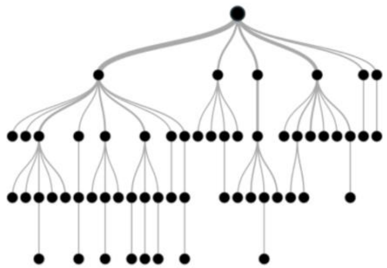
Decision Tree Classification

Decision Tree: (Classification and Regression Trees)

• A decision tree a **decision support** tool that uses a **tree-like model** of decision and their possible consequences.



• Decision tree discover the patterns in the form of rules, which can be easily understood.

•Decision tree is based on two key ideas: **(i) Recursive Partitioning (ii) Pruning.**

1. Recursive partitioning of the space of the predictors using training data (Full grown tree):
Recursive partitioning divides up the p-dimensional of space of x variables into non-overlapping multidimensional space.

How it works ?
• First, **one of the predictor** get selected to **divide** the whole dataset **into two parts** (say for binary classification) and then compute the **impurity reduction** after this split.
• By comparing the **reduction in impurity across all possible splits** for all possible predictors, the **split node get chosen** (Root node).
• Splitting process continue till there is no scope of impurity reduction to reach the leaf node.
• The most **two popular impurity measure** are 'Gini Index' and 'Entropy'.
• **Gini Index :** Impurity Range (0 - 0.5)
• **Entropy :** Impurity Range (0 - 1.0)

2. Pruning using Validation data : The idea behind **pruning** is **to remove the weakest branch** from the **full grown tree** to avoid over fitting.

Major disadvantage of Decision Tree is **over fitting** and that leads to **high-variance estimate of out-of-sample accuracy.**

Decision Tree Classification

Classification Tree Example:

A riding mower manufacturer want to predict the families in a city those would **likely to purchase** a riding mower or **not likely** to buy one.

A pilot random **sample** of owner and non-owner of riding mower is undertaken, based on that data we will build predictive model using decision tree to find the pattern of potential buyer.

Decision Tree **recursive partitioning** procedure:

(i) Idea is to identify the **best split** variable that can split the entire x-space into two partition such that each part as **pure(homogeneous)** as possible.

(ii) By comparing the **reduction in impurity** across all possible splits in all possible predictors, the next split is chosen.

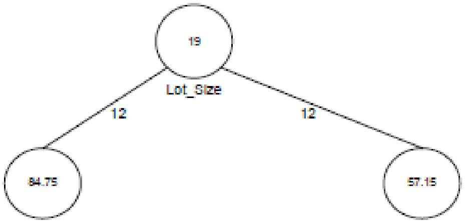
(ii) **Gini index** and **Entropy** can be used to measure the impurity.

Gini Index $I(A) = 1 - \sum_{k=1}^{m} p_k^2$ $\quad$ $Entropy(A) = -\sum_{k=1}^{m} p_k \log_2(p_k)$

