1 LR Loss function - First Step in Fit

The default loss function used is ordinary least squares. Fit works, primarily in the context of linear models. Mathematical function that quantifies how well a machine learning model's predictions align with the actual target values. It measures the "cost" associated with prediction errors. The most common way to learn these parameters in LR is by minimizing the quadratic loss/cost between the actual target and the model predictions. This is called ordinary least squares (OLS).

Squared Error Loss-Regression penalizes really heavily for big errors compared to the absolute loss. If you have a data point with a big error, the loss function is going to focus a lot on it as the penalty is very high for this this data point.

Absolute value loss-Regression we do not punish large errors as severely as ordinary least squares. Gives us robust regression.

0/1 loss-Classifiers Binary state of happiness. Accuracy is directly related to 0-1 loss. Logistic loss uses predicted probabilities. Training error (1 - accuracy) and Logistic loss are fundamentally different things.

Exponential loss-Classifiers The function gets smaller as $y_i x_i^T w$ gets larger, so it encourages correct classification.

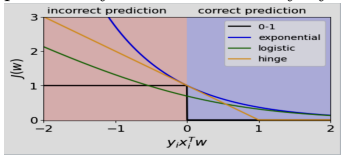
Hinge loss - Classifiers Confident and correct examples are not penalized Grows linearly for negative values in a linear fashion. Hinge loss with L2 regularization, it's called a linear support vector machine. One can choose a loss function from a pre-defined set for a SGDClassifier model. The l1_ratio parameter only matters when penalty='elasticnet' One can choose any loss function out of 'hinge', 'log_loss', 'modified_huber', 'squared_hinge', 'perceptron', 'squared_error', 'huber', 'epsilon_insensitive', 'squared_epsilon_insensitive' for sklearn SGDClassifier.

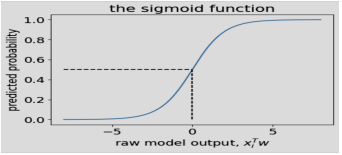
Logistic loss (logloss) - Classifiers Used in logistic regression. Grows linearly for negative values which makes it less sensitive to outliers.

Sigmoid loss - Classifiers Used in logistic regression. Maps $x_i^T w$ to a number in [0,1], to be interpreted as a probability.

Binary cross-entropy loss - Classifiers predicted probability of the positive class as defined above by the sigmoid function. This is also referred to as cross-entropy loss. The 0/1 version is used when we talk about probabilities whereas the -1/+1 version is used when we talk about the raw output of the classification.

Cross-entropy loss for multi-class classification The predicted probability for class is usually by applying the softmax function





1.1 Optimization algorithm - Second Step in Fit

For iteratively updating the weights so as to minimize the loss function. Minimizes the MSE. We use optimization algorithms (like gradient descent) to find the optimal parameters.

1.2 Lasso

regression problem with scaled numeric data with many irrelevant features.

1.3 Logistic regression

Working on text classification problem where you have sparse features encoded with bag-of-words representation. Speed and interpretation are important for you.

1.4 Random forests Trees

Interpretable, They can capture non-linear relationships. You have unscaled numeric data and better scores are important for you. The random state plays a role both in which observations and which features are selected to train an individual decision tree. Feature scaling often has little effect on model performance when using tree-based algorithms such as decision trees or random forests.

1.5 Elastic nets

Combine good properties from both regularization. L1 promotes sparsity and the L2 promotes smoothness. The functional is strictly convex: the solution is unique.

2 Regularization

Add a penalty to the loss function that increases with model complexity. With regularization, we minimize: The Loss measures fit to data, while the regularization term penalizes complexity. The parameter α controls how strong the penalty is.

2.1 L0 regularization:

Counts the number of non-zero weights: To increase the degrees of freedom by one, need to decrease the error by α Prefer smaller degrees of freedom if errors are similar. Penalizes the number of features. (feature selection RFE)

2.2 L1 regularization:

Sums absolute values of the weights. Makes one of the weights to zero for multi-colinear features.

Both result in lower validation error. Encourages sparsity (many zeros). (Lasso Regression)

Almost always improves the validation error.

Can learn with exponential number of irrelevant features

Less sensitive to changes in X

```
pipe_l1_rf = make_pipeline(
    StandardScaler(),
    SelectFromModel(Lasso(alpha=0.01, max_iter=100000)),
    RandomForestRegressor(),
)
```

2.3 L2 regularization:

Sums squared weights: Gives equal but reduce weightage to multi-colinear features. $\alpha = 0$ is same as OLS.

encourages small, smooth weights. (Ridge Regression)

2.4 Regularized logistic regression

Default logistic regression uses L2 regularization. The C hyperparameter decides the strength of regularization. Interpretation of C is inverse of lambda or alpha. L1 regularization is carrying out feature selection; Many coefficients are 0. Similar scores with less features! More interpretable model!

```
pipe_lgr_l2 = make_pipeline(StandardScaler(), LogisticRegression())
pipe_lgr_l1 = make_pipeline(StandardScaler(),
      LogisticRegression(solver="liblinear", penalty="l1"))
```

