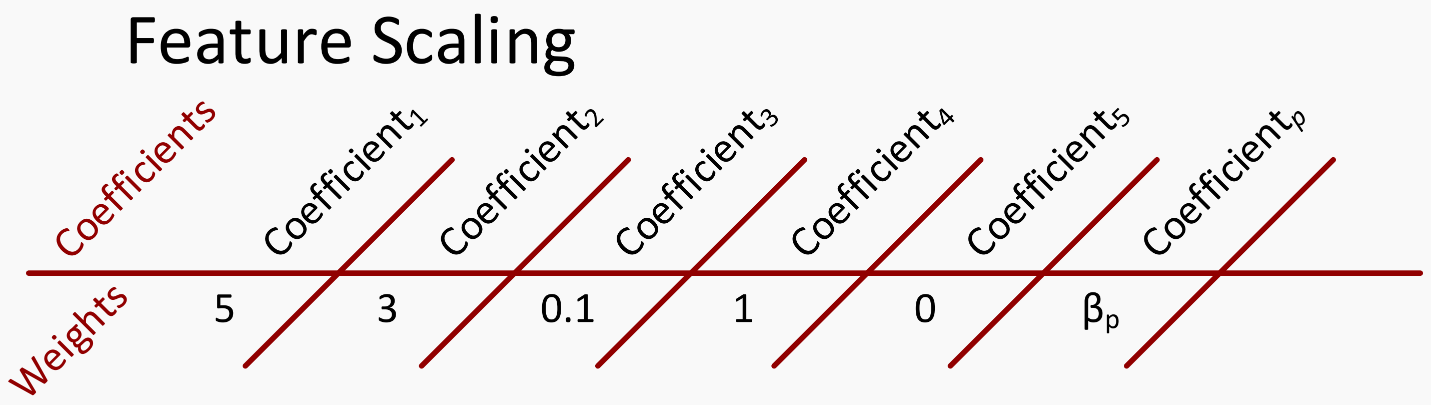
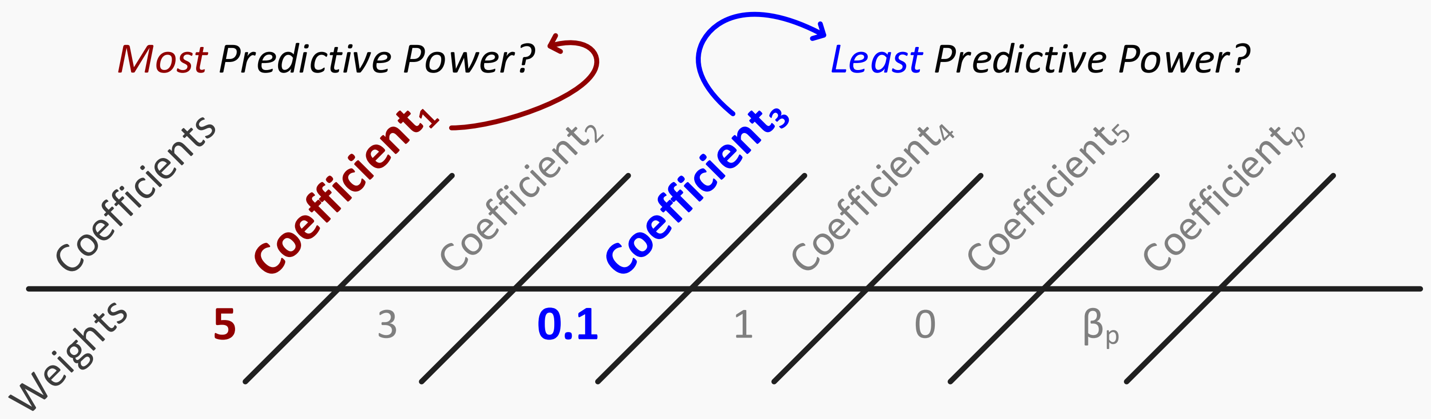
feature scaling

Alternatively, another method for **Feature Selection** is the process of **Feature Scaling**.

