Ridge, Lasso ve Elastic Net

Buse Köseoğlu · Follow

3 min read · Oct 26, 2020

Ridge Regresyon (L2 regularization)

Çok değişkenli regresyon verilerini analiz etmede kullanılır. Amaç hata kareler toplamını minimize eden katsayıları, bu katsayılara bir ceza uygulayarak bulmaktır. Over-fittinge karşı dirençlidir. Çok boyutluluğa çözüm sunar. Tüm değişkenler ile model kurar, ilgisiz değişkenleri çıkarmaz sadece katsayılarını sıfıra yaklaştırır. Modeli kurarken alpha (ceza) için iyi bir değer bulmak gerekir.

Lasso Regresyon (L1 regularization)

Ürettiği modelin tahmin doğruluğunu ve yorumlanabilirliğini arttırmak için hem değişken seçimi hem de regularization yapar. Aynı ridge regresyonda olduğu gibi amaç hata kareler toplamını minimize eden katsayıları, katsayılara ceza uygularayarak bulmaktır. Fakat ridge regresyondan farklı olarak ilgisiz değişkenlerin katsayılarını sıfıra eşitler.

Elastic Net

Amaç ridge ve lasso regresyon ile aynıdır ama elastic net, ridge ve lasso regresyonu birleştirir. Ridge regresyon tarzı cezalandırma ve lasso regresyon tarzında değişken seçimi yapar.

Uygulama

Hepsinin kısaca ne olduğunu öğrendiğimize göre artık uygulama kısmına geçebiliriz. Uygulamamız için “Hitters.csv” veri setini kullanacağız. Öncelikle gerekli olan kütüphaneleri dahil ederek başlayalım.

Gerekli kütüphaneleri dahil ettiğimize göre veri setini incelemeye geçebiliriz.

	AtBat	Hits	HmRun	Runs	RBI	Walks	Years	CAtBat	CHits	CHmRun	CRuns	CRBI	CWalks	League	Division	PutOuts	Assists	Errors	Salary	NewLeague
0	293	66	1	30	29	14	1	293	66	1	30	29	14	A	E	446	33	20	NaN	A
1	315	81	7	24	38	39	14	3449	835	69	321	414	375	N	W	632	43	10	475.0	N
2	479	130	18	66	72	76	3	1624	457	63	224	266	263	A	W	880	82	14	480.0	A
3	496	141	20	65	78	37	11	5628	1575	225	828	838	354	N	E	200	11	3	500.0	N
4	321	87	10	39	42	30	2	396	101	12	48	46	33	N	E	805	40	4	91.5	N

df.info() ile veri setimiz hakkında daha geniş bir bilgiye sahip olabiliriz.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 322 entries, 0 to 321
Data columns (total 20 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   AtBat      322 non-null    int64
 1   Hits       322 non-null    int64
 2   HmRun      322 non-null    int64
 3   Runs       322 non-null    int64
 4   RBI        322 non-null    int64
 5   Walks      322 non-null    int64
 6   Years      322 non-null    int64
 7   CAtBat     322 non-null    int64
 8   CHits      322 non-null    int64
 9   CHmRun     322 non-null    int64
 10  CRuns      322 non-null    int64
 11  CRBI       322 non-null    int64
 12  CWalks     322 non-null    int64
 13  League     322 non-null    object
 14  Division   322 non-null    object
 15  PutOuts    322 non-null    int64
 16  Assists    322 non-null    int64
 17  Errors     322 non-null    int64
 18  Salary     263 non-null    float64
 19  NewLeague  322 non-null    object
dtypes: float64(1), int64(16), object(3)
memory usage: 50.4+ KB
```

Yukarıdaki çıktıyı incelediğimizde veri setinin boyutunun (322,20) olduğunu görüyoruz. Üç tane kategorik değişkenimiz var bunlar object olarak belirtilmiş ve aynı zamanda *Salary* değişkeninin içinde eksik veri mevcut. Bu eksik verilerin sayısını görmek için;

Bunun sonucunda Salary değişkeni içinde 59 tane eksik veri olduğunu görüyoruz. Bu eksik verileri Salary değişkeninin ortalaması ile dolduracağız.

Yukarıdaki *“inplace=True”* parametresi yaptığımız değişikliğin kalıcı bir şekilde uygulanmasını sağlar.

Sıra geldi kategorik değişkenleri numerik değişkenlere dönüştürmeye. Bunun için pandas içindeki *“get_dummies()”*i kullanacağız.

