Ensemble Methods

-MUKESH KUMAR

What is Ensemble Methods

- Ensemble methods combine multiple models to improve predictive performance.
- This approach leverages the strengths of individual models while mitigating their weaknesses, often resulting in better accuracy than any single model could achieve alone.
- Ensemble learning is particularly useful in complex scenarios where a single model may not capture the underlying patterns in the data effectively.

Why Use Ensemble Methods?

- **Improved Accuracy**: By combining multiple models, ensemble methods reduce the likelihood of errors, leading to more accurate and reliable predictions compared to using a single model.
- **Reduced Overfitting**: Ensembles, especially bagging methods like Random Forest, help to reduce overfitting by averaging out the biases of individual models, improving generalization on unseen data.
- **Robustness**: Different models have different strengths and weaknesses. Ensemble methods increase robustness by leveraging diverse models, making the final model more resistant to noise or fluctuations in the data.

Types of Ensemble Methods

There are several types of ensemble methods, but the most common ones are

- **Bagging**
- **Boosting**
- **Stacking**
- **Voting Classifier**

BAGGING

Bagging (Bootstrap Aggregating)

Bagging is short for **Bootstrap Aggregating**. The term comes from the two key components of the technique:

- **Bootstrap**: Refers to creating multiple subsets of the training data by sampling with replacement.
- **Aggregating**: Refers to combining the predictions of each model (e.g., by voting or averaging) to make a final prediction.
- Together, these two steps form the foundation of bagging, which is used to reduce the variance of a model and improve its stability, especially in models prone to overfitting like decision trees.

Bagging (Bootstrap Aggregating)

- Bagging reduces variance and overfitting by training multiple models independently on different subsets of the data.
- The final prediction is made by averaging the predictions (for regression) or taking a majority vote (for classification).

BOOTSTRAP METHOD

What is Bootstrapping?

- **Bootstrapping** is a statistical method that involves creating multiple random samples (called **bootstrap samples**) from your original dataset by **sampling with replacement**.

What does "sampling with replacement" mean?

- When you sample with replacement, each time you pick an item (data point), you put it **back** into the dataset, so it can be selected again. This means that:
- Some data points may appear more than once in a single sample.
- Some data points may not appear at all in that sample.

Steps in Bootstrapping

- **Original Dataset**:
  - Imagine you have a dataset with **N** data points (e.g., 100 rows).
- **Bootstrap Sampling**:
  - You create multiple samples of size **N** by randomly picking data points **with replacement**.
  - Each sample will have **N** data points, but since you are sampling with replacement, some data points will be repeated in the same sample, and others may be missing.
- **Multiple Bootstrap Samples**:
  - Repeat this process several times to get multiple samples (often 100s or 1000s).

BootStrap Example

Lets say we have an original Dataset with 7 records as shown below

S.No	Loves Action Gerne	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
1	Yes	Yes	7	No
2	Yes	No	12	No
3	No	Yes	18	Yes
4	No	Yes	35	Yes
5	Yes	Yes	38	Yes
6	Yes	No	50	No
7	No	No	83	No

BootStrap Example

Original Dataset

S.No	Loves Action Gerne	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
1	Yes	Yes	7	No
2	Yes	No	12	No
3	No	Yes	18	Yes
4	No	Yes	35	Yes
5	Yes	Yes	38	Yes
6	Yes	No	50	No
7	No	No	83	No

BootStrapped Dataset

S.No	Loves Action Gerne	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
1	Yes	Yes	7	No
1	Yes	Yes	7	No
3	No	Yes	18	Yes
4	No	Yes	35	Yes
4	No	Yes	35	Yes
6	Yes	No	50	No
7	No	No	83	No

sampling with replacement

Original Dataset

S.No	Loves Action Gerne	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
1	Yes	Yes	7	No
2	Yes	No	12	No
3	No	Yes	18	Yes
4	No	Yes	35	Yes
5	Yes	Yes	38	Yes
6	Yes	No	50	No
7	No	No	83	No

BootStrapped Dataset

S.No	Loves Action Gerne	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
1	Yes	Yes	7	No
1	Yes	Yes	7	No
3	No	Yes	18	Yes
4	No	Yes	35	Yes
4	No	Yes	35	Yes
6	Yes	No	50	No
7	No	No	83	No

When you sample with replacement, each time you pick an item (data point), you put it **back** into the dataset, so it can be selected again. This means that:

•Some data points may appear more than once in a single sample.

•Some data points may not appear at all in that sample.

