Logistic Regression with Julia

ref. Logistic Regression for Classification

May 2022.

Table of Contents


```
begin
    using PlutoUI ✓
    PlutoUI.TableOfContents(indent=true, depth=4, aside=true)
end
```

Import the packages

```
begin
    # importing the packages
    using DataFrames ✓ , CSV ✓ , GLM ✓ , Lathe ✓ , MLBase ✓
    using ClassImbalance ✓ , ROCAnalysis ✓
end
```

Load the data

	Name	Author	User_Rating	Reviews	Price	Year	Gen
1	"10-Day Green Smoothie Cleanse"	"JJ Smith"	4.7	17350	8	2016	"non_fi
2	"11/22/63: A Novel"	"Stephen King"	4.6	2052	22	2011	"fictior
3	"12 Rules for Life: An Antidote to Cha	"Jordan B. Peterson"	4.7	18979	15	2018	"non_fi
4	"1984 (Signet Classics)"	"George Orwell"	4.7	21424	6	2017	"fictior
5	"5,000 Awesome Facts (About Everything	"National Geographic Kids"	4.8	7665	12	2019	"non_fi
6	"A Dance with Dragons (A Song of Ice a	"George R. R. Martin"	4.4	12643	11	2011	"fictior
7	"A Game of Thrones / A Clash of Kings	"George R. R. Martin"	4.7	19735	30	2014	"fictior

```
begin
    df = CSV.read("./bestsellers.csv", DataFrame)
    first(df, 7)
end
```

Logistic Regression Equation

$$P = \frac{1}{1 + e^{-}(a + bX)}$$

Summary Statistics

	variable	mean	min	median	max	nunique
1	:Name	nothing	"10-Day Green Smoothie Cleanse"	nothing	"You Are a Badass: How to Stop Doubtin	351
2	:Author	nothing	"Abraham Verghese"	nothing	"Zhi Gang Sha"	248
3	:User_Rating	4.61836	3.3	4.7	4.9	nothing
4	:Reviews	11953.3	37	8580.0	87841	nothing
5	:Price	13.1	0	11.0	105	nothing
6	:Year	2014.0	2009	2014.0	2019	nothing
7	:Genre	nothing	"fiction"	nothing	"non_fiction"	2

```
describe(df)
```

Here we are going to use the `:Genre` as our target variable (the one we want to make a prediction for). We will need to *one-hot-encode* this variable.

```
(550, 7)
```

```
size(df)
```

Handling missing values

In this case we will drop all rows containing a missing value. Other more sophsiticated techniques are possible (imputing using the mean ...).

	Name	Author	User_Rating	Reviews	Price	Year	Ge
1	"10-Day Green Smoothie Cleanse"	"JJ Smith"	4.7	17350	8	2016	"non_fi
2	"11/22/63: A Novel"	"Stephen King"	4.6	2052	22	2011	"fictic
3	"12 Rules for Life: An Antidote to Cha	"Jordan B. Peterson"	4.7	18979	15	2018	"non_fi
4	"1984 (Signet Classics)"	"George Orwell"	4.7	21424	6	2017	"fictic
5	"5,000 Awesome Facts (About Everything	"National Geographic Kids"	4.8	7665	12	2019	"non_fi
6	"A Dance with Dragons (A Song of Ice a	"George R. R. Martin"	4.4	12643	11	2011	"fictic
7	"A Game of Thrones / A Clash of Kings	"George R. R. Martin"	4.7	19735	30	2014	"fictic
8	"A Gentleman in Moscow: A Novel"	"Amor Towles"	4.7	19699	15	2017	"fictic
9	"A Higher Loyalty: Truth, Lies, and Le	"James Comey"	4.7	5983	3	2018	"non_fi
10	"A Man Called Ove: A Novel"	"Fredrik Backman"	4.6	23848	8	2016	"fictic