	AtBat	Hits	HmRun	Runs	RBI	Walks	Years	CAtBat	CHits	CHmRun	CRuns	CRBI	CWalks	PutOuts	Assists	Errors
0	293.0	66.0	1.0	30.0	29.0	14.0	1.0	293.0	66.0	1.0	30.0	29.0	14.0	446.0	33.0	20.0
1	315.0	81.0	7.0	24.0	38.0	39.0	14.0	3449.0	835.0	69.0	321.0	414.0	375.0	632.0	43.0	10.0
2	479.0	130.0	18.0	66.0	72.0	76.0	3.0	1624.0	457.0	63.0	224.0	266.0	263.0	880.0	82.0	14.0
3	496.0	141.0	20.0	65.0	78.0	37.0	11.0	5628.0	1575.0	225.0	828.0	838.0	354.0	200.0	11.0	3.0
4	321.0	87.0	10.0	39.0	42.0	30.0	2.0	396.0	101.0	12.0	48.0	46.0	33.0	805.0	40.0	4.0

“get_dummies()” içinde numerik değişkenlere dönüşücek kategorik değişkenleri yazıyoruz. Daha sonra veri setinden bu kategorik değişkenleri ve tahmin etmek istediğimiz değişkeni düşürüyoruz ve bunu X_’ye eşitliyoruz.

	NewLeague_A	NewLeague_N	League_A	League_N	Division_E	Division_W
0	1	0	1	0	1	0
1	0	1	0	1	0	1
2	1	0	1	0	0	1
3	0	1	0	1	1	0
4	0	1	0	1	1	0

dms'i incelediğimizde kategorik değişkenlerin 1 ve 0'lar ile numerik değişkenlere dönüştüğünü görüyoruz. Veri setinde her bir değişkenin bir tanesini kullanmamız yeterli olacaktır.

	AtBat	Hits	HmRun	Runs	RBI	Walks	Years	CAtBat	CHits	CHmRun	CRuns	CRBI	CWalks	PutOuts	Assists	Errors	NewLeague_A	League_A	Division_E
0	293.0	66.0	1.0	30.0	29.0	14.0	1.0	293.0	66.0	1.0	30.0	29.0	14.0	446.0	33.0	20.0	1	1	1
1	315.0	81.0	7.0	24.0	38.0	39.0	14.0	3449.0	835.0	69.0	321.0	414.0	375.0	632.0	43.0	10.0	0	0	0
2	479.0	130.0	18.0	66.0	72.0	76.0	3.0	1624.0	457.0	63.0	224.0	266.0	263.0	880.0	82.0	14.0	1	1	0
3	496.0	141.0	20.0	65.0	78.0	37.0	11.0	5628.0	1575.0	225.0	828.0	838.0	354.0	200.0	11.0	3.0	0	0	1
4	321.0	87.0	10.0	39.0	42.0	30.0	2.0	396.0	101.0	12.0	48.0	46.0	33.0	805.0	40.0	4.0	0	0	1

Dönüştürülen değişkenler ve X_'yi birleştirerek X'i elde ediyoruz. X, y'yi tahmin etmek için kullanacağımız bağımsız değişkenleri içeriyor. y'yi de tahmin etmek istediğimiz değişkene eşitliyoruz.

Bunları yaptıktan sonra artık veriyi train ve test olarak bölebiliriz.

X_train ve **y_train** modeli geliştirmek için **X_test** ve **y_test** ise modeli test etmek için kullanılacak bağımlı ve bağımsız değişkenleri gösteriyor. **Test_size** verilerin % kaçını test için kullanılacağını (%30) belirtir. Burada herhangi bir değer belirtmezsek defaultta gelen 0.25 değeri kullanılır. **Random_state** ise programı her çalıştırdığımızda aynı ayrımı görmek için kullanılır.

StandartScaler verileri, ortalaması 0, standart sapması 1 olacak şekilde dönüştürür. Kısacası verileri standartlaştırır.

RIDGE REGRESYON

İlk başta *"ridge"* adında bir model yaratıyoruz ve modele hiçbir parametre vermeden X_train ve y_train ile eğitiyoruz. Bunun sonucunda aldığımız hata 331.527 oluyor. Daha sonra **RidgeCV** yi kullanarak bir cross validation işlemi uyguluyoruz. Bu işlem veri setimiz için en uygun parametreleri elde etmemizi sağlıyor. İşlem tamamlandıktan sonra elde ettiğimiz en iyi alpha değeri ile yeni bir model (ridge_tuned) kuruyoruz ve eğitiyoruz. Buradan elde ettiğimiz hata ise 331.299 oluyor.