BootStrap Example

- Repeating this process , any number of bootstrapped datasets can be created as shown in the next slide

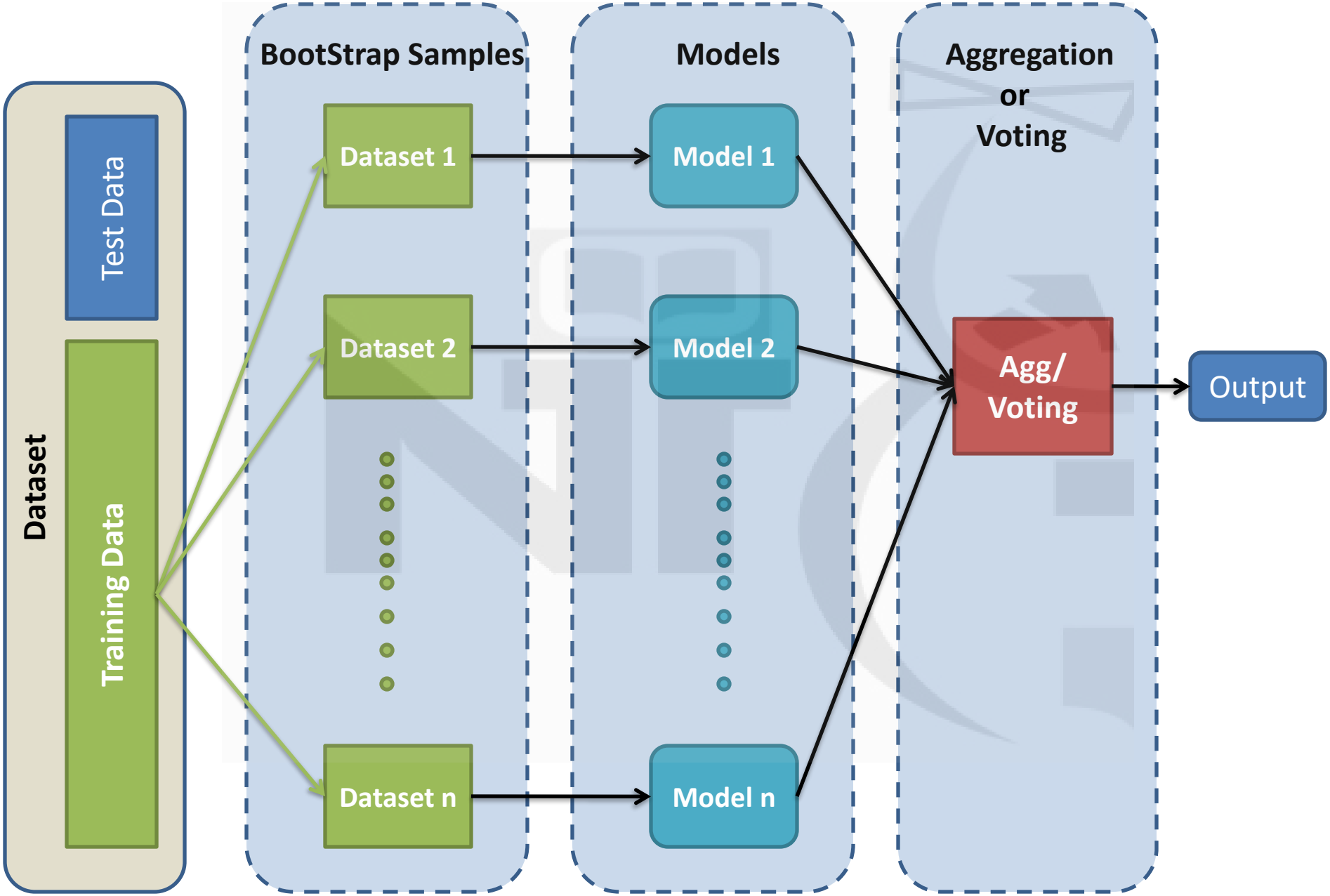
BootStrapped Datasets

S.No	Loves Action Gerne	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
1	Yes	Yes	7	No
1	Yes	Yes	7	No
3	No	Yes	18	Yes
4	No	Yes	35	Yes
4	No	Yes	35	Yes
6	Yes	No	50	No
7	No	No	83	No

S.No	Loves Action Gerne	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
1	Yes	Yes	7	No
2	Yes	No	12	No
3	No	Yes	18	Yes
3	No	Yes	18	Yes
5	Yes	Yes	38	Yes
1	Yes	Yes	7	No
1	Yes	Yes	7	No

