Logistic Regression Classification (Python)

Gepoliano Chaves, Ph. D.

September 18th, 2023

Activity goal

The goal of the present activity was to construct a Machine Learning model to classify a neuroblastoma patient as high risk or low risk based on 22 phenotypic scores derived from genetic material of that patient.

Considerations for this activity

- Neuroblastoma, a cancer of the peripheral nervous system of children, has two main clinical risk groups: low risk and high risk.
- Epigenetic modification on cell free DNA present in the blood of patients may inform the state of disease progression such as cancer in that patient.
- We can apply Machine Learning algorithms on that information to classify a patient to the disease risk category using classifiers that inform disease state.
- We previously investigated hypoxia as a signal that modulates a cellular transition important in neuroblastoma disease progression.
- To test if a therapy decreased or changed disease risk category, we can compare the pattern of gene expression or the epigenetic pattern present in the blood of a patient before and after a specific therapy.
- Here we will go through a logistic regression model using 22 features isolated from the study of hypoxia in driving aggressive disease, to classify neuroblastoma tumors.
- Accuracy of the model measures how efficient it is in predicting the risk category.

References

How to remember classification concepts was found in @JerryAn2020

Real-time precision medicine

- Liquid biopsies may allow the development of real-time precision medicine if coupled to fast computational interpretation of the activity of biomarkers

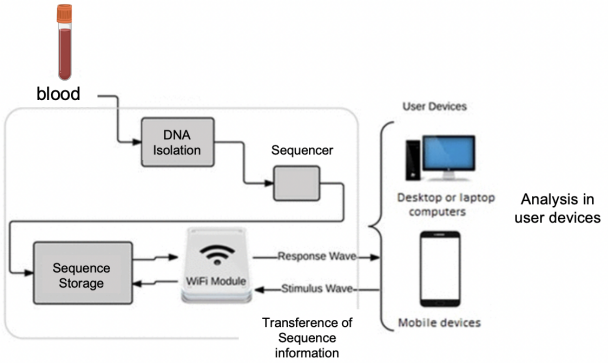


Figure 1: Diagnostic pipeline using liquid biopsies.

Modeling Feature Contribution to High Risk

- Logistic regression model: predict **categorical dependent variable** (y) from a number of independent variables (x)
- 22 neuroblastoma features (x=**independent variable**)
- Features are gene expression/phenotype quantification

Has high risk disease: Yes or No?

Figure 2: Probability function: Foundations of the Logistic Regression model.

Prob. Function: Feature Contribution to High Risk Disease

$$\ln \frac{p}{p+1} = b_0 + X_1 * b_1 + X_2 * b_2 + X_3 * b_3 + (\dots) + X_{22} * b_{22}$$

Figure 3: Features that contribute to probability function.

Load package reticulate

Correct path did not show up using the which command.

Ask reticulate what the possible paths are:

```
reticulate::conda_list()
```

```
##         name                                                    python
## 1  anaconda3                 /Users/gepolianochaves/anaconda3/bin/python
## 2 r-tutorial /Users/gepolianochaves/anaconda3/envs/r-tutorial/bin/python
## 3       base             /Users/gepolianochaves/opt/anaconda3/bin/python
```

Change the python path to include the path corresponding to the r-tutorial, above. Then set up the chunk to run python in rmarkdown.

There was an error at the end regarding matplotlib. One reference link mentioned installing matplotlib before pandas.

```
pip install matplotlib
```

```
## Keyring is skipped due to an exception: 'keyring.backends'
## Requirement already satisfied: matplotlib in /Users/gepolianochaves/anaconda3/lib/python3.7/site-pacl
## Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /Users/gepolianochaves/ana
## Requirement already satisfied: numpy>=1.11 in /Users/gepolianochaves/anaconda3/lib/python3.7/site-pac
## Requirement already satisfied: python-dateutil>=2.1 in /Users/gepolianochaves/anaconda3/lib/python3.7
## Requirement already satisfied: kiwisolver>=1.0.1 in /Users/gepolianochaves/anaconda3/lib/python3.7/si
## Requirement already satisfied: cycler>=0.10 in /Users/gepolianochaves/anaconda3/lib/python3.7/site-pa
## Requirement already satisfied: six>=1.5 in /Users/gepolianochaves/anaconda3/lib/python3.7/site-packag
## Requirement already satisfied: setuptools in /Users/gepolianochaves/anaconda3/lib/python3.7/site-pacl
```