LASSO REGRESYON

“lasso” adındaki modelimizi oluşturup, eğittikten sonra elde ettiğimiz hata 331.238 oluyor. Daha sonra en iyi parametreleri bulmak için **LassoCV** yani cross validation uyguluyoruz. Son olarak da elde ettiğimiz parametrelerle yeni bir model kurup eğittiğimiz zaman hata 331.135 oluyor.

ELASTIC NET

Burada da aynı yukarıdaki işlemleri uyguluyoruz. İlk oluşturduğumuz modelde elde ettiğimiz hata 356.537 oluyor. Tune edildikten sonra elde ettiğimiz hata ise 353.737 oluyor.

Machine Learning | Python | Ridge Regression | Lasso Regression | Elastic Net

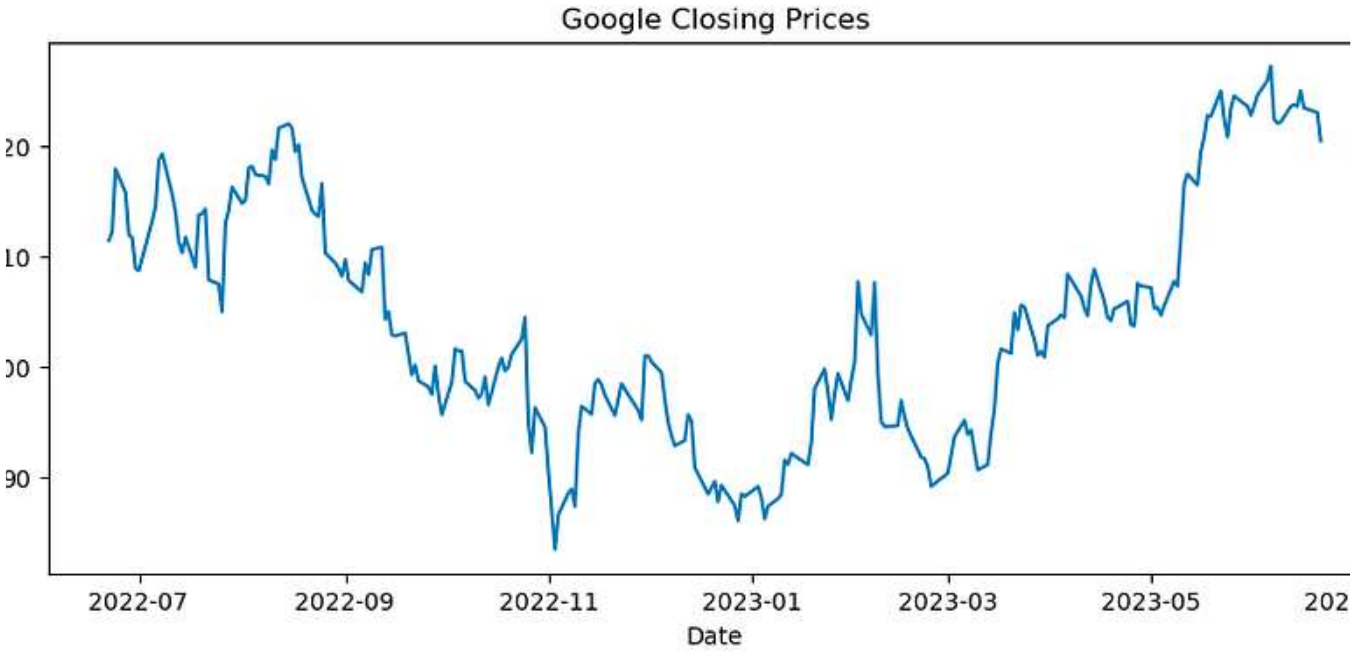
Follow

Written by Buse Köseoğlu

371 Followers