pk is the proportion of class k and m is total classes of response variable

at the maximum impurity Gini Index = 0.5 , Entropy = 1

(iii) First best split is identified at Lot_size = 19

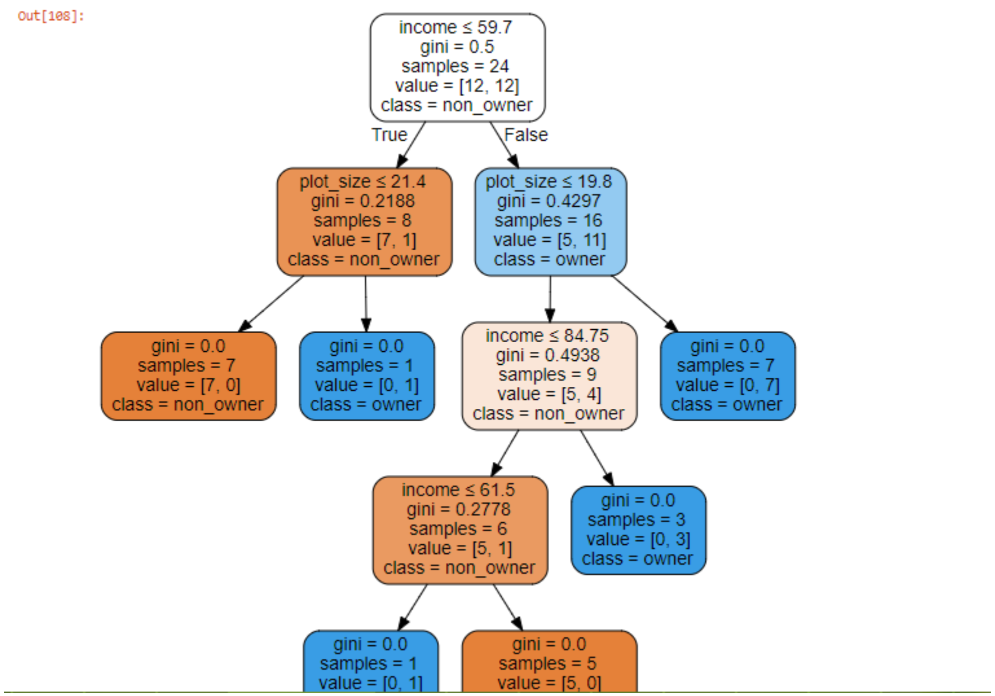


Lot Size, Income, and Ownership of a Riding Mower for 24 Households

Household number	Income ($ 000's)	Lot Size (000's ft^2)	Ownership of, riding mower
1	60	18.4	Owner
2	85.5	16.8	Owner
3	64.8	21.6	Owner
4	61.5	20.8	Owner
5	87	23.6	Owner
6	110.1	19.2	Owner
7	108	17.6	Owner
8	82.8	22.4	Owner
9	69	20	Owner
10	93	20.8	Owner
11	51	22	Owner
12	81	20	Owner
13	75	19.6	Non-Owner
14	52.8	20.8	Non-Owner
15	64.8	17.2	Non-Owner
16	43.2	20.4	Non-Owner
17	84	17.6	Non-Owner
18	49.2	17.6	Non-Owner
19	59.4	16	Non-Owner
20	66	18.4	Non-Owner
21	47.4	16.4	Non-Owner
22	33	18.8	Non-Owner
23	51	14	Non-Owner
24	63	14.8	Non-Owner

Decision Tree Classification

Full grown tree for riding mover use case, started with Root Node, Decision Nodes and Leaf Nodes.



Lot Size, Income, and Ownership of a Riding Mower for 24 Households

Household number	Income ($ 000's)	Lot Size (000's ft^2)	Ownership of, riding mower
1	60	18.4	Owner
2	85.5	16.8	Owner
3	64.8	21.6	Owner
4	61.5	20.8	Owner
5	87	23.6	Owner
6	110.1	19.2	Owner
7	108	17.6	Owner
8	82.8	22.4	Owner
9	69	20	Owner
10	93	20.8	Owner
11	51	22	Owner
12	81	20	Owner
13	75	19.6	Non-Owner
14	52.8	20.8	Non-Owner
15	64.8	17.2	Non-Owner
16	43.2	20.4	Non-Owner
17	84	17.6	Non-Owner
18	49.2	17.6	Non-Owner
19	59.4	16	Non-Owner
20	66	18.4	Non-Owner
21	47.4	16.4	Non-Owner
22	33	18.8	Non-Owner
23	51	14	Non-Owner
24	63	14.8	Non-Owner

Decision Rules Examples:

IF [Income <= 59.7] AND [Plot_Size <= 21.4] ➔ Then "Non-Owner"

IF [Income > 59.7] AND [Plot_Size > 19.8] ➔ Then "Owner"

Decision Tree Classification

Overfitting and Tree Pruning.

Overfitting : a classification tree may over fit the training data.

- Too many branches, some may reflect due to outliers or noise.
- Poor accuracy on unseen data/ validation data.

Two approaches to avoid overfitting.

- Pre-Pruning : Stop the tree construction early, do not split a node if this would result in the goodness measure falling below a threshold (tree depth, min num of records in a node, min reduction of impurity).

- Post-Pruning : Remove branches from a 'fully grown tree' by using the set of data different from training data (validation data) to decide the 'best pruned tree'.