S.No	Loves Action Gerne	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
2	Yes	No	12	No
2	Yes	No	12	No
3	No	Yes	18	Yes
4	No	Yes	35	Yes
5	Yes	Yes	38	Yes
5	Yes	Yes	38	Yes
7	No	No	83	No

S.No	Loves Action Gerne	Has Watched Top Gun	Age	Tom Cruise is Fav Actor
7	No	No	83	No
7	No	No	83	No
3	No	Yes	18	Yes
3	No	Yes	18	Yes
5	Yes	Yes	38	Yes
6	Yes	No	50	No
7	No	No	83	No


Dataset
Test Data
Training Data
BootStrap Samples
Dataset 1
Dataset 2
Dataset n
Models
Model 1
Model 2
Model n
Aggregation
or
Voting
Agg/
Voting
Output

Bagging Steps

- **Create multiple datasets**: Randomly sample (with replacement) from the original dataset to create different training datasets (bootstrap samples).
- **Train models**: Train a separate model (e.g., decision tree) on each dataset.
- **Aggregate predictions**: For regression, average the predictions from all models; for classification, use majority voting.

Build a Bagging Classifier

- We can choose any ML model as **estimator**
- All the models in the bagging will be same(homogeneous)

```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

bagging_clf = BaggingClassifier(
    estimator=DecisionTreeClassifier(),
    n_estimators=50,
    random_state=42
)
```

Assignment

- 1_Bagging_Pima.ipynb
  - Try building Bagging classifier with following models
    - SVM
    - Logistic
    - NB
  - Compare models

Demo

- Show how to build a Bagging classifier in Jupyter notebook

Bagging Example

- **Random Forest** is a popular bagging method that creates multiple decision trees and combines their outputs for a more accurate result.

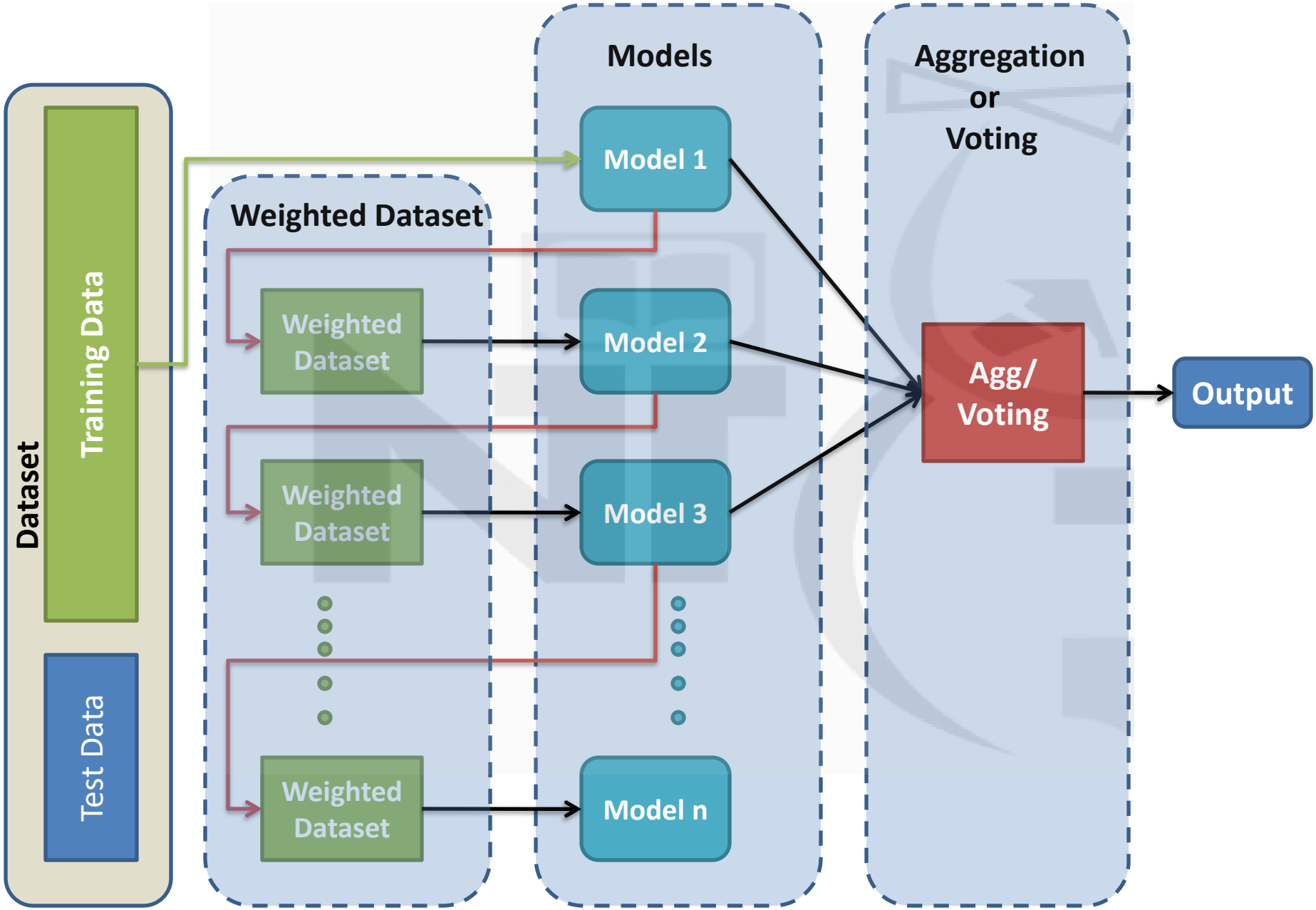
- Explain Random Forest, its in another ppt

