XgBoost

Michael Byrd

September 8, 2025

1 Type of AI models

Supervised Models Classification	**Supervised Models Regression**
Logistic Regression	Linear Regression
Support Vector Machine (SVM)	Polynomial Regression
Decision Tree	Decision Tree Regression
Random Forest	Random Forest Regression
K-Nearest Neighbors (KNN)	Support Vector Regression (SVR)

Unsupervised Models	**Semi-Supervised Models**	**Reinforcement learning Models**
Clustering	Generative Semi-Supervised Learning	Value-based learning
Dimensionality Reduction	Graph-based Semi-Supervised Learning	Policy-based learning
Anomaly Detection		

Deep Learning
Artificial Neural Networks (ANNs)
Convolutional Neural Networks (CNNs)
Recurrent Neural Networks (RNNs)
Long Short-Term Memory Networks (LSTMs)

2 XGBoost

XGBoost (1) uses decision trees as its base learners combining them sequentially to improve the model's performance. Each new tree is trained to correct the errors made by the previous tree and this process is called boosting

2.1 How does XGBoost work

1 Start with a base learner: Which in regression tasks this base model simply predict the average of the target variable.

2 Calculate the errors: After training the first tree the errors between the predicted and actual values are calculated.

3 Train the next tree: The next tree is trained on the errors of the previous tree. This step attempts to correct the errors made by the first tree.

4 Repeat the process: This process continues with each new tree trying to correct the errors of the previous trees until a stopping criterion is met.

5 Combine the predictions: The final prediction is the sum of the predictions from all the trees.

XGBoost Object Function is the loss function discussed below.

3 Tree Boosting

3.1 Boosted Decision Tree

3.1.1 Terminology

- Root Node - the starting point that represents the entire dataset.
- Branches - are the lines that connect nodes. It shows the flow from one decision to another.
- Internal Nodes - are points where decisions are made based on the input features
- Leaf Nodes - are the terminal nodes at the end of branches that represent final outcomes or predictions

3.1.2 Boosting

Is adding weak predictions together to create a stronger prediction.

A weak prediction is a tree that does not meet the expected accuracy.

If we take serval weak tree and have them "vote" on the right solution then we should get a better results from all of the trees compare to just any one tree.

We can focus on the wrong predictions from the previous tree(s) to give to the next tree to straighten the overall predictions.

For the third tree we give the data that first and second tree does not agree on, and then repeat the process.

3.1.3 Classification

Now let's look at the equations behind the process.

Loss Function: $F_m(x) = arg \min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma)$

For a history on the loss function please see ref Cramer 2002 (2)

Some notes on Loss function

- Creates the sigmoid curve which behaviors well on (0,1)
- Created by Pierre-Francois Verhulst
- Created to model growth

Notation y_i is actual value for the i^{th} observation (row)

m is the number of tree.

x_i is the row of data use to predict outcome

n is the number of rows in a leaf

F_{m-1} is the last model prediction

When $m = 0$ then $F_{m-1}(x_i) = 0$

Derivative of Loss Function: $-\sum_{i=1}^{n} y_i * \log(p) + (1 - y_i) * \log(1 - p)$

Where p is the probability of selecting a correct classification. In our example it will be if an animal can swim.

Note the larger the log(likelihood) the worst the prediction, so we want to make it smaller thus multiple by -1.

Log of the odds: $\log(\frac{p}{1-p})$

To make $\log(1 - p)$ into a function of the log(odds) we get

$$-\log(1 + e^{\log(odds)}) \tag{1}$$

Then replacing $\log(1 - p)$ with the function of the log(odds) we get

$$\sum_{i=1}^{N} \text{Observed} * \log(odds) - \log(1 + e^{\log(odds)}) \tag{2}$$

Steps in creating prediction model

Step 1

We need to find $\frac{\partial}{\partial \log(oods)}$

$$\frac{\partial}{\partial \log(oods)} = \text{Observed} - \frac{e^{\log(odds)}}{1+e^{\log(odds)}} \tag{3}$$

We can write the above as Observed $- p$

To find first prediction we use $F_0(x) = \arg\min \sum_{i=1}^{n} L(y_i, \gamma)$

To find the min use $\frac{\partial}{\partial \log(oods)} \sum_{i=1}^{n} L(y_i, \gamma) = 0$

Note we will get probability then plug it into the $\log(odds)$

Step 2

Calculate the next prediction $m + 1$

Compute the Pseudo Residual $r_{i,m} = -[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}]_{F(x)=F_{m-1}(x)} = (\text{Observed} - p)$

Fit a regression tree to the $r_{i,m}$ values and create terminal regions $R_{j,m}$, for $j = 1, ..., J_d$

Where J_d are the Leaf Nodes of a tree.