Problem with decision tree is having **low bias** and suffer from **high Variance**, that can be reduced by using the ensemble concepts (Bagging and Boosting).

Decision Tree Classification – Test Accuracy

```python
# Train and test data validation
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.4, random_state=1)
```

	actual	predicted
13	non-owner	non-owner
18	non-owner	owner
3	owner	owner
14	non-owner	owner
20	non-owner	non-owner
17	non-owner	non-owner
10	owner	non-owner
4	owner	owner
2	owner	owner
19	non-owner	owner

```python
# Training accuracy
y_train_pred = ctree_model_2.predict(x_train)
print(metrics.accuracy_score(y_train, y_train_pred))
```

```
1.0
```

```python
# Testing accuracy
y_test_pred = ctree_model_2.predict(x_test)
print(metrics.accuracy_score(y_test, y_test_pred))
```

```
0.6
```

Lot Size, Income, and Ownership of a Riding Mower for 24 H

Household number	Income ($ 000's)	Lot Size (000's ft^2)	Ownership of, riding mower
1	60	18.4	Owner
2	85.5	16.8	Owner
3	64.8	21.6	Owner
4	61.5	20.8	Owner
5	87	23.6	Owner
6	110.1	19.2	Owner
7	108	17.6	Owner
8	82.8	22.4	Owner
9	69	20	Owner
10	93	20.8	Owner
11	51	22	Owner
12	81	20	Owner
13	75	19.6	Non-Owner
14	52.8	20.8	Non-Owner
15	64.8	17.2	Non-Owner
16	43.2	20.4	Non-Owner
17	84	17.6	Non-Owner
18	49.2	17.6	Non-Owner
19	59.4	16	Non-Owner
20	66	18.4	Non-Owner
21	47.4	16.4	Non-Owner
22	33	18.8	Non-Owner
23	51	14	Non-Owner
24	63	14.8	Non-Owner

```python
param_grid = { "criterion"      : ['gini', 'entropy'],
               "max_features"   : [1, 2 ],
               "splitter"       : ['best', 'random'],
               "min_samples_split"  : [2, 3],
               "min_samples_leaf"  : [ 2, 3]
              |
              }
```

Model Evaluation Process

Model Evaluation Process: Estimate the Model predictive performance on out of sample data.

- Maximizing the **Training Accuracy** will lead to complex model that over fit the model.
- Split the dataset into two parts **Training** and **Test** datasets so model can be trained on training data and tested on Test data.
- **Testing Accuracy** is better estimates than training accuracy for out of sample predictive performance.
- But **train-test** technique provides **high variance**.
- **K-fold Cross validation** can over come of high variance of train-test strategy.
- K-fold cross validation can be used the **tuning parameters** and choosing between the models.

K-fold Cross validation Steps:

(i) Split the complete datasets into K equal partitions.
(ii) Use One partition for testing set and union of remaining other partitions for training set.
(iii) Calculate the Testing Accuracy
(iv) Repeat the step Two and Three K times using different partitions.
(v) Calculate the average accuracy from all the partitions that would be used for out of sample accuracy.