BOOSTING

Boosting

- Boosting reduces both bias and variance by **sequentially** training models, where each new model focuses on the mistakes made by the previous ones.

- Models are trained one after the other, and each tries to correct the errors of its predecessor.


Dataset
Training Data
Test Data
Weighted Dataset
Weighted Dataset
Weighted Dataset
Weighted Dataset
Models
Model 1
Model 2
Model 3
Model n
Aggregation or Voting
Agg/ Voting
Output

Boosting Steps

- **Initial model**: Train the first model on the full dataset.
- **Adjust weights**: Increase the weights of misclassified data points, making the next model focus more on those hard-to-classify examples.
- **Train subsequent models**: Each new model is trained on the weighted data to focus on errors made by previous models.
- **Final prediction**: Combine the predictions of all models, often by weighted averaging or voting.

Boosting Example

- **AdaBoost (Adaptive Boosting)**: Focuses on misclassified examples by adjusting their weights.

- **Gradient Boosting**: Sequentially builds models that correct the residuals (errors) of the previous models. **XGBoost** and **LightGBM** are efficient implementations of gradient boosting.

Stacking Vs Bagging/Boosting

- Stacking allows Heterogeneous models
- Meta model

Stacking

- Stacking combines multiple models of different types.
- **Base Models (Level-0 Models)**: Multiple models of different types (e.g., decision trees, SVMs, etc.) are trained on the same dataset independently.
- **Meta-Model (Level-1 Model)**: The predictions of the base models serve as inputs to a meta-model. This model learns how to combine the base model outputs to improve overall performance.
- **Final Prediction**: The meta-model outputs the final prediction by combining the individual base model predictions in an optimal way, enhancing the ensemble's accuracy.

- The final model (level-1) learns to correct the errors made by the base models based on their predictions. It can combine their outputs in a way that minimizes the overall error on the validation set.

Dataset
Test Data
Training Data
Base Models
Model 1
Model 2
Model n
New Training Data
Meta Model
Final Prediction

Stacking Steps

- **Train base models**: Train several different models (e.g., logistic regression, decision trees, SVM) on the same dataset.
- **Meta-model**: Use the predictions of the base models as inputs for the meta-model, which then makes the final prediction.

Stacking Example

- A stacking ensemble might combine predictions from a **decision tree, a support vector machine, and a neural network**, with a **logistic regression** model as the meta-model to combine their outputs.

Stacking Vs Blending

- Stacking uses k-fold approach for training Base models

- Blending uses train_test_split approach for training base models

Steps in Stacking with k-Fold Cross-Validation

1. **Split the Data into k Folds:**
   - The data is divided into **k equal parts** (folds). For example, if **k = 5**, you split the data into 5 folds.
2. **Train Base Models on k-1 Folds:**
   - For each fold (iteration), base models are trained on **k-1 folds** of the data.
   - For example, if you're on the first fold, you train the model on folds 2 to 5, and keep fold 1 for validation.