Reading the output of a trees.

Calculate the output values for each terminal regions j using

$\gamma_{j,m} = \arg\min \sum_{x_i \in R_{i,j}}^{n} L(y_i, F_{m-1}(x_i) + \gamma)$

To find the minimal function we need to derived with respect to γ

We use a second order Taylor Polynomial

$L(y_i, F_{m-1}(x_i)+\gamma) \approx L(y_i, F_{m-1}(x_i))+\frac{\partial}{\partial F(x_0)}(y_i, F_{m-1}(x_i))\gamma+\frac{1}{2}\frac{\partial^2}{\partial F(x_0)^2}(y_i, F_{m-1}(x_i))\gamma^2$

Let's take the derivative with respect to γ of the above function

$\frac{\partial}{\partial \gamma} L(y_i, F_{m-1}(x_i) + \gamma) \approx \frac{\partial}{\partial F(x_0)}(y_i, F_{m-1}(x_i)) + \frac{\partial^2}{\partial F(x_0)^2}(y_i, F_{m-1}(x_i))\gamma$

Solve for γ

$\gamma = \frac{-\frac{\partial}{\partial F(x_0)}(y_i, F_{m-1}(x_i))}{\frac{\partial^2}{\partial F(x_0)^2}(y_i, F_{m-1}(x_i))}$

Note that $-\frac{\partial}{\partial F(x_0)}(y_i, F_{m-1}(x_i)) = \text{Observed} - \frac{e^{\log(odds)}}{1+e^{\log(odds)}} = \text{Observed} - p =$Pseudo Residual

Now to find $\frac{\partial^2}{\partial \log(odds)^2}$ of $-\text{Observed} * \log(odds) - \log(1 + e^{\log(odds)})$

$$p(1-p) \tag{4}$$

Then $\gamma_{i,m} = \frac{r_{i,m}}{p(1-p)}$

Now calculate $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{j,m} y_{j,m} \mid x \in R_{j,m}$ and ν is learning rate.

Note that $\sum_{j=1}^{j,m}$ is there in the case a sample is in multiple leaves.

Repeat for each row and we now have our tree Pseudo Residual $F_m(x_i)$

Then repeat step 2 for till m (Max number of trees)

3.1.4 Classification Example

We used three significant figures because of the data collected.

We also a learning rate of .1 for the boosting.

Animal Name	Weight KG	Height cm	Number of Walking limbs	Can Fly	Can Swim	Can Swim Text
Giraffe	1192	550	4	0	0	No
Golden Retriever	32	61	4	0	1	Yes
Gorillas (Male)	136.1	180	2	0	0	No
House Cat	4.53	24	4	0	1	Yes
Human (Male)	90.7	171	2	0	1	Yes
Indian Peafowl	4.7	200	2	1	0	No
Mallard	1.59	38.5	2	1	1	Yes
Northern Mockingbird	0.0482	24	2	1	0	No
Orangutan (Male)	86.6	137	2	0	1	Yes

Data used for prediction.

Animal Name	p_0	F_0	$Residual_0$	Output for m_1	F_1	p_1	$Residual_1$	Output for m_2	F_2
Giraffe	0.555	0.097	-0.0969	-0.559	0.0411	0.51	-0.51	-0.603	-0.0192
Golden Retriever	0.555	0.097	0.903	5.211	0.6181	0.65	0.35	-0.603	0.5578
Gorillas (Male)	0.555	0.097	-0.0969	-0.559	0.0411	0.51	-0.51	-0.603	-0.0192
House Cat	0.555	0.097	0.903	5.211	0.6181	0.65	0.35	-0.603	0.5578
Human (Male)	0.555	0.097	0.903	5.211	0.6181	0.65	0.35	-0.603	0.5578
Indian Peafowl	0.555	0.097	-0.0969	-0.559	0.0411	0.51	-0.51	-0.908	-0.0497
Mallard	0.555	0.097	0.903	5.211	0.6181	0.65	0.35	-0.908	0.5273
Northern Mockingbird	0.555	0.097	-0.0969	-0.559	0.0411	0.51	-0.51	-0.908	-0.0497
Orangutan (Male)	0.555	0.097	0.903	5.211	0.6181	0.65	0.35	-0.603	0.5578

Calculated numbers. The below tree was created using sklearn.tree and matplotlib.pyplot using python

Note: The python code is in the work explain section.