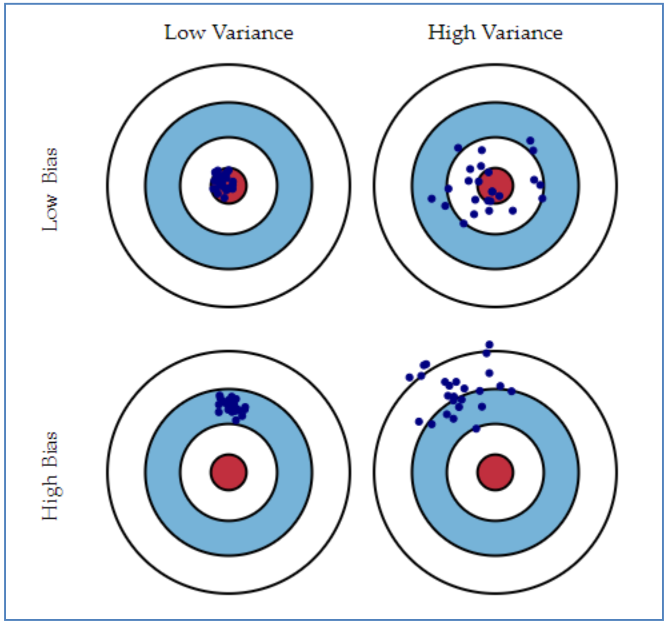
Bias vs Variance

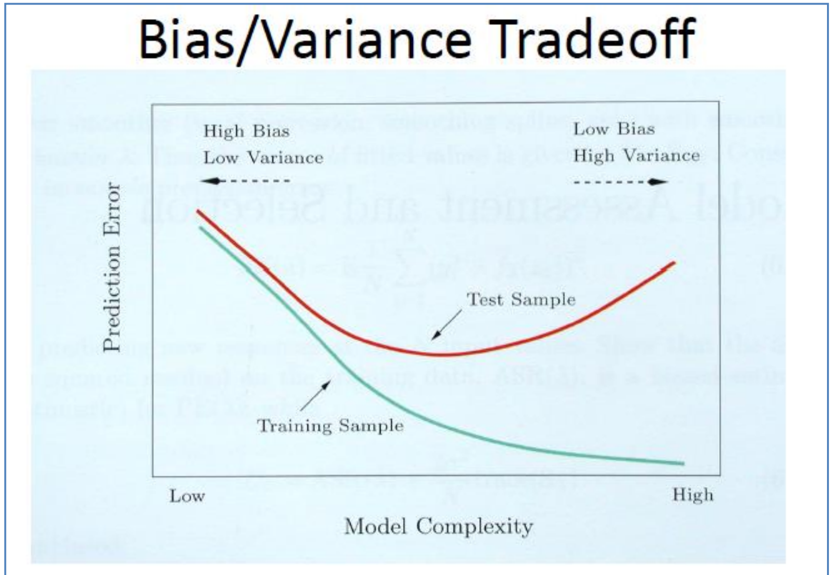
Prediction errors can be classified into two parts: (i) **errors due to bias** (ii) **errors due to variance.**

Error due to bias - is the difference between the average prediction of model and correct value.
Error due to variance – is the variability of model prediction for a given data points.

Imagine that the center of the target is a model that perfectly predicts the correct values. As we move away from the target, our predictions get worse and worse. Each hit represents an individual realization of our model, given the chance variability in the training data we gather.
Ex : Lets assume we have collected five different training data sets for the same problem, now imagine using same one algorithm to train five models.

Low bias algorithm train models that are accurate on average while high bias models are inaccurate on average.
Low variance algorithm train models that are consistent while high variance models are inconsistent.





Hastie, Tibshirani, Friedman "Elements of Statistical Learning" 2001

Reference : http://scott.fortmann-roe.com/docs/BiasVariance.html

Classification Tree Tuning

Classification Tree Tuning using SKLearn.

Ref : http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

(i) **criteria :** function to measure the quality of split (default is 'gini').
(a) 'gini' – Gini Index, (b) 'entropy' - Information Gain

(ii) **max_features :** Number of features to select while looking for best split.
(a) None - max_features=n_features (default is None)
(b) 'auto' or 'sqrt' – sqrt(n_features)
(c) 'log2' – log2(n_features)