Steps in Stacking with k-Fold Cross-Validation

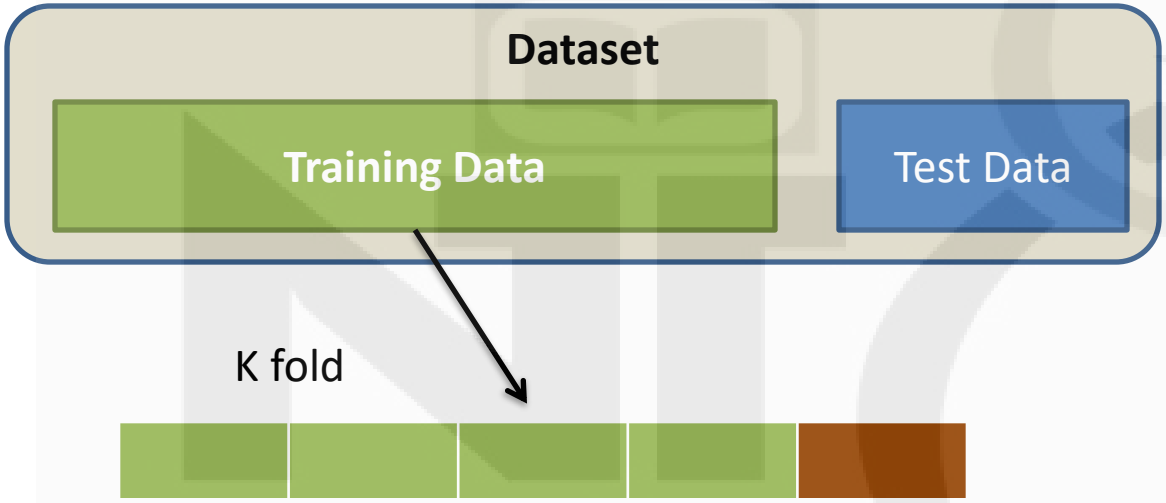
3. **Generate Out-of-Fold Predictions:**
   - The trained base models make predictions on the **left-out fold** (the 1 fold that wasn't used for training). These predictions are called **out-of-fold predictions**.
   - Repeat this process **k times**, so every fold has its out-of-fold predictions.
4. **Collect Out-of-Fold Predictions:**