Install pandas in this environment

```
pip install pandas
```

```
## Keyring is skipped due to an exception: 'keyring.backends'
## Requirement already satisfied: pandas in /Users/gepolianochaves/anaconda3/lib/python3.7/site-packages
## Requirement already satisfied: numpy>=1.13.3 in /Users/gepolianochaves/anaconda3/lib/python3.7/site-p
## Requirement already satisfied: python-dateutil>=2.6.1 in /Users/gepolianochaves/anaconda3/lib/python3
## Requirement already satisfied: pytz>=2017.2 in /Users/gepolianochaves/anaconda3/lib/python3.7/site-pa
## Requirement already satisfied: six>=1.5 in /Users/gepolianochaves/anaconda3/lib/python3.7/site-packag
```

Machine Learning definition

Machine learning is the use of algorithms and statistical models to analyze and draw inferences from *patterns* in data.

Part 1: Data Pre-processing; Get started on data analysis

Import pandas and dataset

```
import pandas as pd
dataset_kocak = pd.read_csv("/Users/gepolianochaves/Desktop/Gepoliano/Bioinformatics Bridge Course/2023
dataset_kocak.head(10)
```

```
##     sample id      MYCN    STC1  ...  CHIP_HIF1A_300  CHIP_EPAS1_300  high_risk
## 0  GSM1205238    42.499   2.948  ...       -0.306735       -0.319118        yes
## 1  GSM1205239    25.488   3.088  ...        0.002388       -0.201904        yes
## 2  GSM1205240   883.223   6.025  ...        0.061273       -0.174953        yes
## 3  GSM1205241   433.333   3.091  ...        0.131030        0.057692        yes
## 4  GSM1205242   854.354  10.856  ...       -0.132852       -0.133280        yes
## 5  GSM1205243   111.049   9.317  ...        0.059308       -0.070998         no
## 6  GSM1205244    63.553   6.560  ...       -0.265630       -0.114986        yes
## 7  GSM1205245  1558.482  58.680  ...       -0.066774       -0.146944        yes
## 8  GSM1205246   726.585  16.095  ...        0.138918        0.102813        yes
## 9  GSM1205247  1234.037   4.240  ...        0.295862        0.094235        yes
##
## [10 rows x 24 columns]
```

Getting the inputs and output

Get inputs and outputs from the Kocak dataset

```
X = dataset_kocak.iloc[:, 1:-1].values
y = dataset_kocak.iloc[:, -1]
```

```
X
```

```
## array([[ 4.24990000e+01,  2.94800000e+00,  1.28600000e+01, ...,
##         -1.35035382e-01, -3.06734981e-01, -3.19117757e-01],
##        [ 2.54880000e+01,  3.08800000e+00,  1.08560000e+01, ...,
##         -2.27619380e-01,  2.38836797e-03, -2.01903629e-01],
##        [ 8.83223000e+02,  6.02500000e+00,  1.26800000e+01, ...,
##         -3.67977581e-01,  6.12733677e-02, -1.74953282e-01],
##        ...,
##        [ 5.57960000e+01,  1.06330000e+01,  1.86180000e+01, ...,
##          1.21852708e-01,  1.30950985e-01,  2.67927509e-01],
##        [ 6.72110000e+01,  6.51000000e+00,  1.55710000e+01, ...,
##         -3.76899205e-02, -1.48956164e-01, -1.55718814e-01],
##        [ 4.08360000e+01,  2.91880000e+01,  1.79240000e+01, ...,
##          1.81855608e-01, -1.60964714e-01, -3.02846039e-02]])
```

```
y
```

```
## 0      yes
## 1      yes
## 2      yes
## 3      yes
## 4      yes
```

```
##       ...
## 493     no
## 494     no
## 495     no
## 496     no
## 497     no
## Name: high_risk, Length: 498, dtype: object
```