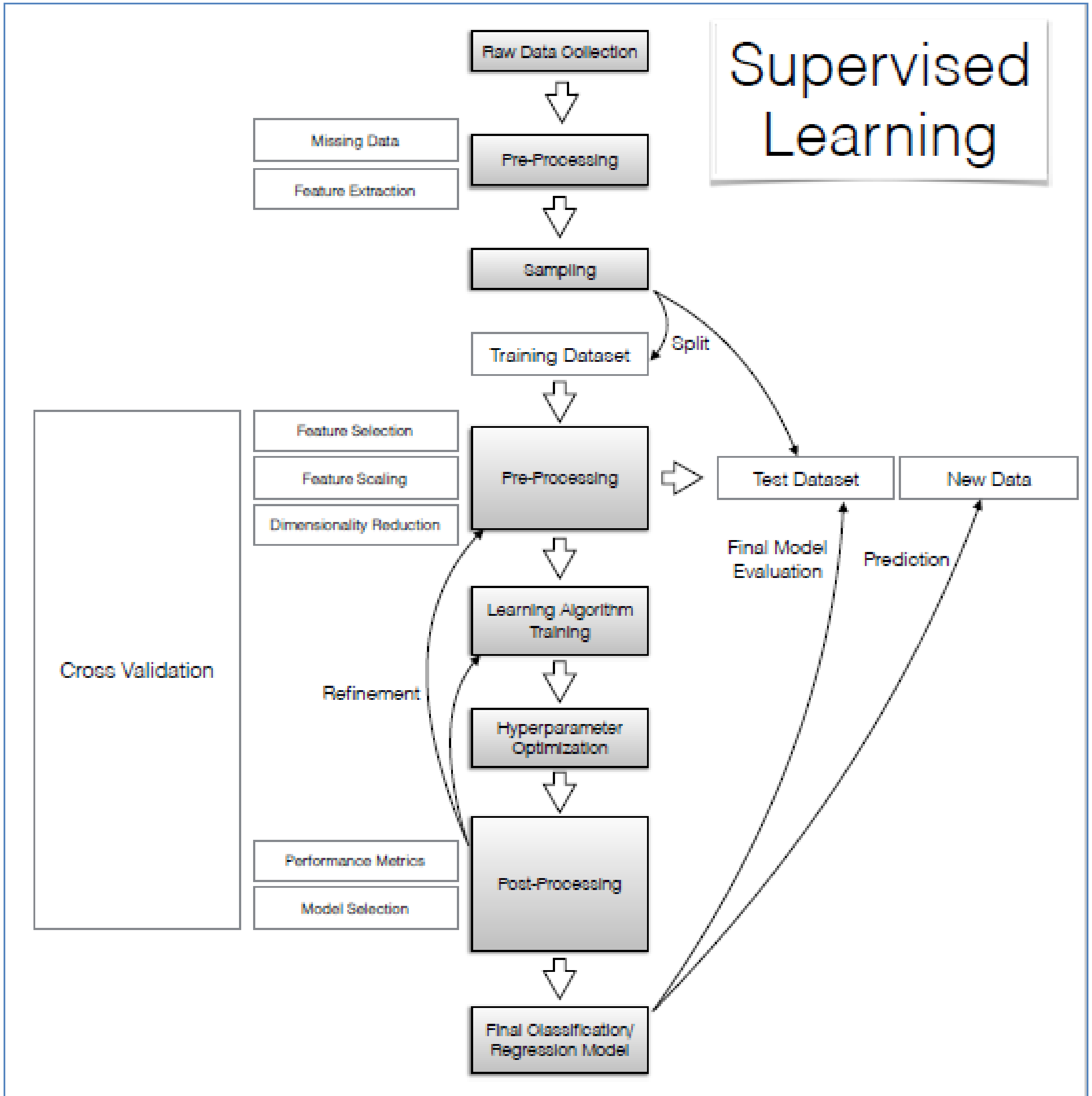
(iii) **min_samples_split :** Minimum number of samples reqd. to split an internal node (default = 2)

(iv) **min_samples_leaf :** Minimum number of samples required to be at a leaf node (default = 1)

(v) **max_depth :** Maximum depth of tree (default is None then nodes will get expanded until all leaves get pure).

(vi) **splitter** : Define the strategy to choose the split at each node. (default = 'best')
(a) – 'best' , (b) = 'random'

Supervised Learning Model Workflow



Classification Model Use Case

Model evaluation metrics:

Regression problems: Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
Classification problems: Classification accuracy

Classification accuracy is the easiest way to understand the classification accuracy, but it does not tell the complete details about classification errors.

Confusion matrix give us the complete picture how Classifier is performing, which Metrics should be used will depend on the business objective.

Confusion Matrix and ROC Curve

		Predicted Class	
		No	Yes
Observed Class	No	TN	FP
	Yes	FN	TP

TN	True Negative
FP	False Positive
FN	False Negative
TP	True Positive

Model Performance

Accuracy	=(TN+TP)/(TN+FP+FN+TP)
Precision	=TP/(FP+TP)
Sensitivity	=TP/(TP+FN)
Specificity	=TN/(TN+FP)

Fraud Detection Model. (Positive class – Fraud) In this case focus should be more reducing the FN and increasing the TP : optimize the sensitivity for higher %.