Feature selection using L1 regularization and pass selected features to another model.

```
  pipe_lgr_lgbm = make_pipeline(
    StandardScaler(),
    SelectFromModel(LogisticRegression(solver="liblinear",
         penalty="l1")),
    LGBMClassifier(verbose=-1),
)
```

Regression

- Least squares with L1- and L2-regularization: **ElasticNet**
- SVR (ϵ-insensitive loss function) epsilon = 0 gives us KernelRidge model (least squares with RBF)

Classification

- SVC (supports L2-regularization)
- LogisticRegression (support L1 and L2 with different solvers)

Regulization does not matter on decision trees or naive Bayes. Matters for k-NN since Distance will be affected more by large features than small features. It matters for regularized least squares: If you have colinear features, the weights would go crazy with regular linear regression.

With L2 regularization: The weight will be equally distributed among all collinear features because the solution is unique.

2.5 Why are small weights better?

Somewhat non-intuitive. Suppose x_1 and x_2 are nearby each other. We might expect that they have similar y. If we change feature1 value by a small amount, leaving everything else the same, we might think that the prediction would be the same. But if we have bigger weights, small change has a large effect on the prediction.

3 Ensemble Learning

Group of models or predictors. Often get better and reliable predictions than with a single best model. **High Variance:** high error during testing and validation. Overfitting. **High bias:** high error during training. Underfitting.

4 Ensemble techniques

Greater the diversity among the individual models, the better the ensemble will perform compared to its individual model because different models capture different patterns in the data, and by combining their predictions, the ensemble can leverage the strengths of each model while mitigating their weaknesses.

4.1 Bagging (Bootstrap Aggregating):Random Forests

Several models are trained independently (can be done in parallel) on different random subsets of the data and their predictions are averaged (for regression) or voted upon (for classification). fit a diverse set of that many decision trees by injecting randomness in the model construction. predict by voting (classification) or averaging (regression) of predictions given by individual models

Strengths Usually one of the best performing off-the-shelf classifiers without heavy tuning of hyperparameters. Don't require scaling of data. Less likely to overfit. Slower than decision trees because we are fitting multiple trees but can easily parallelize training because all trees are independent of each other.

Weaknesses Require more memory. Hard to interpret. Tend not to perform well on high dimensional sparse data such as text data.

4.2 Boosting:Gradient Boosted Trees (e.g., XGBoost, LightGBM, CatBoost)

Weak learner are trained sequentially where each new model focuses on correcting the errors made by the previous models. The final prediction is a weighted combination of all models.

4.3 Averaging

In the last two methods (bagging and boosting) we combined models of the same type (e.g. decision trees). Another way of introducing diversity is to combine different types of models. The idea behind averaging is to combine very different machine learning models. Random forest is an averaging model where the base estimators are decision trees.

4.4 Stacking

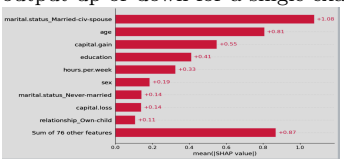
Base learner models of different types (different algorithm) are trained on the same dataset and a meta-model is trained to combine their predictions. Stacking is a heterogenous parallel method.

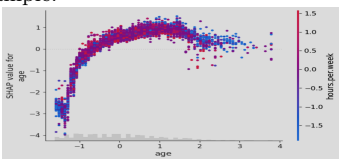
5 Permutation importance

By randomly shuffling the feature's values and measuring the resulting performance decrease. Permutation importances give us a sense of global feature relevance. But they only convey magnitude, not direction.

6 SHAP

The feature pushes the prediction higher or lower than the baseline. A Shapley value is created for each example and each feature. explanation is additive, so each feature is interpreted as nudging the output up or down for a single example.





6.1 Example

The plot shows a nonlinear relationship between age and the predicted probability of belonging to class 1 (earning more than $50K), as indicated by the SHAP values on the y-axis. Lower values of age have smaller SHAP values for class "¿50K". This means being younger generally decreases the predicted probability of earning more than $50K Similarly, very high age values also correspond to slightly smaller SHAP values for class "¿50K". This pattern is intuitive: younger individuals tend to earn less due to limited work experience, while older individuals may earn less as they approach retirement. Between these two extremes, there appears to be an optimal age range (around a scaled age of 1) where the SHAP values peak, indicating the highest contribution of age toward predicting income above $50K. hours.per.week The color gradient suggests a possible interaction effect of hours.per.week with age. Although this interaction is subtle across most of the plot, very young and very old individuals tend to work fewer hours per week. This lower number of working hours may partially explain the reduced SHAP values for these age groups and reinforces the relationship between age, work hours, and income prediction.