⋮ more

```
dropmissing!(df)
```

```
(550, 7)
```

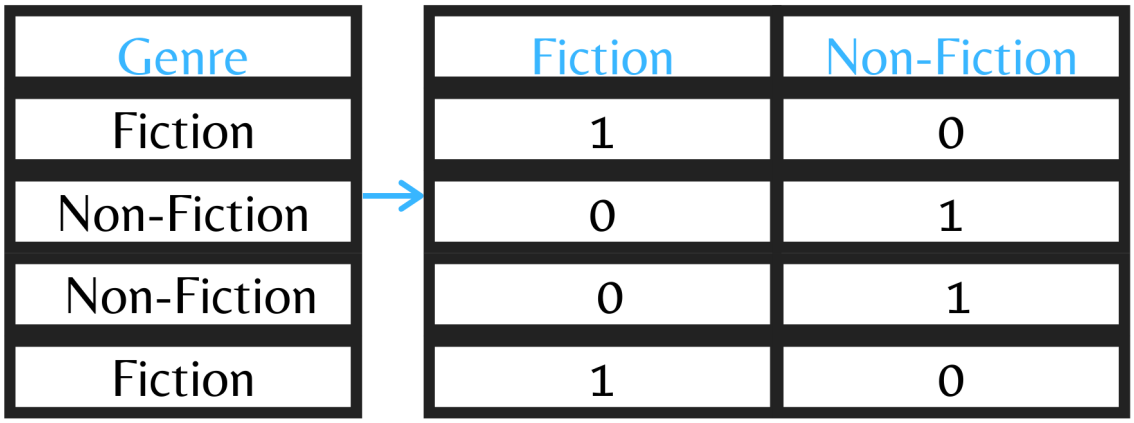
```
size(df) # there was no missing values actually!
```

One Hot Encoding

By one hot encoding the `:Genre` feature, we will get two new columns in our dataframe, namely:

- fiction
- non fiction

One-Hot Encoding



Vinita Silaparasetty

```
`LocalResource` **will not work** when you share the script/notebook with so
meone else, _unless they have those resources at exactly the same location o
n their file system_.

## Recommended alternatives (images)
1. Go to [imgur.com](https://imgur.com) and drag&drop the image to the page.
Right click on the image, and select "Copy image location". You can now use
the image like so: `PlutoUI.Resource("https://i.imgur.com/SAzsMMA.jpg")`.
2. If your notebook is part of a git repository, place the image in the repo
sitory and use a relative path: `PlutoUI.LocalResource("../images/cat.jpg")
`.
```

	Name	Author	User_Rating	Reviews	Price	Year	Gen
1	"10-Day Green Smoothie Cleanse"	"JJ Smith"	4.7	17350	8	2016	"non_fi
2	"11/22/63: A Novel"	"Stephen King"	4.6	2052	22	2011	"fictio
3	"12 Rules for Life: An Antidote to Cha	"Jordan B. Peterson"	4.7	18979	15	2018	"non_fi
4	"1984 (Signet Classics)"	"George Orwell"	4.7	21424	6	2017	"fictio
5	"5,000 Awesome Facts (About Everything	"National Geographic Kids"	4.8	7665	12	2019	"non_fi

```
begin
    target_final = Lathe.preprocess.OneHotEncode(df, :Genre)
    first(df, 5)
end
```

```
(550, 9)
```

```
size(df) # 9 columns
```

```
[:Name, :Author, :User_Rating, :Reviews, :Price, :Year, :Genre, :non_fiction, :fiction]
```

```
names(df)
```

Feature Selection

We are going to select the numerical features of our dataframe and `:fiction` as our target (which is derived form :Genre and one-hot-encoded)