Spam Filter (positive class is 'spam'): In this case focus should be more reducing the FP and increasing the TN

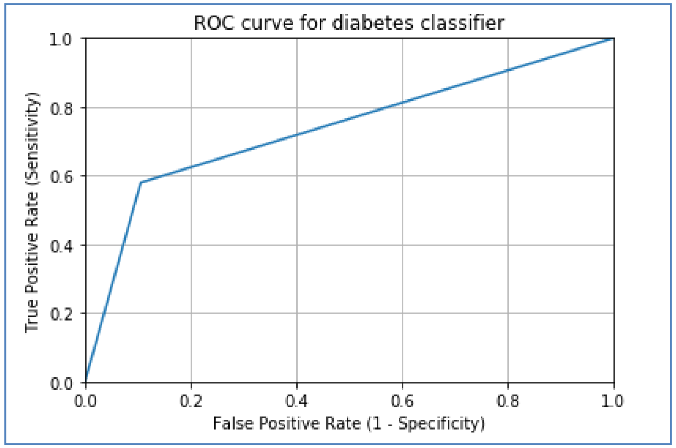
Basic terminology

True Positives (TP): Correctly prediction of Positive Class.
True Negatives (TN): Correctly prediction of Positive Class.
False Positives (FP): Incorrectly prediction as a Positive Class..(a "Type I error")
False Negatives (FN): Incorrectly prediction as a Negative Class..(.(a "Type II error“)

Classification Model Evaluation

ROC and AUC (Area under Curve): Provide the **sensitivity** and **specificity** values for different threshold values without actually changing the threshold. ROC will help to choosing the threshold that balance the sensitivity and specificity.

AUC is the percentage of the **ROC** plot that is underneath the curve:
AUC is useful as a single number summary of **classifier performance**.
If you randomly chose one positive and one negative observation, AUC represents the likelihood that your classifier will assign a higher predicted probability to the positive observation.
AUC is useful even when there is **high class imbalance** (unlike classification accuracy).



Fraud Transaction Detector (positive class is 'fraud'):
FP : predicting as 'fraud' for 'non-fraud' txn
FN : predicting as 'non-fraud' for fraud txn

In this case **FP**(normal txn that is flagged as possible fraud) would be more acceptable than **FN** (fraudulent txn is flagged as normal txn):
in this case **focus** should be more **reducing the FN** and increasing the **TP** : optimize the sensitivity for higher % .

Spam Filter (positive class is 'spam'): in this case **focus** should be more reducing the **FP** and increasing the **TN .**

Classification Model Evaluation

Sensitivity = TP / (TP + FN)

Sensitivity(True positive Rate) : When the actual value is positive, how often the prediction correct.

Precision = TP / (TP + FP) : When a positive value is predicted, how often is the prediction correct.

How precise is the classifier when predicting positive instances

False Positive Rate = FP/(FP + TN) : When actual value is negative, how often the prediction incorrect.

Specificity(1 - FPR) = **TN/(TN + FP)** : When the actual value is negative, how often the prediction correct .

Regression Tree (Split Decision)

In **regression trees** a typical **impurity measure** is the **sum of the squared deviations from the mean of the leaf**. This is equivalent to the **squared errors.**

The output variable, Y is a **continuous variable** in this case, but both the principle and the procedure are the same:

- Many splits are attempted and, for each, we measure **"impurity"** in each branch of the resulting tree.
- The tree procedure then selects the **split that minimizes the sum** of such measures.

Thus when training a tree, it can be computed **how much each feature** decreases the weighted impurity in a tree.

Squared Errors : is used to measure the **quality of a split**, which is equal to **variance reduction** as a feature selection criteria.

(1) Mean Squared Error (MSE) - L2 Loss (Default value in SKLearn)

(a) Minimize the L2 loss using the mean of each terminal Node
(b) Sensitive to Outliers (Largest value record)

Mean squared error	$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$

(2) Mean Absolute Error (MAE) – L1 Loss

(a) Minimize the L1 loss using the median of each terminal Node
(b) Not sensitive to Outliers (Largest value records)

Mean absolute error	$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$