BAU Software Engineering | Data Science | https://www.linkedin.com/in/busekoseoglu/

More from Buse Köseoğlu



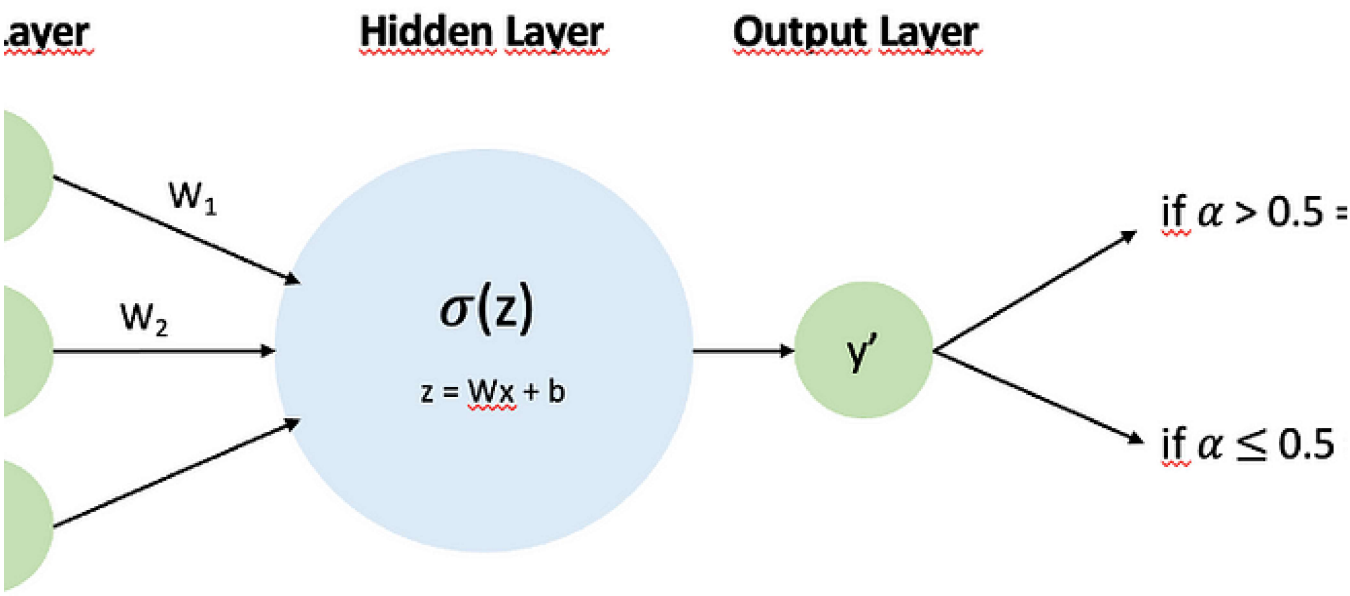
Buse Köseoğlu

Guide to Time Series Analysis with Python—1: Analysis Techniques and Baseline Model

Time series analysis is a method used in various industries, such as stock market prediction, product price forecasting, weather...

6 min read · Jul 1

313



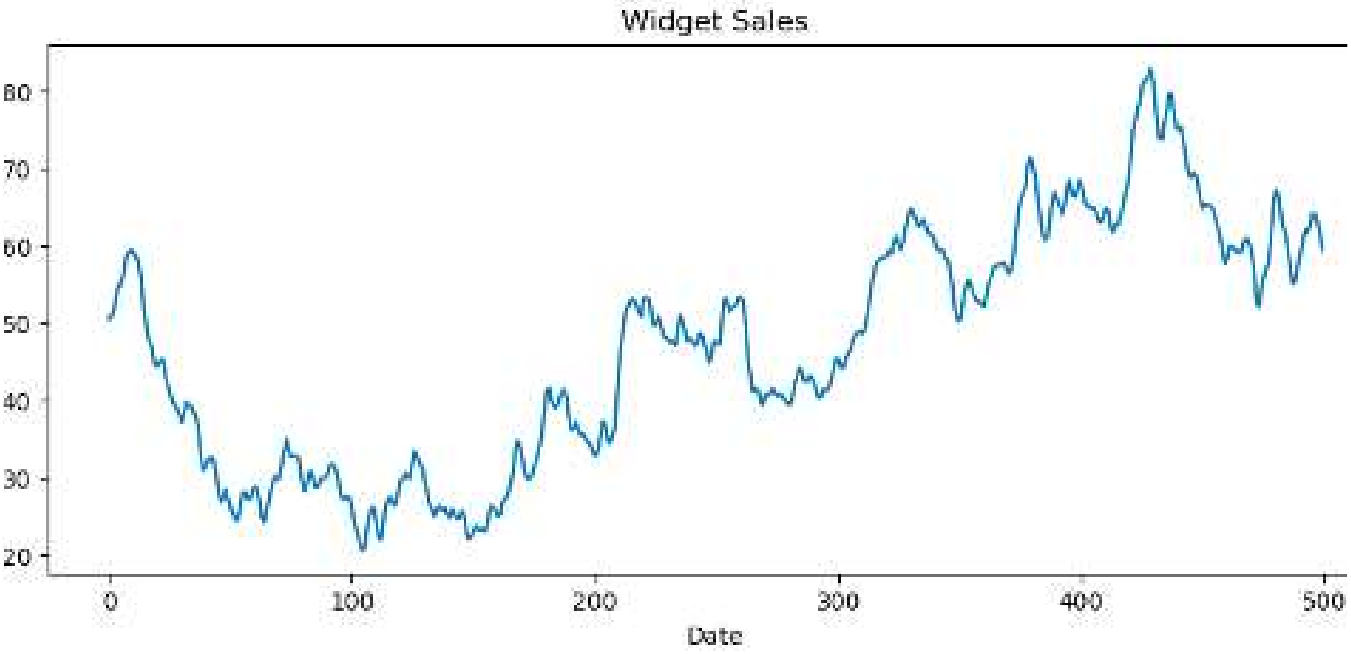
Buse Köseoğlu

One Layer Neural Network From Scratch—Classification

Classification is a supervised learning method. In supervised learning, we have labels for our data. Algorithms learn from these labels...

