Ensemble Methods: Elegant Techniques to Produce Improved Machine Learning Results

Previous post
Next post

Tags: Boosting, Ensemble methods, Machine Learning, Python

Get a handle on ensemble methods from voting and weighting to stacking and boosting, with this well-written overview that includes numerous Python-style pseudocode examples for reinforcement.

By Necati Demir.

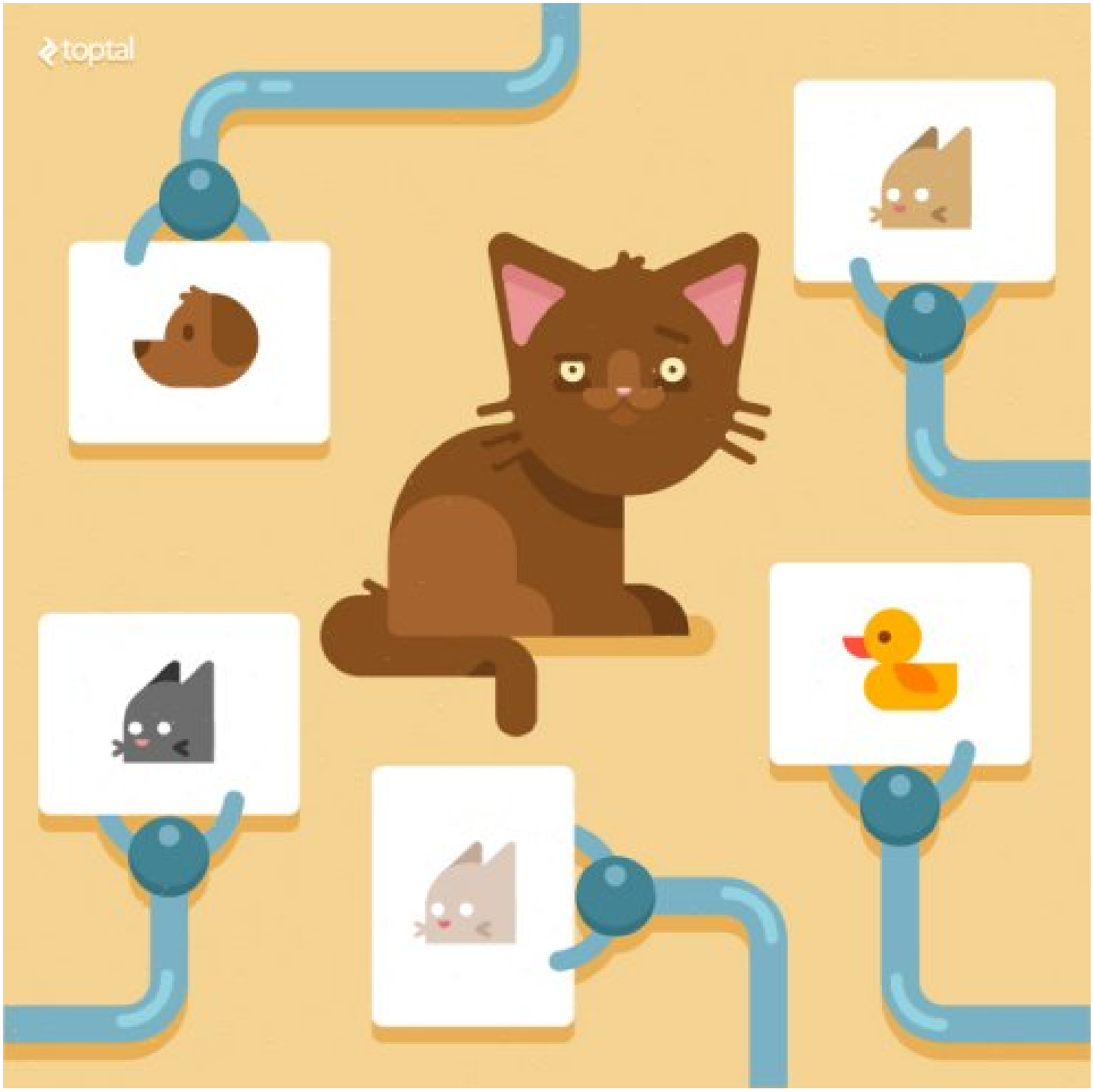
Ensemble methods are techniques that create multiple models and then combine them to produce improved results. Ensemble methods usually produces more accurate solutions than a single model would. This has been the case in a number of machine learning competitions, where the winning solutions used ensemble methods. In the popular Netflix Competition, the winner used an ensemble method to implement a powerful collaborative filtering algorithm. Another example is KDD 2009 where the winner also used ensemble methods. You can also find winners who used these methods in Kaggle competitions, for example here is the interview with the winner of CrowdFlower competition.

