global Surrogate

One way to interpret a blax-box model f is to train a interpretable (glm, dt) surrogate model f on a selected dataset (e.g., subset of the training set) to predict the outcomes of f. How well the surrogate model replicates the black-box model can be measured using R^2

5.7 Local Surrogate (LIME)

- Local surrogate models are interpretable models that are used to explain ***individual*** predictions of black box machine learning models.
- Recipe for training local surrogate models
    - Select individual instance of interest
    - Perturb dataset (re-sampling from the original distribution, etc) and get black-box model predictions
    - Weight the new samples according to their proximity to the instance of interest.
    - Train a weighted, interpretable model
    - Explain the prediction by interpreting the local model on the instance of interest
- The mathematical expression of local surrogate models with interpretability/complexity constraints is as follows $explanation(x) = argmin_{g \in G} L(f, g, \pi_x) + \Omega(g)$,
  Where $L(\cdot)$ is performance measure (e.g., RMSE of predicting the black-box model scores), π_x is the proximity measure that defines how large the neighborhood around instance x is that we consider for the explanation. $\Omega(g)$ is the complexity measure defined by users (e.g., number of features g could use)
- LIME stands for **L**ocal **I**nterpretable **M**odel-agnostic **E**xplanations. It works for various types of data (i.e., tabular, text, image)
    - For tabular data, LIME currently samples data points from a Gaussian distribution (mean/variance extracted from the training data) ignoring the correlation between features. It also uses an exponential smoothing kernel to define the neighborhood, with kernel width being $0.75 \times \sqrt{ncols}$. The resulting local surrogate model could be very sensitive to the proximity measure parameters, which is a major disadvantage of LIME
    - LIME as currently implemented is not sufficient for complete attributions thus not suitable for interpretability in legal/compliance scenarios.

    - LIME paper: "Why Should I Trust You?": Explaining the Predictions of Any Classifier

5.8 Scoped Rules (Anchors)

- Anchors explains **individual predictions** of any black-box classification model by finding a decision rule that "**anchors**" the prediction sufficiently. A rule (*IF-THEN*) anchors a prediction if changes in other feature values do not affect the prediction.

5.9 Shapley Values

- Goal
  - Shapley values are used to distribute the “payout” (i.e., the difference between the prediction of a single instance and the average prediction) among features.
- Interpretation
  - **The shapley value is the average contribution of a feature value to the prediction in different coalitions.**
  - The exact interpretation of the Shapley value for feature value j is: The value of the j-th feature contributed φ_j to the prediction of this particular instance compared to the average prediction for the dataset.
- Details
  - In a linear model, the contribution of the j-th feature on the prediction $\hat{f}(x)$ is $\varphi_j(\hat{f}) = \beta_j x_j - E(\beta_j X_j) = \beta_j x_j - \beta_j E(X_j)$. To sum up the contribution of all features, we have the expected value for the datapoint x minus the average prediction $\sum_{j=i}^{p} \varphi_j(\hat{f}) = \sum_{j=i}^{p}(\beta_j x_j - \beta_j E(X_j)) = (\beta_0 + \sum_{j=i}^{p} \beta_j x_j) - (\beta_0 + \beta_j E(X_j)) = \hat{f}(x) - E(\hat{f}(X))$
  - Similarly, the Shapley value of a feature value is its contribution to the payout, weighted and summed over all possible feature value combinations.
  - An approximation method based on monte carlo sampling is proposed

Approximate Shapley estimation for single feature value:

- Output: Shapley value for the value of the j-th feature
- Required: Number of iterations M, instance of interest x, feature index j, data matrix X, and machine learning model f
- For all m = 1,...,M:
  - Draw random instance z from the data matrix X
  - Choose a random permutation o of the feature values
  - Order instance x: $x_o = (x_{(1)}, \ldots, x_{(j)}, \ldots, x_{(p)})$
  - Order instance z: $z_o = (z_{(1)}, \ldots, z_{(j)}, \ldots, z_{(p)})$
  - Construct two new instances
  - With feature j: $x_{+j} = (x_{(1)}, \ldots, x_{(j-1)}, x_{(j)}, z_{(j+1)}, \ldots, z_{(p)})$
  - Without feature j: $x_{-j} = (x_{(1)}, \ldots, x_{(j-1)}, z_{(j)}, z_{(j+1)}, \ldots, z_{(p)})$
  - Compute marginal contribution: $\phi_j^m = \hat{f}(x_{+j}) - \hat{f}(x_{-j})$
- Compute Shapley value as the average: $\phi_j(x) = \frac{1}{M} \sum_{m=1}^{M} \phi_j^m$

    - Notice that in the feature order step(s), all features are randomly ordered so feature j could be placed such that a randomly selected set of features will be replaced by values from instance z.

        - Averaging implicitly weighs samples by the probability distribution of X. The procedure has to be repeated for each of the features to get all Shapley values.
- Advantages
    - The difference between the prediction and the average prediction is **fairly distributed** among feature values of an instance, which makes Shapley Value preferable for legal & compliance explanations.
    - Theoretically proved nice features such as Efficiency, Symmetry, Dummy, and Additivity, as detailed in Chapter 5.8.3.
- Disadvantages
    - Impossible to compute the exact value, monte-carlo based approximation method can lead to high variance in computed Shapley values.
    - Not very easy to interpret, always need to use all features for explanation, cannot be used to make pairwise relational statement between features and target.
    - From an implementation perspective
        - Need to access the training set instead of just the prediction function to calculate Shapley values
        - Permutation-based method suffers from inclusion of unrealistic data instances when features are correlated.