7 min read · Jul 28

185 1



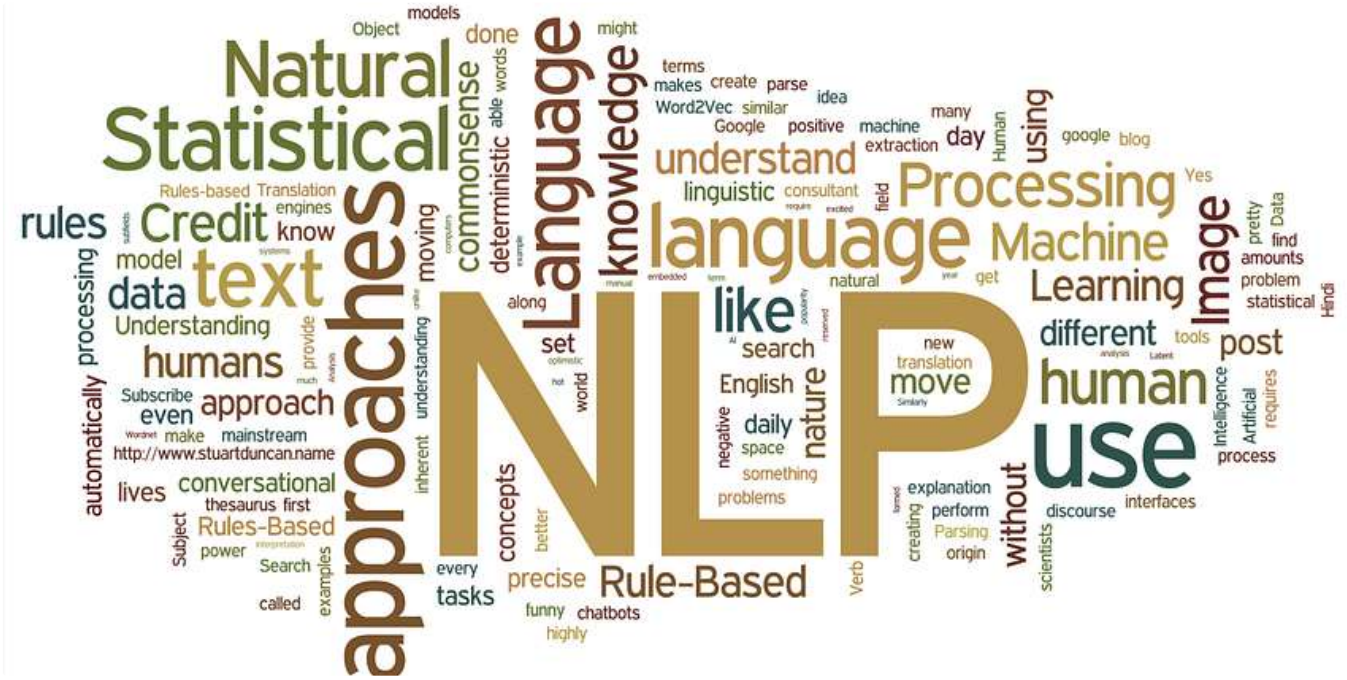
Buse Köseoğlu

Guide to Time Series Analysis with Python—2: Moving Average Process

In my previous article, I talked about how time series analysis should be done, analysis methods and baseline model creation. If you do...