It is important that we understand a few terminologies before we continue with this article. Throughout the article I used the term “model” to describe the output of the algorithm that trained with data. This model is then used for making predictions. This algorithm can be any machine learning algorithm such as logistic regression, decision tree, etc. These models, when used as inputs of ensemble methods, are called ”base models”.

In this blog post I will cover ensemble methods for classification and describe some widely known methods of ensemble: voting, stacking, bagging and boosting.

Voting and Averaging Based Ensemble Methods

Voting and averaging are two of the easiest ensemble methods. They are both easy to understand and implement. Voting is used for classification and averaging is used for regression.



In both methods, the first step is to create multiple classification/regression models using some training dataset. Each base model can be created using different splits of the same training dataset and same algorithm, or using the same dataset with different algorithms, or any other method. The following Python-esque pseudocode shows the use of same training dataset with different algorithms.

```
train = load_csv("train.csv")
target = train["target"]
train = train.drop("target")
test = load_csv("test.csv")

algorithms = [logistic_regression, decision_tree_classification, ...] #for classification
algorithms = [linear_regression, decision_tree_regressor, ...] #for regression

predictions = matrix(row_length=len(target), column_length=len(algorithms))

for i,algorithm in enumerate(algorithms):
    predictions[,i] = algorithm.fit(train, target).predict(test)
```

According to the above pseudocode, we created predictions for each model and saved them in a matrix called predictions where each column contains predictions from one model.

Majority Voting

Every model makes a prediction (votes) for each test instance and the final output prediction is the one that receives more than half of the votes. If none of the predictions get more than half of the votes, we may say that the ensemble method could not make a stable prediction for this instance. Although this is a widely used technique, you may try the most voted prediction (even if that is less than half of the votes) as the final prediction. In some articles, you may see this method being called "plurality voting".

Weighted Voting

Unlike majority voting, where each model has the same rights, we can increase the importance of one or more models. In weighted voting you count the prediction of the better models multiple times. Finding a reasonable set of weights is up to you.

Simple Averaging

In simple averaging method, for every instance of test dataset, the average predictions are calculated. This method often reduces overfit and creates a smoother regression model. The following pseudocode code shows this simple averaging method:

```
final_predictions = []
for row_number in len(predictions):
    final_predictions.append(
        mean(prediction[row_number, ])
    )
```

