After training the model, we may want to dive deeper to see why the model makes such predictions. A popular approach to explaining machine predictions is to identify important features. Typically, these explanations assign a value to each feature (usually a word in NLP).

In this project, we adopt two methods for assigning feature importance:

1. built-in feature importance
2. permutation feature importance.

Built-in feature importance is embedded in the machine learning model, such as coefficients in our SVM and logistic regression models.

Permutation feature importance measures the increase in the prediction error of the model after we permuted the feature’s values, which breaks the relationship between the feature and the true outcome.

The concept is really straightforward: We measure the importance of a feature by calculating the increase in the model’s prediction error after permuting the feature. A feature is “important” if shuffling its values increases the model error, because in this case the model relied on the feature for the prediction. A feature is “unimportant” if shuffling its values leaves the model error unchanged, because in this case the model ignored the feature for the prediction.

The permutation feature importance algorithm based on Fisher, Rudin, and Dominici (2018):

Input: Trained model f, feature matrix X, target vector y, error measure L(y,f).

1. Estimate the original model error eorig = L(y, f(X)) (e.g. mean squared error)
2. For each feature j = 1, …, p do:
   * Generate feature matrix Xperm by permuting feature j in the data X. This breaks the association between feature j and true outcome y.
   * Estimate error eperm= L(Y,f(Xperm)) based on the predictions of the permuted data.
   * Calculate permutation feature importance FIj= eperm/eorig. Alternatively, the difference can be used: FIj = eperm- eorig
3. Sort features by descending FI.

The following table shows 20 most important features identified by different methods for different models. Note that neural network does not have built-in importance for each feature as SVM and LR.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SVM  built-in | SVM permutation | NN permutation | LR  built-in | LR permutation |
| |  | | --- | | great | | not | | works | | excellent | | t | | price | | money | | fine | | work | | junk | | poor | | bad | | good | | value | | best | | love | | fast | | back | | out | | not\_the | | |  | | --- | | not | | my | | love | | far | | easy | | best | | easy\_to | | than | | great | | t | | work | | works | | bad | | that\_i | | fast | | after | | your | | money | | m | | time | | |  | | --- | | not | | great | | work | | using | | like | | easy | | use | | easy\_to | | price | | works | | you | | excellent | | out | | very | | then | | never | | money | | unit | | now | | better | | |  | | --- | | great | | not | | work | | excellent | | price | | t | | works | | good | | waste | | money | | out | | fast | | fine | | love | | back | | best | | after | | not\_work | | junk | | very | | |  | | --- | | not | | great | | t | | work | | money | | ipod | | your | | how | | never | | well | | buy | | my | | good | | product | | you | | value | | sound | | very | | price | | easy | |