5 min read · Jul 15

105



Buse Köseoğlu

NLP—Text Preprocessing

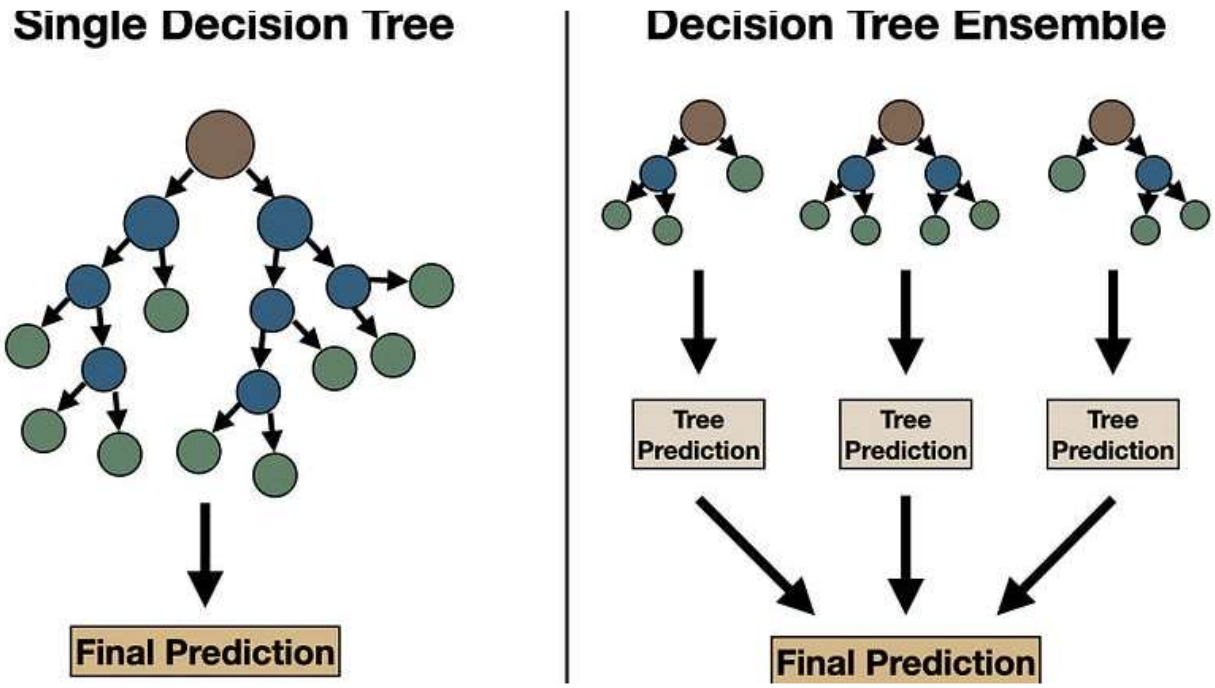
Since the data to be used in NLP projects is text data, it has an unstructured structure and, as in other projects, it is very important...

3 min read · Dec 15

78

See all from Buse Köseoğlu

Recommended from Medium



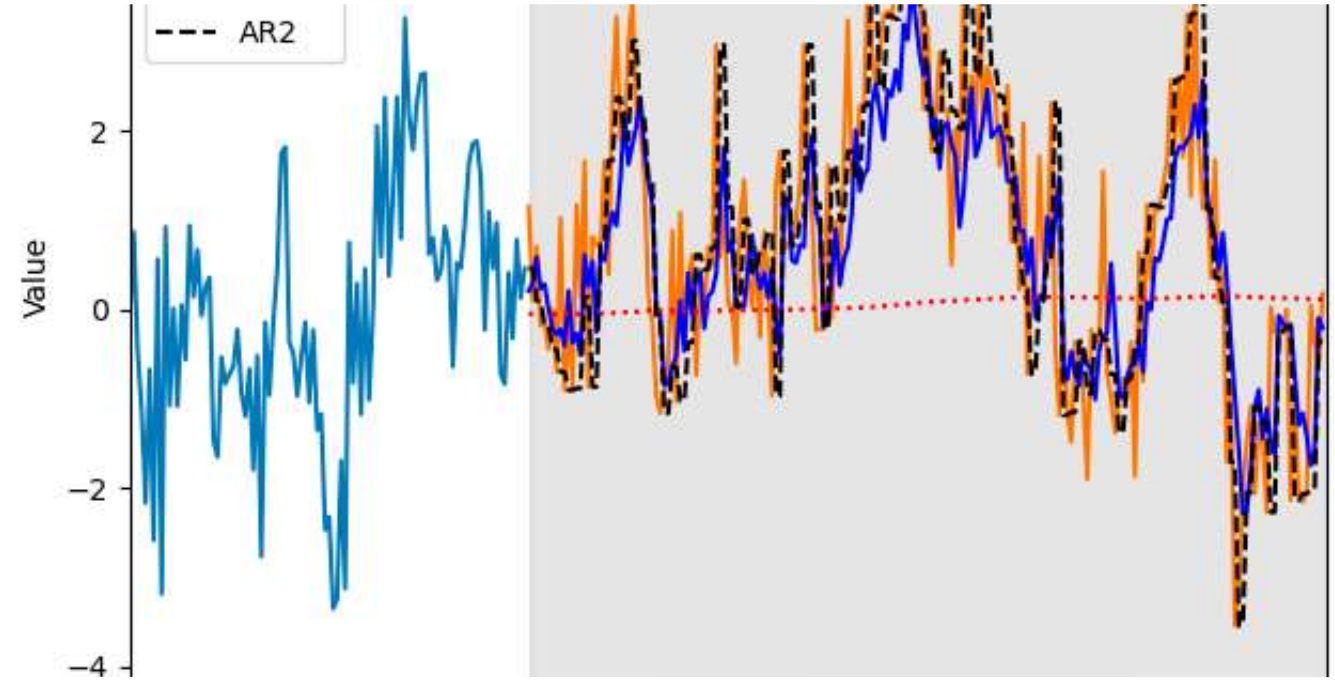
Shaw Talebi in Towards Data Science

10 Decision Trees are Better Than 1

Breaking down bagging, boosting, Random Forest, and AdaBoost

✦ · 9 min read · Feb 27

185 5



Buse Köseoğlu

Guide to Time Series Analysis with Python—3: Autoregressive Process