Creating the Training Set and the Test Set

```
pip install -U scikit-learn
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=0)
```

```
X_train
```

```
## array([[ 5.17030000e+01,  3.66500000e+00,  1.60740000e+01, ...,
##         -5.51240948e-02,  4.38225490e-02,  4.69844543e-02],
##        [ 6.25630000e+01,  6.25600000e+00,  1.26160000e+01, ...,
##          3.24380005e-03, -5.07007084e-03,  1.37092245e-01],
##        [ 5.19500000e+01,  2.13220000e+01,  1.12320000e+01, ...,
##         -4.26169855e-02, -4.76777079e-02,  1.47106598e-01],
##        ...,
##        [ 5.61390000e+01,  5.35900000e+00,  1.20730000e+01, ...,
##         -1.30077928e-01, -1.92233451e-01, -1.90603826e-01],
##        [ 3.66050000e+01,  1.20428000e+02,  4.18990000e+01, ...,
##          3.67339872e-01,  2.32576458e-01,  2.87177075e-01],
##        [ 3.97470000e+01,  2.69020000e+01,  1.76450000e+01, ...,
##         -1.08140694e-02,  8.94605090e-02,  8.48513966e-02]])
```

```
X_test
```

```
## array([[ 2.15350000e+01,  1.07000000e+01,  2.88580000e+01, ...,
##          3.49475701e-01, -7.92837190e-02,  1.89132709e-02],
##        [ 1.50396000e+02,  4.70600000e+00,  1.32100000e+01, ...,
##         -1.58693792e-01, -1.36054269e-01, -1.30933517e-01],
##        [ 2.80580000e+01,  3.37500000e+00,  1.11960000e+01, ...,
##          1.27460670e-02, -2.99917507e-02, -8.37254155e-02],
##        ...,
##        [ 8.18960000e+01,  5.00300000e+00,  1.33460000e+01, ...,
##          1.81134155e-01,  3.15643118e-02, -9.30597999e-02],
##        [ 7.71620000e+01,  5.57900000e+00,  9.53900000e+00, ...,
##         -9.91121617e-02, -1.52009345e-01, -2.56797654e-01],
##        [ 8.11150000e+01,  4.45300000e+00,  9.16800000e+00, ...,
##         -7.75943903e-02, -1.32675093e-01,  3.64093853e-02]])
```

```
y_train
```

```
## 107     no
## 397     no
## 71      no
## 482     no
## 6      yes
##       ...
## 323     no
## 192     no
## 117     no
## 47      no
## 172     no
## Name: high_risk, Length: 398, dtype: object
```

```
y_test
```

```
## 90     yes
## 254    yes
## 283     no
## 443    yes
## 336    yes
##       ...
## 391    yes
## 56      no
## 438    yes
## 60      no
## 208     no
## Name: high_risk, Length: 100, dtype: object
```

Feature Scaling: a transformation applied to columns

- Feature scaling is never applied across columns

	sample id	MYCN	STC1	...	CHIP_HIF1A_300	CHIP_EPAS1_300	high_risk
0	GSM1205238	42.499	2.948	...	-0.306735	-0.319118	yes
1	GSM1205239	25.488	3.088	...	0.002388	-0.201904	yes
2	GSM1205240	883.223	6.025	...	0.061273	-0.174953	yes
3	GSM1205241	433.333	3.091	...	0.131030	0.057692	yes
4	GSM1205242	854.354	10.856	...	-0.132852	-0.133280	yes
5	GSM1205243	111.049	9.317	...	0.059308	-0.070998	no
6	GSM1205244	63.553	6.560	...	-0.265630	-0.114986	yes
7	GSM1205245	1558.482	58.680	...	-0.066774	-0.146944	yes
8	GSM1205246	726.585	16.095	...	0.138918	0.102813	yes
9	GSM1205247	1234.037	4.240	...	0.295862	0.094235	yes

Figure 4: Features that contribute to probability function.

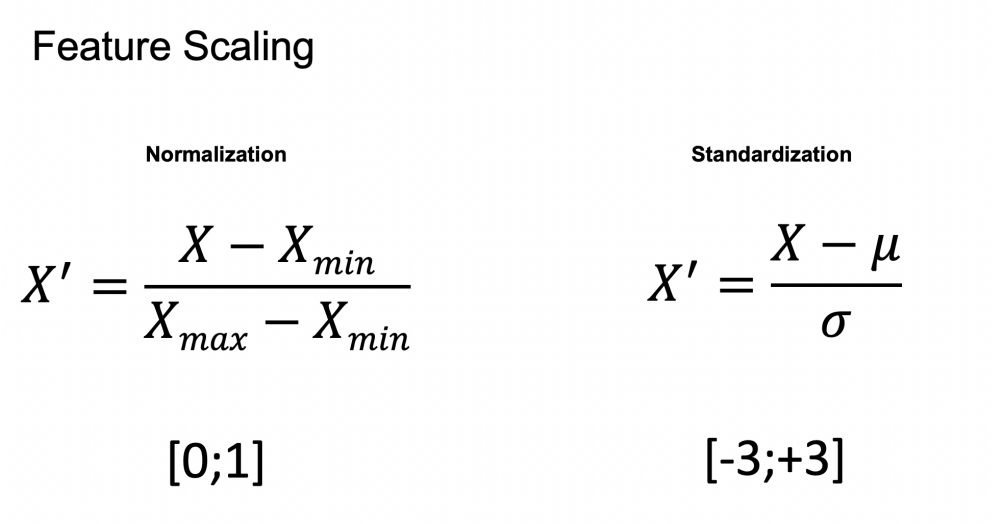


Figure 5: The two main feature scaling methods are Normalization and Standardization.