   - Once all folds have been used for training and prediction, you collect all the out-of-fold predictions. This gives you a complete set of predictions for the entire dataset (since each data point will have been in a left-out fold once).

Steps in Stacking with k-Fold Cross-Validation

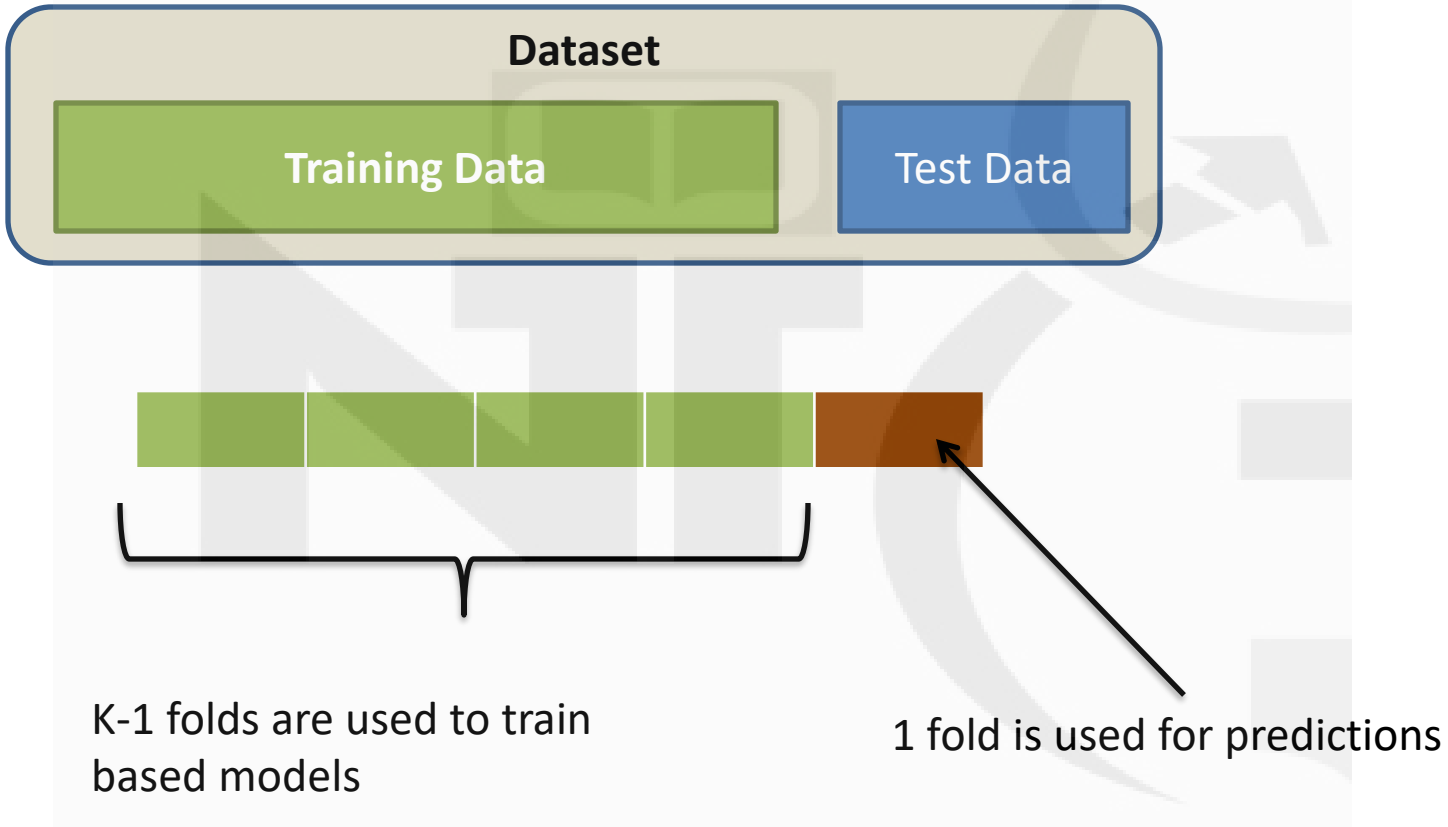
5. **Train the meta model:**
   - The **meta model** (stacking model) is then trained on these out-of-fold predictions (not the original features), learning how to combine the predictions of the base models.