Below is the process to find the Predictions for Giraffe

Start by finding the mean log(probability) for the population

$$\log(\frac{5}{4}) = .0969$$

we then find the probability F_0 by using the Logistic Function

$$F_0 = \frac{e^{.0969}}{1 + e^{.0969}} = .5242$$

Then calculate the Residual for each below is the the residual for the Giraffe

$$\gamma = y_i - F_0 = 0 - 0.5242 = -0.5242$$

We then create the first tree output $m = 1$ and calculate the out come.

Noting that the Indian Peafowl and Gorillas (Male) are in the leaf with the Giraffe.

$$\frac{-.5242 + (-.5242) + (-.5242)}{(.5242(1 - .5242) + .5242(1 - .5242) + .5242(1 - .5242))} = -2.1017$$

Calculate $F_1 = F_0 + .1*$ output of m_1 (above)

$$F_1 = -0.5242 + .1 * -2.1017 = -.314$$

Convert the above number to probability into a function of the log(odds)

$$\frac{e^{.314}}{1 + e^{.314}} = .5779$$

Calculate the γ for the new prediction and we get

$$\gamma = 0 - .51 = -0.5779$$

Now increase $m = 2$ and repeat the above steps

We then create the first tree output $m = 2$ and calculate the out come.

Noting that the House Cat and Golden Retriever are in the leaf with the Giraffe.

$$\frac{.3586 + .3586 + .(-.6101)}{(.6414(1 - .6414) + .6414(1 - .6414) + .6101(1 - .6101)} = 0.1535$$

Calculate $F_2 = F_1 + .1*$ output of m_2 (above)

$$F_2 = -0.5242 + .1 * -2.1017 + .1 * 1535 = 0.463$$

And to find the prediction probability that an Giraffe can't swim is

$$\frac{e^{.463}}{1 + e^{.463} = .6137}$$

The below tress was created by the python code seen in Work Explained.

The first tree used the entire data set, the second tree we removed Height and Weight to get a new tree.