Feature Scaling the Feature Array

```
from sklearn.preprocessing import StandardScaler
# Standardization based on the features of the whole dataset (?)
# Compute in the training set (?)
# instance of the class
sc = StandardScaler()
# compute average and sd of the features
# Takes on the array of independent variables you want to scale
sc.fit_transform(X_train)
# We will only need the new array of independent variables in the training set
```

```
## array([[-0.41489262, -0.62102922,  0.18920614, ..., -0.16252926,
##          0.64150125,  0.43541129],
##        [-0.39033589, -0.46701445, -0.33651953, ...,  0.13228874,
##          0.22814504,  1.04548857],
##        [-0.4143341 ,  0.4285419 , -0.54693142, ..., -0.09935547,
##         -0.13207561,  1.11329103],
##        ...,
##        [-0.4048619 , -0.52033411, -0.41907275, ..., -0.54112334,
##         -1.35420309, -1.1731867 ],
##        [-0.44903236,  6.31962162,  4.11542517, ...,  1.97134904,
##          2.23729628,  2.06164204],
##        [-0.44192764,  0.76022943,  0.42804797, ...,  0.06128202,
##          1.02734136,  0.69179047]])
```

```
X_train = sc.fit_transform(X_train)
```

Part 2 - Building and training the model

Building the model

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(random_state = 0 )
```

Training the model

```
model.fit(X_train, y_train)
```

```
## LogisticRegression(random_state=0)
```

Access coefficients and variable importance

```
coefficients = model.coef_[0]
coefficients
```

```
## array([ 2.70252999,  0.86574185,  0.17201298, -0.54226061, -0.27969284,
##         0.48759667,  0.50566359,  0.01271285, -0.51483966,  0.12173305,
##        -0.81402512, -1.16523226,  0.28565474, -0.53312979,  0.60379161,
##        -0.16074749, -0.20371404, -0.11058777, -0.21464494,  0.6959539 ,
##        -0.29853096, -0.76650485])
```

Plot variable importance

Deal with Variable Importance in Kocak dataset. Get features from feature dataframe using the column method

```
X_test_kocak = dataset_kocak.drop('high_risk', axis=1)
X_test_kocak = X_test_kocak.drop('sample id', axis=1)
X_test_kocak
```