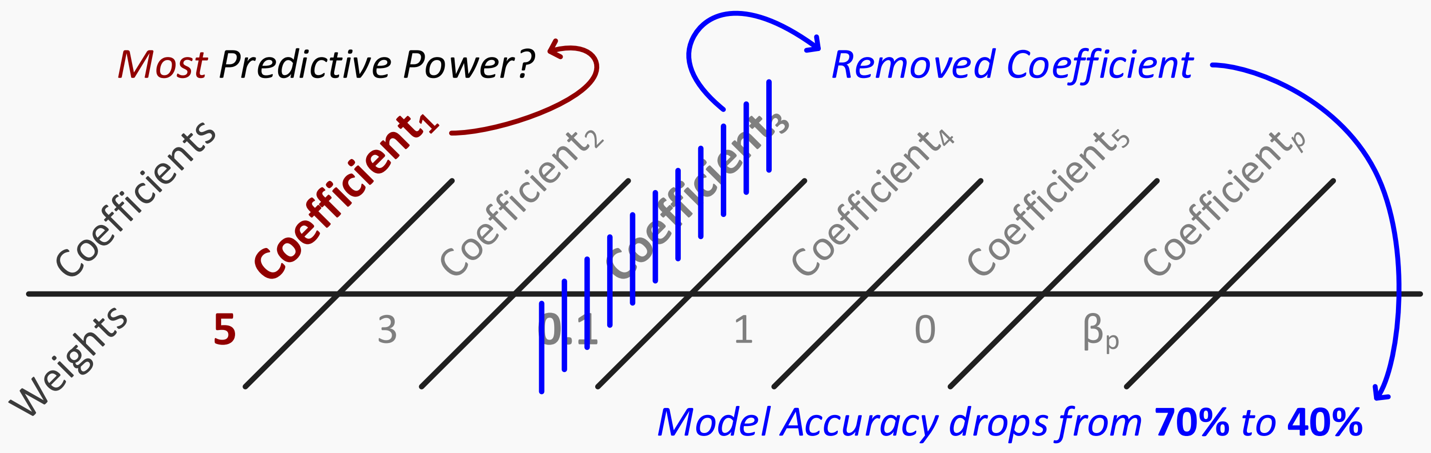
The following hypothetical illustrates **Feature Scaling**:

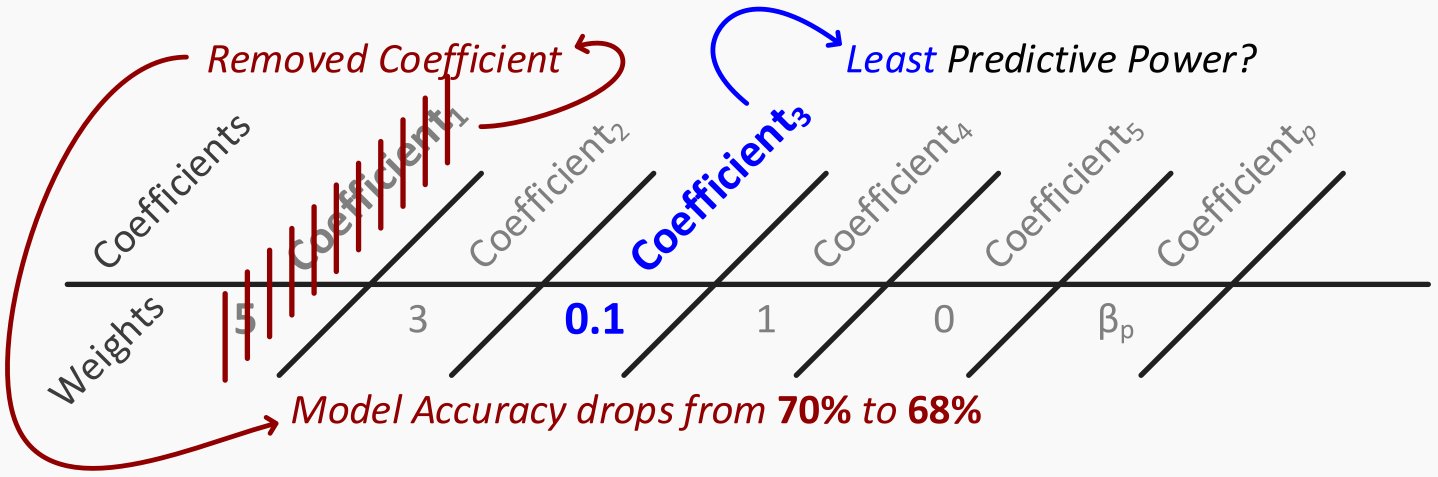


Assuming the above weights assigned to a given machine learning algorithm,



It can be noted that $\mathbf{Coefficient_1}$ appears with have the **most** predictive power with $\mathbf{Coefficient_3}$ having the **least** predicative power. With the latter intuition, it is expected to remove $\mathbf{Coefficient_3}$. However the decision to drop the low-valued parameter is a **learning fallacy**. $\mathbf{Coefficient_3}$ could potentially be a highly valuable parameter to model performance illustrated as follows:





Furthering the illustration above, intentionally (or unintentionally) removing the **most** valuable $\mathbf{Coefficient_1}$ would expect a high penalty on the model's performance. Conversely, the result could be less severe, proving the **appearance** of $\mathbf{Coefficient_1}$ having high importance is a **learning fallacy**.

The above outcome could be explained by $\mathbf{Coefficient_1}$ being measures on a **Range of 0 to 1**, contributing $\leq \mathbf{5\%}$ of the model's score. Additionally, $\mathbf{Coefficient_3}$ could be responsible for contributing $\leq \mathbf{14.5\%}$ to the model's score and measured on a **range of 0 to 145**.

Poor scaling equally distorts the interpretation of coefficients and has a negative effect on regularization. **Regularization** applies pressure on the larger coefficients to become smaller, meaning the features with the smallest values will be reduced; the latter is **not the intent of the data scientist**.

For example, when attempting to **regularize** all coefficients equally:

$$\text{Regularization}(f) = \beta_1^2 + \beta_2^2$$

$$5^2 + 0.10^2 \quad \rightarrow \quad 25.01$$

$$\text{Reduce Feature}_1 \text{ by } 20\% \rightarrow 4^2 + 0.10^2 \quad \rightarrow \quad 16.01$$

$$\text{Reduce Feature}_2 \text{ by } 20\% \rightarrow 5^2 + 0.08^2 \quad \rightarrow \quad 25.0064$$

Given the application of $\boldsymbol{\ell_2}$ **Regularization** and two equally important features that are measured on **different scales**, the value of one coefficient is considerably larger than the other. The values of the first feature are small (0 or 1), therefore the coefficient is large. Conversely, the values of the second feature are small (up to 145), therefore the coefficient is small. **Regularization** is applied by reducing the coefficient of each feature by 20% as seen in the illustration. The impact on the aggregate **Regularization Term** is highly disparate depending upon the chosen coefficient to reduce. The example illustrates the tendency for the **Regularization Term** to prefer reduction of the first coefficient opposed to the second, regardless of the features having **equal importance**. Ensuring the features are on a relative scale will prevent regularization from biasing features for arbitrary reasons.

Improperly Scaled features will also have a negative impact towards **distance metrics**. Concretely, **regularization** can have an impact on the **loss function** itself. If a loss function depends on Euclidean distances between $\boldsymbol{x_i's}$, the same problem arises as with the above illustration; the algorithm may inappropriately determine certain dimensions more valuable over others.

An appropriate solution would be to scale each feature to be measured **between -1 and 1** (or between 0 and 1). The latter is achieved through the following expression, however, there are various methods to effectively scale features in a dataset. It is important to take these measures as a part of **preprocessing** data prior to constructing a functioning model.

$$x^{\text{scaled}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

sweeping parameters » tuning model hyperparameters

The **Tune Model Hyperparameters** module in Microsoft AzureML is used to automatically search for optimal combinations of machine learning model hyperparameters.

The hyperparameter space can be searched in several ways; on a regular or randomized grid of the hyperparameter space. Alternatively, the hyperparameter space can be randomly sampled. In either case, the search of hyperparameter space can be computationally intensive, since the model must re-compute and evaluate the model. Fortunately, most commonly used machine learning models are not particularly sensitive to the choice of hyperparameters.