In the previous article, we examined the moving average process. This time we will examine another statistical model, the autoregressive...

4 min read · Aug 7

112

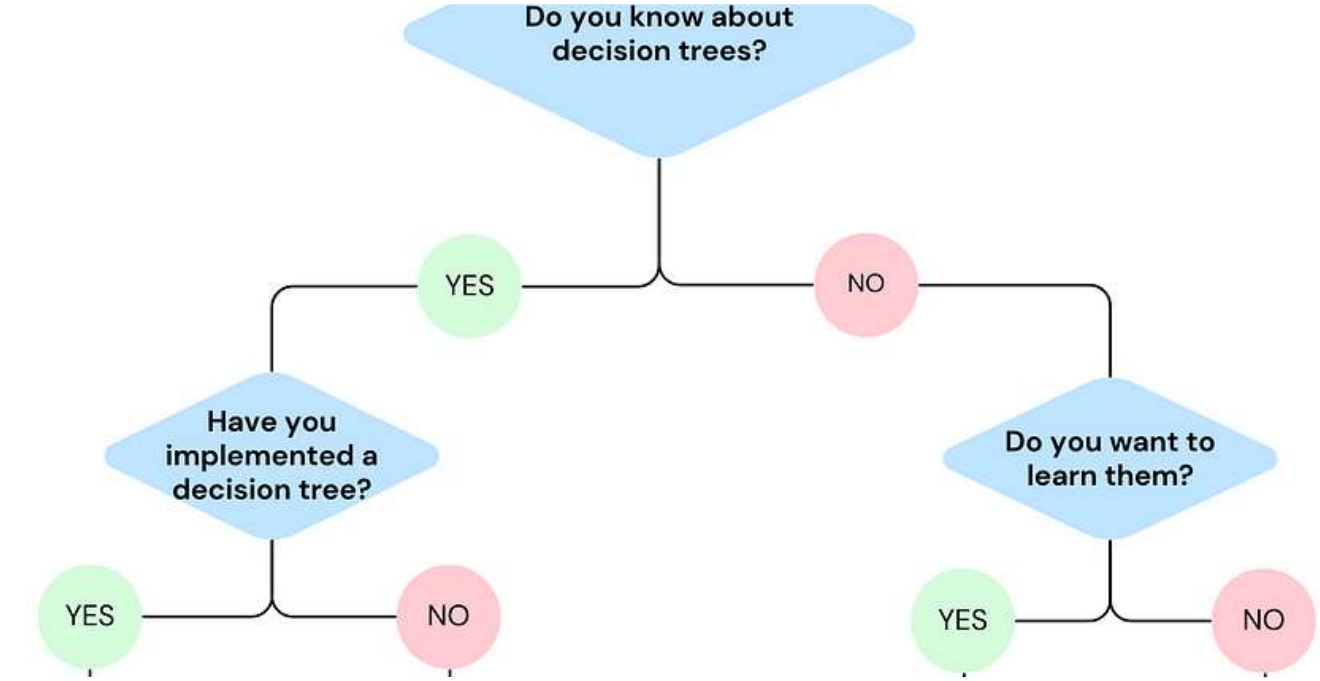
Lists

Predictive Modeling w/ Python
20 stories · 737 saves

Practical Guides to Machine Learning
10 stories · 848 saves

Coding & Development
11 stories · 349 saves

Natural Language Processing
1042 stories · 516 saves



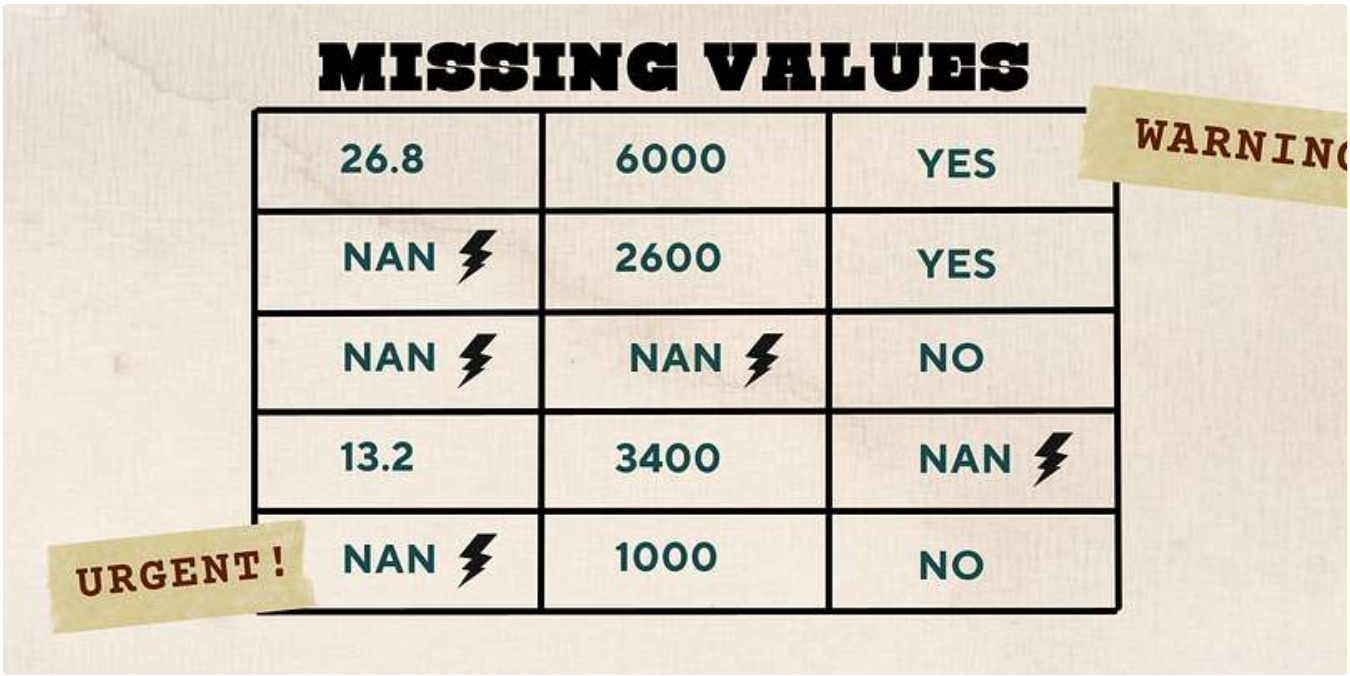
Enozeren

Building a Decision Tree From Scratch with Python

Decision Trees are machine learning algorithms used for classification and regression tasks with tabular data. Even though a basic decision...

8 min read · Oct 13

62



Hasan Hüseyin Coşgun

Dealing with Missing Data from Zero to Advanced

Simple and advanced imputation; Drop, Mode, Median, KNN, MICE

8 min read · Aug 23

79 2

devin schumacher in SERP AI

AdaBoost

AdaBoost: Definition, Explanations, Examples & Code

5 min read · Jul 21

759

Chinenye Okeke

Introduction to BIG O Notation—Time and Space Complexity

A clear and concise introduction to Big O Notation, focusing on time and space complexity in algorithm analysis

6 min read · Jul 17

2