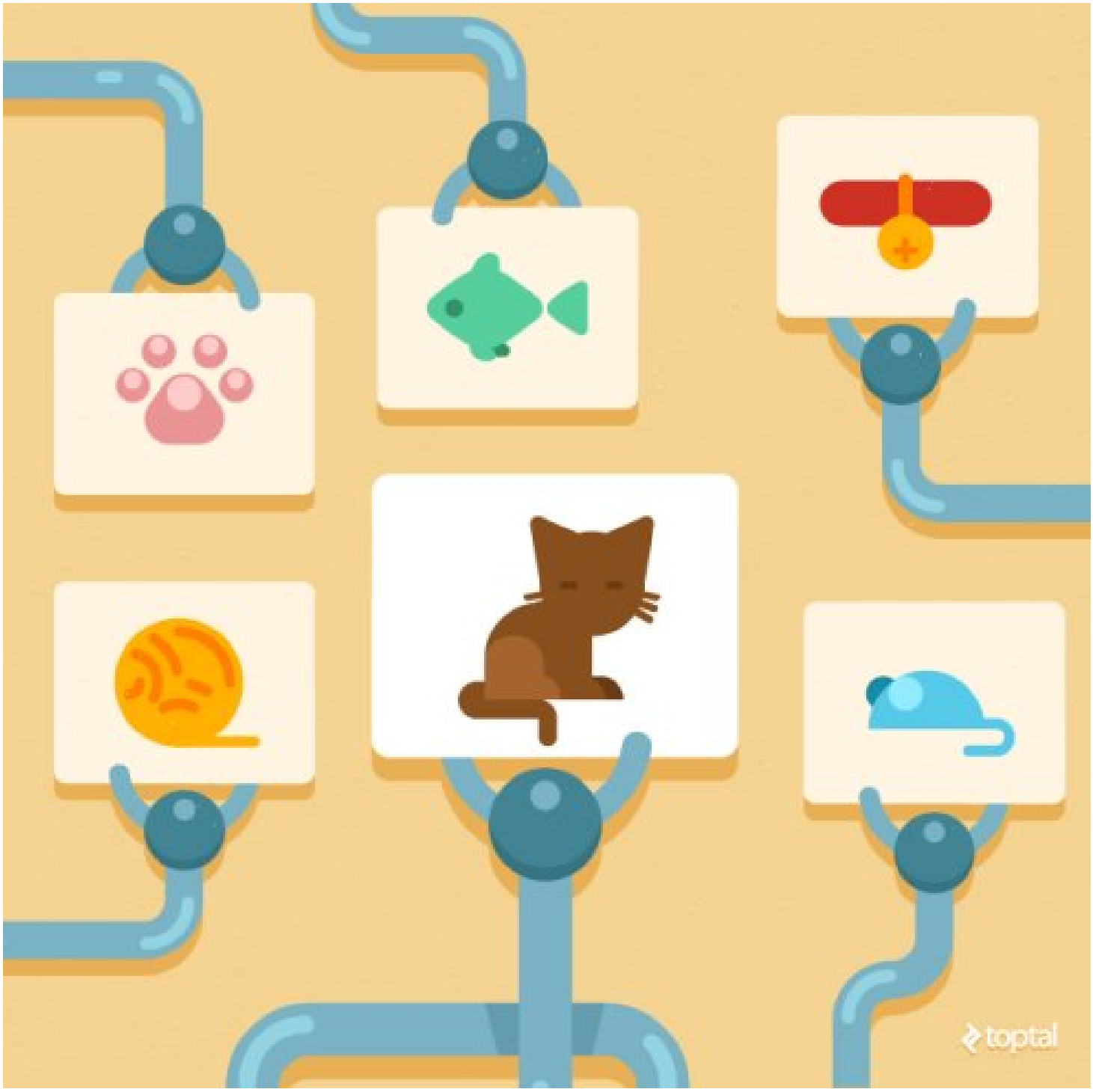
Weighted Averaging

Weighted averaging is a slightly modified version of simple averaging, where the prediction of each model is multiplied by the weight and then their average is calculated. The following pseudocode code shows the weighted averaging:

```
weights = [..., ..., ...] #length is equal to len(algorithms)
final_predictions = []
for row_number in len(predictions):
    final_predictions.append(
        mean(prediction[row_number, ]*weights)
    )
```

Stacking Multiple Machine Learning Models

Stacking, also known as stacked generalization, is an ensemble method where the models are combined using another machine learning algorithm. The basic idea is to train machine learning algorithms with training dataset and then generate a new dataset with these models. Then this new dataset is used as input for the combiner machine learning algorithm.



The pseudocode of a stacking procedure is summarized as below:

```
base_algorithms = [logistic_regression, decision_tree_classification, ...] #for classification

stacking_train_dataset = matrix(row_length=len(target), column_length=len(algorithms))
stacking_test_dataset = matrix(row_length=len(test), column_length=len(algorithms))


for i,base_algorithm in enumerate(base_algorithms):
    stacking_train_dataset[,i] = base_algorithm.fit(train, target).predict(train)
    stacking_test_dataset[,i] = base_algorithm.predict(test)

final_predictions = combiner_algorithm.fit(stacking_train_dataset, target).predict(stacking_test_dataset)
```

As you can see in the above pseudocode, the training dataset for combiner algorithm is generated using the outputs of the base algorithms. In the

pseudocode, the base algorithm is generated using training dataset and then the same dataset is used again to make predictions. But as we know, in the real world we do not use the same training dataset for prediction, so to overcome this problem you may see some implementations of stacking where training dataset is splitted. Below you can see a pseudocode where the training dataset is split before training the base algorithms:

```
base_algorithms = [logistic_regression, decision_tree_classification, ...] #for classification

stacking_train_dataset = matrix(row_length=len(target), column_length=len(algorithms))
stacking_test_dataset = matrix(row_length=len(test), column_length=len(algorithms))

for i,base_algorithm in enumerate(base_algorithms):
    for trainix, testix in split(train, k=10): #you may use sklearn.cross_validation.KFold of sklearn library
        stacking_train_dataset[testcv,i] = base_algorithm.fit(train[trainix], target[trainix]).predict(train[testix])
stacking_test_dataset[,i] = base_algorithm.fit(train).predict(test)


final_predictions = combiner_algorithm.fit(stacking_train_dataset, target).predict(stacking_test_dataset)
```