6. **Final Prediction on Test Set:**
   - After the meta model is trained, the base models are retrained on the **entire dataset**, and their predictions on the test data are combined by the meta model to make the final prediction.

Stacking



Stacking


Dataset
Training Data
Test Data
K-1 folds are used to train based models
1 fold is used for predictions

If there are 3 base models predictions will look like this:

K-1 folds are used to train based models

1 fold is used for predictions

y_pred1	ypred2	ypred3
1	0	1
0	1	1
1	1	1
1	0	0
0	0	0
1	1	1

Meta model input data will look like this

y_pred1	ypred2	ypred3	yactual
1	0	1	1
0	1	1	0
1	1	1	0
1	0	0	1
0	0	0	1
1	1	1	1

Yactual is the output label and rest are feature columns,

Meta model is trained on this data and generates final predictions

finally like any other SL we can evaluate this model on TestData

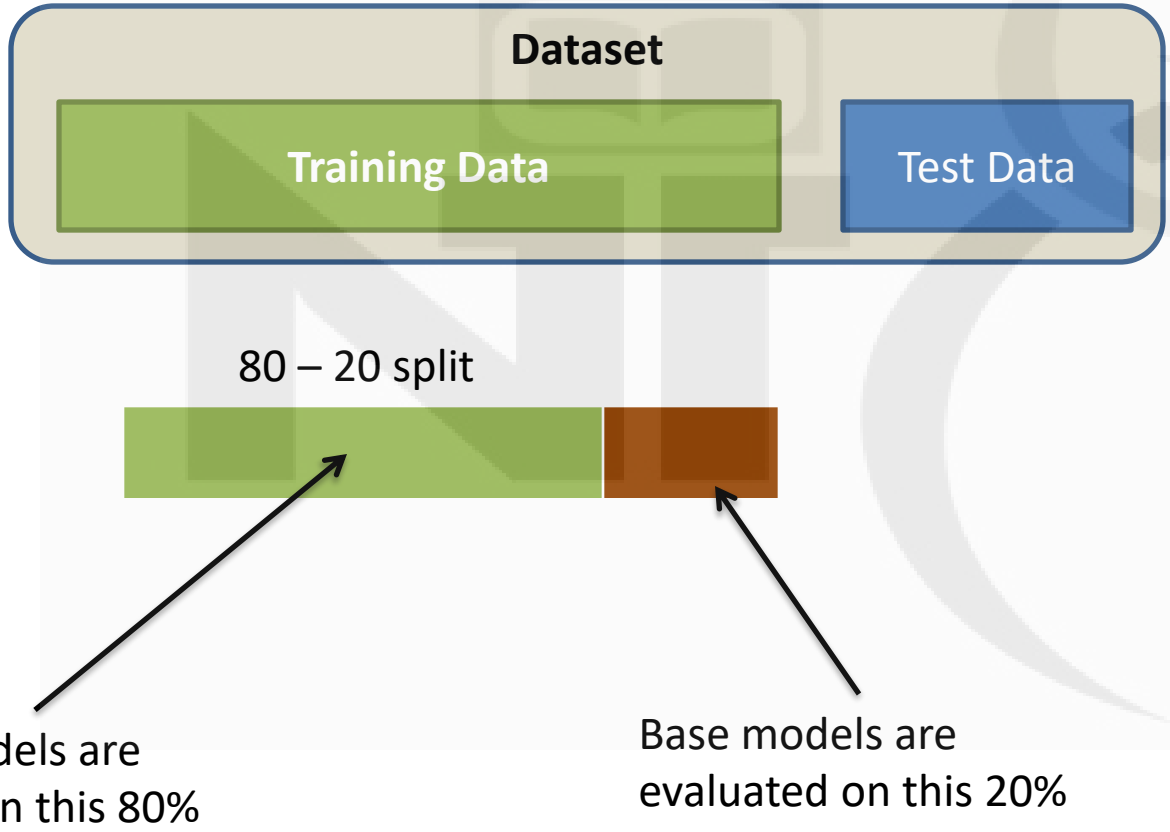
Blending

1. **Data Split:**
   - You split the dataset into two parts:
   - A **training set** (e.g., 90% of the data).
   - A **validation set** (e.g., 10% of the data).
2. **Train Base Models:**
   - Train each of your base models (e.g., random forest, logistic regression, etc.) on the **training set** (90%).

3. **Generate Predictions for Validation Set:**
   - Use the trained base models to make predictions on the **validation set** (the hold-out 10%).
4. **Train Meta Model:**
   - The predictions from the base models on the validation set are used as input features to train the **meta model** (blender).

5. **Final Predictions:**
   - The meta model is then used to make final predictions on new test data, based on predictions from the base models.

Blending


Dataset
Training Data
Test Data
80 – 20 split
Base models are trained on this 80%
Base models are evaluated on this 20%

Commonly Used Algorithms in Ensemble Methods

- **Random Forest**: Uses bagging with decision trees, averaging their results for regression or taking a vote for classification.
- **XGBoost/LightGBM**: Gradient boosting algorithms that are highly optimized and scalable, frequently winning machine learning competitions.
- **AdaBoost**: Focuses on misclassified instances by reweighting them.
- **Voting Classifier**: A simple method that combines predictions from multiple models using majority voting for classification or averaging for regression.

Advantages of Ensemble Methods:

- **Better performance**: They generally outperform individual models, especially in complex tasks.
- **Reduced overfitting**: Bagging, in particular, helps to reduce overfitting by reducing the model variance.
- **Robustness**: They are less sensitive to the quirks of individual models or outliers

Disadvantages

- **Increased complexity**: Ensembles are harder to interpret than single models, especially when using boosting or stacking.
- **Longer training times**: Training multiple models can take significantly longer.
- **Overfitting risk**: Some boosting methods can overfit if not carefully tuned (especially when the base learners are too complex).