	User_Rating	Reviews	Price	Year	fiction
1	4.7	17350	8	2016	false
2	4.6	2052	22	2011	true
3	4.7	18979	15	2018	false
4	4.7	21424	6	2017	true
5	4.8	7665	12	2019	false
6	4.4	12643	11	2011	true
7	4.7	19735	30	2014	true

```
begin
    ndf = df[:, [:User_Rating, :Reviews, :Price, :Year, :fiction]]
    first(ndf, 7)
end
```

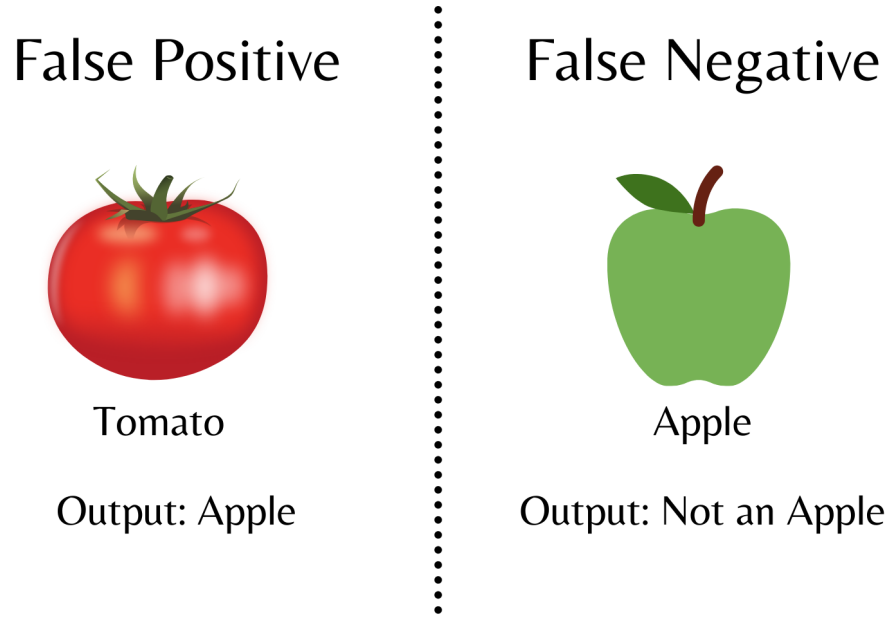
Check how the data is balanced

```
using FreqTables
```

```
classes =
▸ NamedArrays.NamedVector{Int64, Vector{Int64}, Tuple{OrderedCollections.OrderedDict{Bool, Int64}}}: [310, 240]
```

```
classes = freqtable(target_final[:fiction])
```

It looks like our dataset is slightly unbalanced with more non-fiction books than fiction books




```
`LocalResource` **will not work** when you share the script/notebook with so
meone else, _unless they have those resources at exactly the same location o
n their file system_.

## Recommended alternatives (images)
1. Go to [imgur.com](https://imgur.com) and drag&drop the image to the page.
Right click on the image, and select "Copy image location". You can now use
the image like so: `PlutoUI.Resource("https://i.imgur.com/SAzsMMA.jpg")`.
2. If your notebook is part of a git repository, place the image in the repo
sitory and use a relative path: `PlutoUI.LocalResource("../images/cat.jpg")
`.
```

Split the Data

```
using Lathe .preprocess:TrainTestSplit
```

```
((410, 5), (140, 5))
```

```
begin
    train, test = TrainTestSplit(ndf, .75)
    size(train), size(test)
end
```

Build the model

```
fm = FormulaTerm
     Response:
       fiction(unknown)
     Predictors:
       User_Rating(unknown)
       Reviews(unknown)
       Price(unknown)
       Year(unknown)
```

```
fm = @formula(fiction ~ User_Rating + Reviews + Price + Year)
```

```
logit =
StatsModels.TableRegressionModel{GeneralizedLinearModel{GLM.GlmResp{Vector{Float64}, Binomial{Float64}, LogitLink}, (

fiction ~ 1 + User_Rating + Reviews + Price + Year

Coefficients:
```