Pages: 1 2

Previous post
Next post

Top Stories Past 30 Days

Most Popular

1. The 10 Algorithms Machine Learning Engineers Need to Know
2. 7 More Steps to Mastering Machine Learning With Python
3. 50 Companies Leading The AI Revolution, Detailed
4. Every Intro to Data Science Course on the Internet, Ranked
5. Gartner 2017 Magic Quadrant for Data Science Platforms: gainers and losers
6. 17 More Must-Know Data Science Interview Questions and Answers
7. An Overview of Python Deep Learning Frameworks

Most Shared

1. 50 Companies Leading The AI Revolution, Detailed
2. What Is Data Science, and What Does a Data Scientist Do?
3. Getting Started with Deep Learning
4. Getting Up Close and Personal with Algorithms
5. What makes a good data visualization – a Data Scientist perspective
6. 17 More Must-Know Data Science Interview Questions and Answers, Part 3
7. 7 Types of Data Scientist Job Profiles

Latest News

- Top tweets, Mar 29 – Apr 04: Free Must-Read Boo...
- Join 350+ Insurance executives at Insurance A...
- Make Analytics Pay with Live Immersive Training.
- Red Ventures: Data Scientist
- New James Bond is a Data Scientist: Data Scie...
- Samsung: Senior Data Scientist – Device Ins...

Top Stories
Last Week

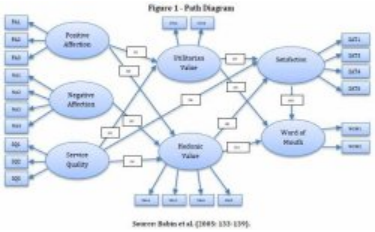
Most Popular

1. The 10 Algorithms Machine Learning Engineers Need to Know
2. NEW Standardization and Specialization in Analytics, Data Science, and BI

3. NEW The Best R Packages for Machine Learning
4. NEW What makes a great data scientist?
5. NEW Getting Started with Deep Learning
6. NEW What Is Data Science, and What Does a Data Scientist Do?
7. NEW 50 Companies Leading The AI Revolution, Detailed

Most Shared

1. What is Structural Equation Modeling?



2. The Best R Packages for Machine Learning
3. What makes a great data scientist?
4. Deep Stubborn Networks - A Breakthrough Advance Towards Adversarial Machine Intelligence
5. A Short Guide to Navigating the Jupyter Ecosystem
6. Standardization and Specialization in Analytics, Data Science, and BI
7. From Big Data Platforms to Platform-less Machine Learning

More Recent Stories

- Samsung: Senior Data Scientist – Device Insights
- Top March Stories: 7 More Steps to Mastering Machine Learning ...
- Putting Together A Full-Blooded AI Maturity Model
- Samsung: Senior Data Scientist – Strategic Analytics
- Does the Muslim Ban Make Us Safer?: Data Science vs Fake News
- KDnuggets 17:n13, Apr 5: What makes a great data scientist?...
- University of San Francisco: Assistant Professor, Master of Sc...
- Finding "Gems" in Big Data
- Upcoming Meetings in Analytics, Big Data, Data Science, Machin...
- Top /r/MachineLearning Posts, March: A Super Harsh Guide to Ma...
- Must-Know: Why it may be better to have fewer predictors in Ma...
- Introduction to Anomaly Detection
- Wharton: Successful Applications of Customer Analytics, May 10...
- Beware of Two Data Obfuscation Tactics
- What is AI? Ingredients for Intelligence
- Strata London: Learn about all things data. KDnuggets Offer en...
- PAW Healthcare – Apply to speak!
- Top Stories, Mar 27-Apr 2: Standardization and Specialization ...
- Deep Stubborn Networks – A Breakthrough Advance Towards Adve...
- A Short Guide to Navigating the Jupyter Ecosystem

KDnuggets Home » News » 2016 » Feb » Tutorials, Overviews » Ensemble Methods: Elegant Techniques to Produce Improved Machine Learning Results (16:n06)