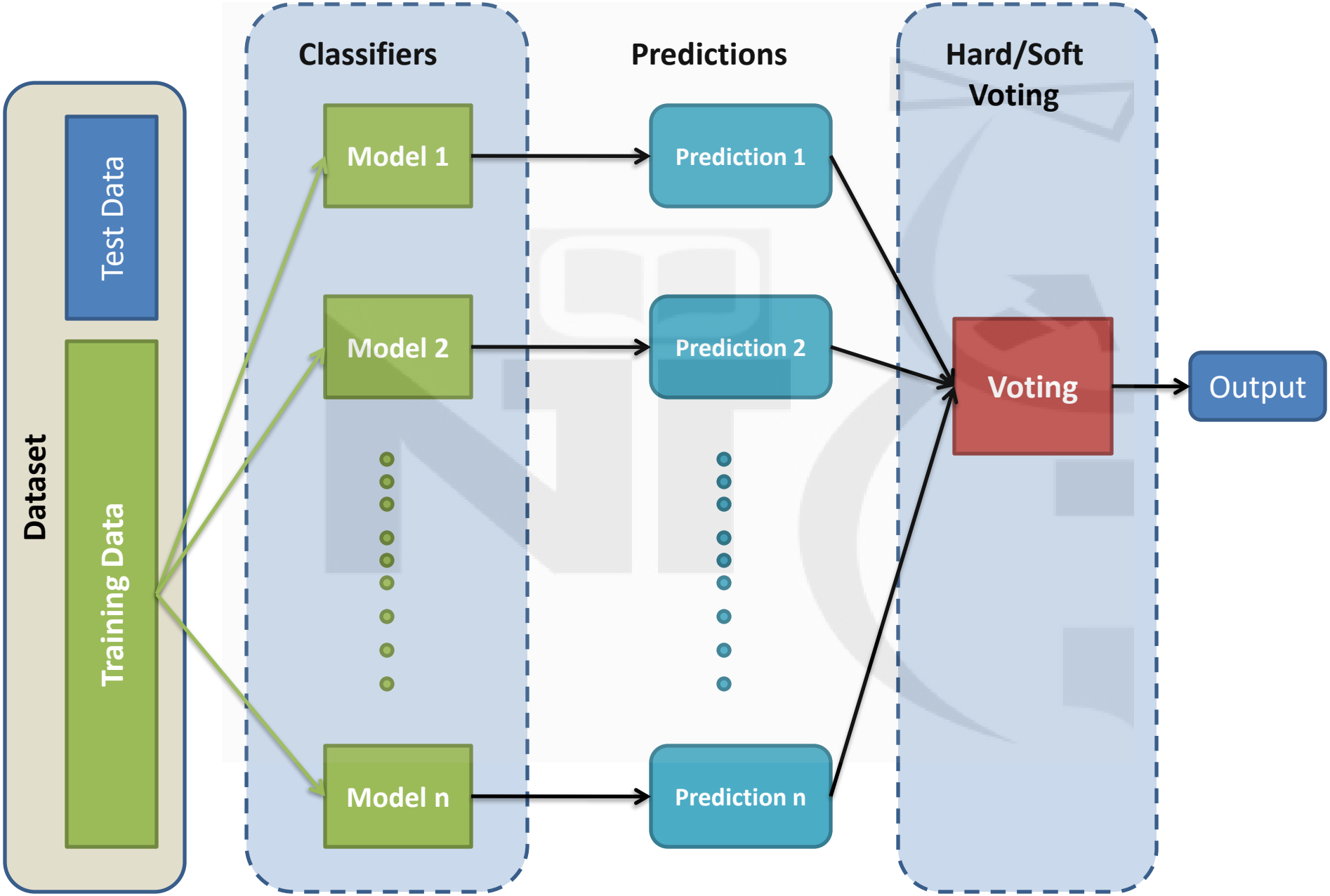
Practical Implementations of Ensemble Methods

- https://blogs.sas.com/content/subconsciousmusings/2017/05/18/stacked-ensemble-models-win-data-science-competitions/

Voting Classifier

- A **Voting Classifier** is an ensemble learning technique that combines the predictions of multiple classifiers to make a final decision

- The main idea is that by combining several different models, the ensemble can improve the overall prediction accuracy by leveraging the strengths of each individual model


Classifiers
Predictions
Hard/Soft Voting
Dataset
Test Data
Training Data
Model 1
Model 2
Model n
Prediction 1
Prediction 2
Prediction n
Voting
Output

Types of Voting Classifiers

Hard Voting (Mojority Rule):

- The classifier assigns a class label based on the majority voting of the individual classifiers.
- Each model in the ensemble casts a vote for a class, and the class with the most votes becomes the final prediction.

- **Soft Voting (Average of Probabilities)**:
  - Instead of casting votes for the class labels, each model in the ensemble predicts the class probabilities.
  - The class probabilities from all the models are averaged, and the class with the highest average probability is selected as the final prediction.

Hard Voting Example

- Suppose we have 3 classifiers:
  - Classifier 1 predicts Class A,
  - Classifier 2 predicts Class B,
  - Classifier 3 predicts Class A.

- With **hard voting**,:
  - the final prediction will be **Class A** because it gets the **majority** of votes (2 votes for Class A and 1 vote for ClassB).

Soft Voting Example

- Suppose we have 3 classifiers:
  - Classifier 1 predicts probabilities: Class A: 0.7, Class B: 0.3
  - Classifier 2 predicts probabilities: Class A: 0.4, Class B: 0.6
  - Classifier 3 predicts probabilities: Class A: 0.8, Class B: 0.2

- The **soft voting** approach would average the probabilities for each class:
  - For Class A: (0.7 + 0.4 + 0.8) / 3 = 0.63
  - For Class B: (0.3 + 0.6 + 0.2) / 3 = 0.37
- Since **Class A** has the higher average probability (0.63), the final prediction will be **Class A**.

Voting Classifier

- Implementation in 1_PFPima.ipynb notebook

Assignment

- Heart Disease dataset:
  - Build all the SL models
  - Apply bagging, RF
  - Apply boosting: ada, gradient, xg, lightgbm
  - Apply stacking
  - Apply voting classifier
- Compare the performance of all the models