```
##         MYCN    STC1  ...  CHIP_HIF1A_300  CHIP_EPAS1_300
## 0     42.499   2.948  ...        -0.306735       -0.319118
## 1     25.488   3.088  ...         0.002388       -0.201904
## 2    883.223   6.025  ...         0.061273       -0.174953
## 3    433.333   3.091  ...         0.131030        0.057692
## 4    854.354  10.856  ...        -0.132852       -0.133280
## ..       ...     ...  ...              ...             ...
## 493   40.864  10.966  ...         0.000186        0.016204
## 494   65.579  19.896  ...        -0.098292       -0.037657
## 495   55.796  10.633  ...         0.130951        0.267928
## 496   67.211   6.510  ...        -0.148956       -0.155719
## 497   40.836  29.188  ...        -0.160965       -0.030285
##
## [498 rows x 22 columns]
```

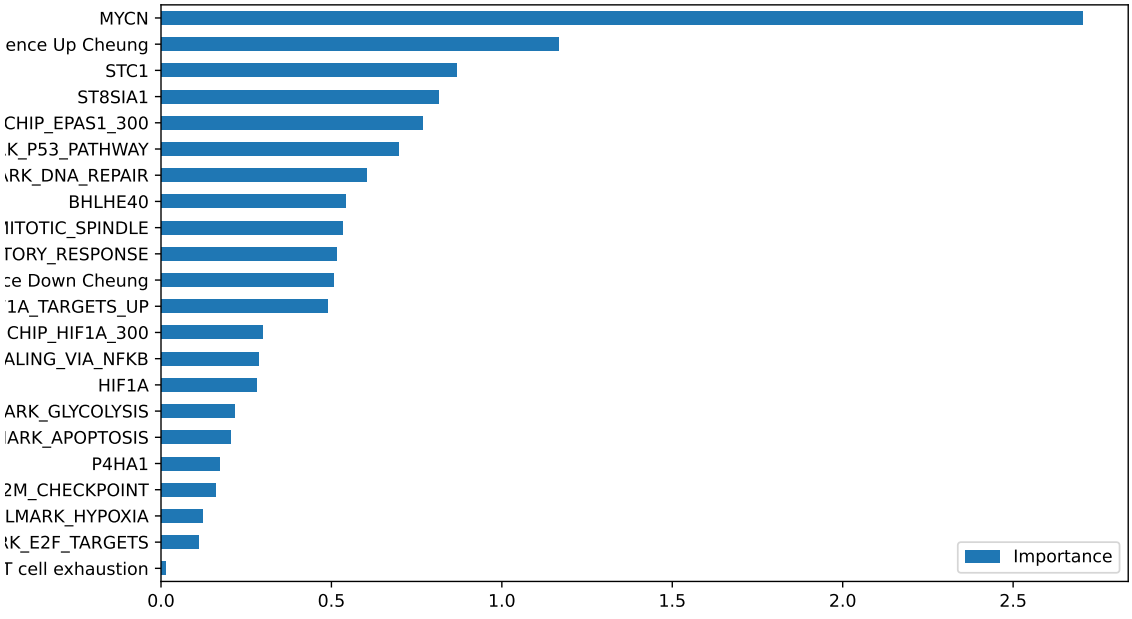
```
X_test_kocak.columns
```

```
## Index(['MYCN', 'STC1', 'P4HA1', 'BHLHE40', 'HIF1A', 'ELVIDGE_HIF1A_TARGETS_UP',
##        'Quiescence Down Cheung', 'T cell exhaustion',
##        'HALLMARK_INFLAMMATORY_RESPONSE', 'HALLMARK_HYPOXIA', 'ST8SIA1',
##        'Quiescence Up Cheung', 'HALLMARK_TNFA_SIGNALING_VIA_NFKB',
```

```
##        'HALLMARK_MITOTIC_SPINDLE', 'HALLMARK_DNA_REPAIR',
##        'HALLMARK_G2M_CHECKPOINT', 'HALLMARK_APOPTOSIS', 'HALLMARK_E2F_TARGETS',
##        'HALLMARK_GLYCOLYSIS', 'HALLMARK_P53_PATHWAY', 'CHIP_HIF1A_300',
##        'CHIP_EPAS1_300'],
##       dtype='object')
```

```
import matplotlib
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

feature_importance = pd.DataFrame({'Feature': X_test_kocak.columns, 'Importance': np.abs(coefficients)}
# feature_importance = feature_importance.sort_values('Importance', ascending=True).head(70)
feature_importance = feature_importance.sort_values('Importance', ascending=True)
# feature_importance = feature_importance[:5000]
feature_importance.plot(x='Feature', y='Importance', kind='barh', figsize=(10, 6))
```



Inference

Predictions for the test set and for a particular patient

```
y_pred = model.predict(sc.transform(X_test)) # First, call the scaler object
```

Prediction in our model for all patients in the test set

```
y_pred
```

```
## array(['yes', 'no', 'no', 'yes', 'no', 'no', 'no', 'no', 'no', 'no',
##        'yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'yes', 'no', 'no',
##        'no', 'no', 'yes', 'no', 'no', 'yes', 'no', 'no', 'no', 'no', 'no',
##        'yes', 'no', 'yes', 'yes', 'no', 'no', 'yes', 'no', 'no', 'yes',
##        'no', 'no', 'yes', 'no', 'yes', 'no', 'no', 'no', 'no', 'yes',
##        'no', 'no', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes', 'no',
##        'no', 'no', 'no', 'no', 'yes', 'no', 'yes', 'yes', 'no', 'no',
##        'no', 'yes', 'yes', 'yes', 'yes', 'yes', 'no', 'yes', 'no', 'no',
##        'yes', 'yes', 'no', 'no', 'yes', 'no', 'no', 'no', 'no', 'no',
##        'yes', 'yes', 'no', 'no', 'yes', 'no', 'yes', 'no', 'no'],
##       dtype=object)
```

Prediction in our model for one patient

```
model.predict(sc.transform([[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
```