	Coef.	Std. Error	z	Pr(>|z|)	Lower 95%	Upper 95%
(Intercept)	227.78	74.3908	3.06	0.0022	81.9764	373.583
User_Rating	1.26959	0.550483	2.31	0.0211	0.190664	2.34852
Reviews	6.8363e-5	1.20317e-5	5.68	<1e-07	4.47812e-5	9.19448e-5
Price	-0.0339151	0.0129206	-2.62	0.0087	-0.0592391	-0.00859119
Year	-0.116349	0.0372802	-3.12	0.0018	-0.189417	-0.0432808

```
logit = glm(fm, train, Binomial(), LogitLink())
```

Predictions

```
predictions = predict(logit, test);
```

```
prediction_class = [x < 0.5 ? false : true for x ∈ predictions];
```

Accuracy

```
prediction_df = DataFrame(y=test.fiction, ŷ=prediction_class, prob_predicted=predictions);
```

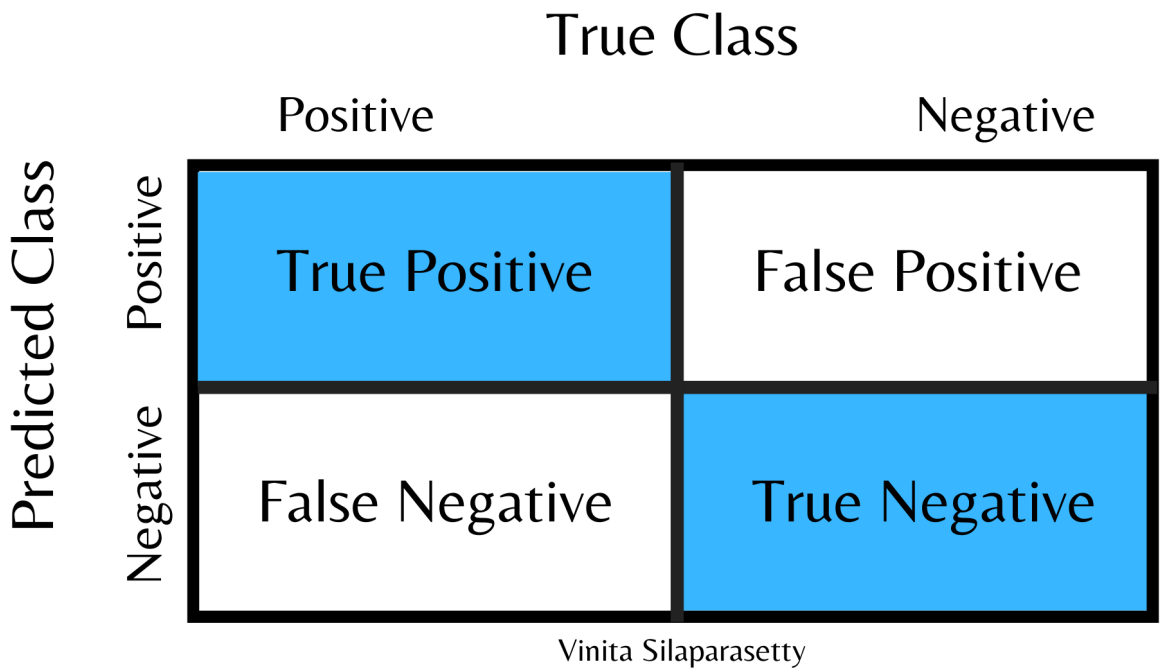
```
prediction_df_correctly_classified = prediction_df.y .== prediction_df.ŷ;
```

```
accuracy = 0.6357142857142857
```

```
accuracy = prediction_df_correctly_classified |> mean
```