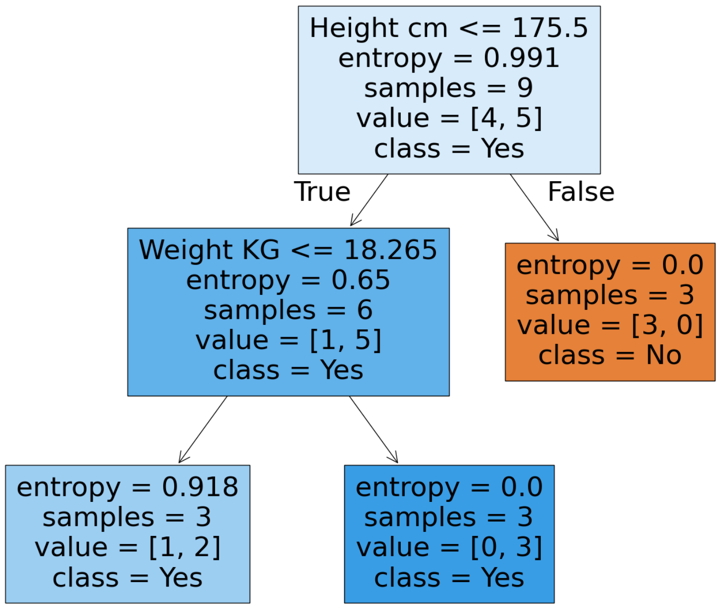

Figure 1: Decision Tree 1

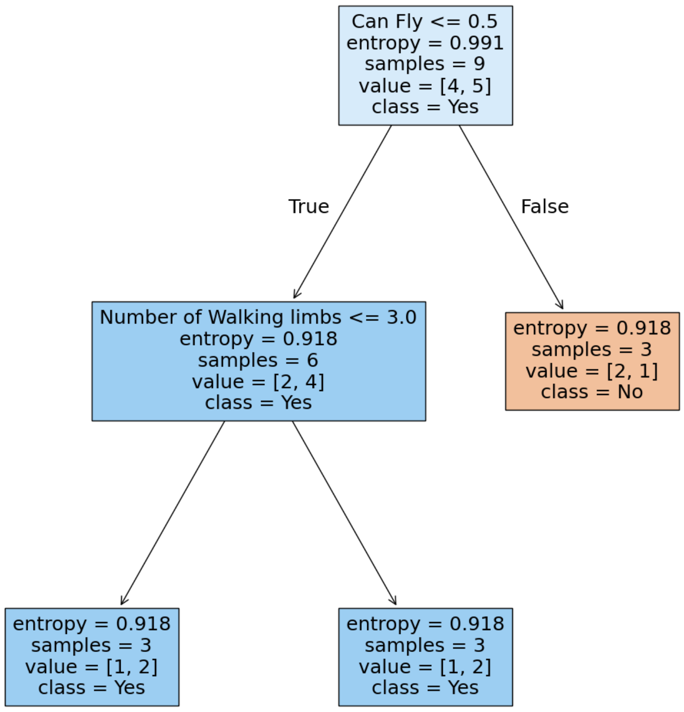

Figure 2: Decision Tree 2

3.1.5 Regression: HERE FOR REFERENCE WILL NOT BE INCLUDED IN TALK UNLESS ASKED

Use for prediction on a continuous number line.

$F_0(x) = \arg\min\limits_{\gamma} \sum\limits_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma)$

$L = \frac{1}{2} \sum\limits_{i=1}^{n} (|y_i - \hat{y}_i|)^2$ the Mean Absolute Error function

Note on $\frac{1}{2}$ to make calculation easier and can be done since the minima of L and $\frac{1}{2}L$ at the same place.

Noting this will be the average of the column that we want to predict.

y_i is actual value for the i^{th} observation (row)

$\hat{y}_i$ is calculated value for the i^{th} observation

To find the min we use $\partial L = -\sum\limits_{i=0}^{n} (y_i - \hat{y}_i)$

Pseudo Residual: $r_{i,m} = -[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}]_{F(x)=F_{m-1}(x)}$ for $i = 1, .., m,$

Because of the L we pick $\gamma = (y_i - \hat{y}_i)$

$F(x_i)$ is the previous model prediction and m is the index of trees (M)

Let y_i be the observed value then $\frac{\partial L}{\partial \lambda} = (\frac{2}{2} \sum\limits_{0=1} (y_i - \lambda_i))$

To find the output of the decision tree use $y_m = arg \min\limits_{y} \sum\limits_{x_i \in \mathbf{R}_{i,m}} L(y_i, F_{m-1}(x_i) + \gamma)$

Where $i, L, y_i, \gamma, F(x)$ is the same as above and $R_{i,m}$ is the terminal region where j is index of the leaf.

4 Work explained

To make $\log(1-p)$ into a function of the log(odds) we get

$$\begin{aligned}\log(1-p) = \log(1 - \frac{e^{\log(odds)}}{1+e^{\log(odds)}}) =\\ \log(\frac{1+e^{\log(odds)}}{1+e^{\log(odds)}} - \frac{e^{\log(odds)}}{1+e^{\log(odds)}}) =\\ \log(\frac{1}{1+e^{\log(odds)}}) =\\ \log(1) - \log(1+e^{\log(odds)}) =\\ -\log(1+e^{\log(odds)})\end{aligned}$$

Then replacing $\log(1-p)$ with the function of the log(odds) we get

$$\begin{aligned}-\sum_{i=1}^{N} y_i * \log(p) + (1 - y_i * \log(1-p)) =\\ \sum_{i=1}^{N} -\text{Observed} * \log(p) - (1 - \text{Observed}) * \log(1-p) =\\ \sum_{i=1}^{N} -\text{Observed} * \log(p) - \log(1-p) + \text{Observed} * \log(1-p) =\\ \sum_{i=1}^{N} -\text{Observed} * [\log(p) - \log(1-p)] - \log(1-p) =\\ \sum_{i=1}^{N} -\text{Observed} * [\log(p) - \log(1-p)] - \log(1-p) =\\ \sum_{i=1}^{N} -\text{Observed} * [\log(\frac{p}{1-p})] - \log(1-p) =\\ \sum_{i=1}^{N} -\text{Observed} * \log(odds) - (-\log(1+e^{\log(odds)})) =\\ \sum_{i=1}^{N} -\text{Observed} * \log(odds) + \log(1+e^{\log(odds)}) =\\ \sum_{i=1}^{N} \text{Observed} * \log(odds) - \log(1+e^{\log(odds)})\end{aligned}$$