```
## array(['no'], dtype=object)
```

Part 3: Evaluating the Model

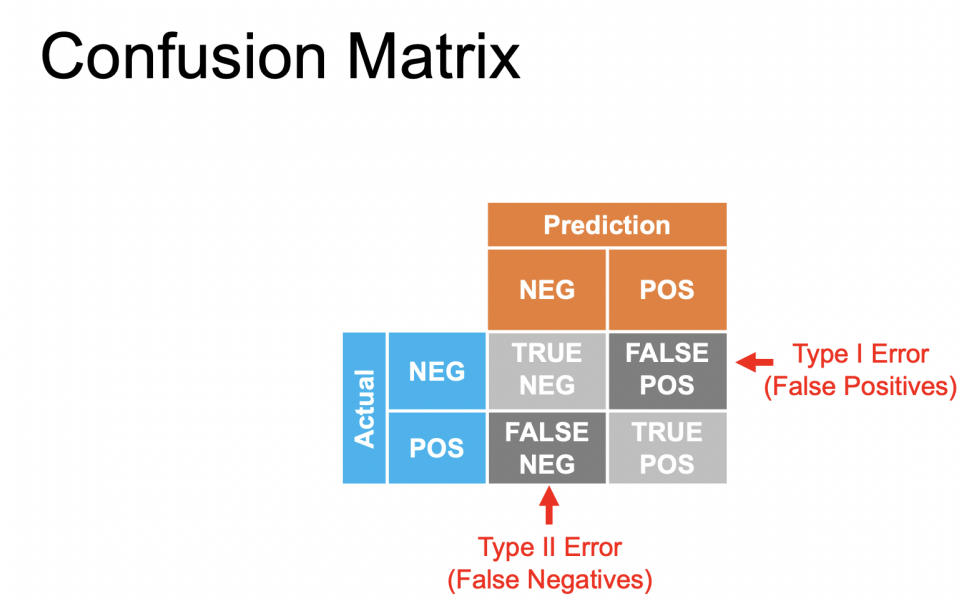


Figure 6: The construction of a confusion matrix.

To calculate the confusion matrix, we need the vector of ground-truth and the vector of predictions.

ground-truth vector

```
y_test
```

```
## 90     yes
## 254    yes
## 283     no
## 443    yes
## 336    yes
##       ...
## 391    yes
## 56      no
## 438    yes
## 60      no
## 208     no
## Name: high_risk, Length: 100, dtype: object
```

prediction vector

```
y_pred
```

```
## array(['yes', 'no', 'no', 'yes', 'no', 'no', 'no', 'no', 'no', 'no',
##        'yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'yes', 'no', 'no',
##        'no', 'no', 'yes', 'no', 'no', 'yes', 'no', 'no', 'no', 'no', 'no',
##        'yes', 'no', 'yes', 'yes', 'no', 'no', 'yes', 'no', 'no', 'yes',
##        'no', 'no', 'yes', 'no', 'yes', 'no', 'no', 'no', 'no', 'yes',
##        'no', 'no', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes', 'no',
##        'no', 'no', 'no', 'no', 'yes', 'no', 'yes', 'yes', 'no', 'no',
##        'no', 'yes', 'yes', 'yes', 'yes', 'yes', 'no', 'yes', 'no', 'no',
##        'yes', 'yes', 'no', 'no', 'yes', 'no', 'no', 'no', 'no', 'no',
##        'yes', 'yes', 'no', 'no', 'yes', 'no', 'yes', 'no', 'no'],
##       dtype=object)
```

Construct confusion matrix

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

```
## array([[55,  8],
##        [ 9, 28]])
```

Acuracy

Acuracy = (number of correct predictions)/(total number of observations)

Manually calculate acuracy

```
(55+28)/(55+28+9+8)
```

```
## 0.83
```

Calculate acuracy using sklearn

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
## 0.83
```

References

Session Info

```
sessionInfo()
```

```
## R version 4.1.1 (2021-08-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] reticulate_1.31
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.10     lattice_0.21-8  png_0.1-8       withr_2.5.0
##  [5] digest_0.6.32   grid_4.1.1      jsonlite_1.8.7  evaluate_0.21
##  [9] highr_0.10      rlang_1.1.1     cli_3.6.1       rstudioapi_0.14
## [13] Matrix_1.5-1    rmarkdown_2.22  tools_4.1.1     xfun_0.39
## [17] yaml_2.3.7      fastmap_1.1.1   compiler_4.1.1  htmltools_0.5.5
## [21] knitr_1.43
```