Confusion matrix

We are going to use the ROC curve to evaluate the performance of our current model.




```
`LocalResource` **will not work** when you share the script/notebook with so
meone else, _unless they have those resources at exactly the same location o
n their file system_.

## Recommended alternatives (images)
1. Go to [imgur.com](https://imgur.com) and drag&drop the image to the page.
Right click on the image, and select "Copy image location". You can now use
the image like so: `PlutoUI.Resource("https://i.imgur.com/SAzsMMA.jpg")`.
2. If your notebook is part of a git repository, place the image in the repo
sitory and use a relative path: `PlutoUI.LocalResource("../images/cat.jpg")
`.
```

```
confusion_matrix = MLBase.ROCNums{Int64}
                     p = 67
                     n = 73
                     tp = 33
                     tn = 56
                     fp = 17
                     fn = 34
```

```
confusion_matrix = MLBase.roc(prediction_df.y, prediction_df.ŷ)
```

False negative rate

```
0.5074626865671642
```

```
false_negative_rate(confusion_matrix)
```

Now we are going to focus on the false negative and try to lower this rate.

Class Imbalance and SMOTE

```
▸ NamedArrays.NamedVector{Int64, Vector{Int64}, Tuple{OrderedCollections.OrderedDict{Bool, Int64}}}: [310, 240]
```

```
classes
```

Let's use the SMOTE technique. SMOTE stands *synthetic minority over sampling technique*.

```
X₂, y₂ = smote(
    ndf[!, [:User_Rating, :Reviews, :Price, :Year]], ndf.fiction,
    # ! to tell that we do not want the selected features to be balanced
    k=1, pct_under=200, pct_over=100
    # the value chosen for the percentages are derived from the class imbalacne (classes)
    # thus 200 because max is 310 and minimum is 240 - with 200 we incraese the likelihood of achieving balance
);
```

```
balanced_classes = freqtable(y₂);
# smote add some synthetic values - this is why we get 480 in both cases now
```

```
ndf₂ = hcat(X₂, y₂);
```

▸ [:User_Rating, :Reviews, :Price, :Year, :target]

A new model

StatsModels.TableRegressionModel{GeneralizedLinearModel{GLM.GlmResp{Vector{Float64}, Binomial{Float64}, LogitLink}, (

target ~ 1 + User_Rating + Reviews + Price + Year

Coefficients:

	Coef.	Std. Error	z	Pr(>|z|)	Lower 95%	Upper 95%
(Intercept)	206.742	48.7467	4.24	<1e-04	111.201	302.284
User_Rating	1.64958	0.348258	4.74	<1e-05	0.967005	2.33215
Reviews	8.19437e-5	8.99482e-6	9.11	<1e-19	6.43142e-5	9.95732e-5
Price	-0.0459172	0.00936165	-4.90	<1e-06	-0.0642657	-0.0275687
Year	-0.10662	0.0244044	-4.37	<1e-04	-0.154452	-0.0587887

```
begin
    fm₂ = @formula(target ~ User_Rating + Reviews + Price + Year)
    logit₂ = glm(fm₂, ndf₂, Binomial(), LogitLink())
end
```

```
predictions₂ = predict(logit₂, test);
```

```
prediction₂_class = [x < 0.5 ? false : true for x ∈ predictions₂];
```

```
prediction₂_df = DataFrame(y=test.fiction, ŷ=prediction₂_class, prob_predicted=predictions₂);
```

```
prediction₂_df_correctly_classified = prediction₂_df.y .== prediction₂_df.ŷ;
```

Accuracy

```
accuracy₂ = 0.6642857142857143
```

```
accuracy₂ = prediction₂_df_correctly_classified |> mean
```

Confusion matrix

```
confusion₂_matrix = MLBase.ROCNums{Int64}
                    p = 67
                    n = 73
                    tp = 47
                    tn = 46
                    fp = 27
                    fn = 20
```

```
confusion₂_matrix = MLBase.roc(prediction₂_df.y, prediction₂_df.ŷ)
```

False negative rate

```
0.29850746268656714
```

```
false_negative_rate(confusion₂_matrix)
```