Now to find $\frac{\partial^2}{\partial \log(odds)^2}$ of $-\text{Observed} * \log(odds) - \log(1+e^{\log(odds)})$

$$\frac{\partial^2}{\partial \log(odds)^2} =$$

$$\frac{\partial}{\partial \log(odds)} - \text{Observed} - \frac{e^{\log(odds)}}{1+e^{\log(odds)}}) =$$

rewrite to make easier

$$\frac{\partial}{\partial \log(odds)} - \text{Observed} + (1+e^{\log(odds)})^{-1} * e^{\log(odds)}) =$$

$$-1(1+e^{\log(odds)})^{-2} e^{\log(odds)} * e^{\log(odds)} + (1+e^{\log(odds)})^{-1} e^{\log(odds)} =$$

$$\frac{-e^{2\log(odds)}}{(1+e^{\log(odds)})^2} + \frac{e^{\log(odds)}}{(1+e^{\log(odds)})} =$$

$$\frac{-e^{2\log(odds)}}{(1+e^{\log(odds)})^2} + \frac{e^{\log(odds)}}{(1+e^{\log(odds)})} \frac{(1+e^{\log(odds)})}{(1+e^{\log(odds)})} =$$

$$\frac{-e^{2\log(odds)}}{(1+e^{\log(odds)})^2} + \frac{e^{\log(odds)}}{+} e^{2\log(odds)} (1+e^{\log(odds)})^2 =$$

$$\frac{-e^{2\log(odds)} + e^{\log(odds)}}{+} e^{2\log(odds)} (1+e^{\log(odds)})^2 =$$

$$\frac{e^{\log(odds)}}{(1+e^{\log(odds)})(1+e^{\log(odds)})} =$$

$$\frac{e^{\log(odds)}}{(1+e^{\log(odds)})} \frac{1}{(1+e^{\log(odds)})} =$$

$$p(1-p)$$

4.1 Python code

Below is the code to recreate the trees used in this example

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import pandas as pd
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
from sklearn.model_selection import GridSearchCV
X_train = df.iloc[:,1:5]
y_train = df.iloc[:,5:6]
dtc = DecisionTreeClassifier(random_state=1)
dtc.fit(X_train,y_train)
param_grid =
'max_depth': range(2, 10, 3),
'min_samples_leaf': range(1, 20, 2),
'min_samples_split': range(2, 20, 2),
'criterion': ["entropy", "gini"]
grid_search = GridSearchCV(estimator=dtc, param_grid=param_grid,
cv=5, verbose=True)
grid_search.fit(X_train, y_train)
print("best accuracy", grid_search.best_score_)
print(grid_search.best_estimator_)
tree_clf = grid_search.best_estimator_
plt.figure(figsize=(18, 15))
plot_tree(tree_clf, filled=True, feature_names=list(df.columns[1:5]),
class_names=df.Can_Swim_Text)
plt.savefig('tree_1.png')
plt.show()
X_train2 = df.iloc[:,3:5]
dtc2 = DecisionTreeClassifier(random_state=1)
dtc2.fit(X_train2,y_train)
grid_search2 = GridSearchCV(estimator=dtc2, param_grid=param_grid,
cv=5, verbose=True)
grid_search2.fit(X_train2, y_train)
print("best accuracy", grid_search2.best_score_)
print(grid_search2.best_estimator_)
tree_clf2 = grid_search2.best_estimator_
plt.figure(figsize=(14, 15))
plot_tree(tree_clf2, filled=True, feature_names=list(df.columns[3:5]),
class_names=df.Can_Swim_Text)
plt.savefig('tree_2')
plt.show()
```

5 Special Thanks

I wanted to give special thanks to StatQuest from YouTube who had a great video series on Gradient boost, the video are listed below and in the references.

Gradient Boost Part 1 (of 4): Regression Main Ideas(3)
Gradient Boost Part 2 (of 4): Regression Details(4)
Gradient Boost Part 3 (of 4): Classification(5)
Gradient Boost Part 4 (of 4): Classification Details(6)

6 References

References

[1] T. Chen and C. Guestrin, *Xgboost: A scalable tree boosting system*, CoRR, abs/1603.02754 (2016).

[2] J. Cramer, *The Origins of Logistic Regression*, Tinbergen Institute Discussion Papers 02-119/4, Tinbergen Institute, Dec. 2002.

[3] S. with Josh Starmer, *Gradient boost part 1 (of 4): Regression main ideas*, March 2019.

[4] ——, *Gradient boost part 2 (of 4): Regression details*, April 2019.

[5] ——, *Gradient boost part 3 (of 4): Classification*, April 2019.

[6] ——, *Gradient boost part 4 (of 4): Classification details*, April 2019.