Universidade Presbiteriana Mackenzie

Lab: Métricas, Regressão Logística & K-vizinhos mais Próximos

Referências e Materiais úteis para este Lab: *ver notas de aula de regressão logística e K-vizinhos mais Próximos*. Acesse também https://en.wikipedia.org/wiki/Precision_and_recall.

Caso: **Predição de Diagnósticos a partir de Dados de Imagens**

https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

Os dados acima são características computadas a partir de uma imagem digitalizada por *agulha fina* (PAAF) de uma massa mamária. Eles descrevem características dos núcleos celulares presentes na imagem e classificam os tumores como malignos ou benignos.

Classificação

Breast Cancer Data

diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean
M	15.340	14.26	102.50	704.4
B	12.880	28.92	82.50	514.3
M	17.080	27.15	111.20	930.9
B	16.140	14.86	104.30	800.0
M	13.480	20.82	88.40	559.2
B	14.470	24.99	95.81	656.4
B	12.490	16.85	79.19	481.6
M	23.210	26.97	153.50	1670.0
B	11.620	18.18	76.38	408.8
B	9.787	19.94	62.11	294.5
M	21.750	20.99	147.30	1491.0
B	10.800	21.98	68.79	359.9
M	25.730	17.46	174.20	2010.0
B	11.870	21.54	76.83	432.0
B	7.691	25.44	48.34	170.4

y ← X...

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

Set Up Lab (*run this before continue*)

Mostrar código

E você empregará os dados já tratados no set up acima, com os dataframes `breast` e `new_breast`. A idea aqui é empregar os dados de `breast` para estimar as classes dos novos casos desconhecidos de `new_breast`.

```
breast.head()
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
106	91505	12.540	16.32	81.25	476.3	0.1158
500	90524101	17.990	20.66	117.80	991.7	0.1036
68	915186	9.268	12.87	61.49	248.7	0.1634
515	897880	10.050	17.53	64.41	310.8	0.1007
138	881046502	20.580	22.14	134.70	1290.0	0.0909

5 rows × 32 columns

```
new_breast.head()
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
215	904689	12.96	18.29	84.18	525.2	0.07351
39	902975	12.21	14.09	78.78	462.0	0.08108
99	871201	19.59	18.15	130.70	1214.0	0.11200
70	846226	19.17	24.80	132.40	1123.0	0.09740
264	90769601	11.13	16.62	70.47	381.1	0.08151

5 rows × 32 columns

Exercício

Empregue a regressão logística (`max_iter=10000`) para classificar os casos como malignos ou benignos.

- Empregue os seguintes parâmetros para separação dos conjuntos de treinamento e teste `test_size=0.3, stratify=y, random_state=1`.
- Observe, ao menos um atributo não pode ser empregado no treinamento.
- Faça a normalização dos dados (por simplicidade, sendo todos valores positivos, divida o valor de cada atributo pelo maior da coluna).
- Não empregue outros parâmetros não informados aqui pois os resultados podem divergir.

Analise o desempenho do modelo obtido e em seguida empregue para determinar o diagnóstico provável dos novos casos (`new_breast`).

(*) **Atenção:** *Na predição, não esqueça de aplicar a *mesma* transformação (a normalização) empregada durante o treinamento!*

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Entradas e Saídas
X = breast.drop(columns=['id','diagnosis'])
X = X / X.max()
y = breast['diagnosis']

# Separando conjuntos de Treinamento e Teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y, rand

# Definição do modelo
clf = LogisticRegression(max_iter=10000)

# Treinamento
clf.fit(X_train,y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=10000)
```

```
# Avaliação
y_pred = clf.predict(X_test)

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

cm = confusion_matrix(y_test, y_pred)
print('\nMatriz de Confusão:\n')
print(cm)
print()

cm_df = pd.DataFrame(cm,index=[clf.classes_],columns=[clf.classes_])
display(cm_df)
print()

accuracy = accuracy_score(y_test, y_pred)
print('\nScore de Acuracidade (1):\n')
print(f'{accuracy:.4f}')

accuracy = clf.score(X_test, y_test)
print('\nScore de Acuracidade (2):\n')
print(f'{accuracy:.4f}')

print('\nClassification Report:\n')
print(classification_report(y_test, y_pred))
```

```
Matriz de Confusão:

[[82  0]
```

Q1. Qual a acuracidade geral do modelo?

0.9769

Q2. Qual a precisão obtida para classe M?

1.00

Q3. Qual classe teve Falso Positivos? Qual o percentual de Falso Positivos para essa classe?

B, 0.04

Q4. Qual classe não teve todos as instâncias identificadas corretamente? Qual o percentual de Falso Negativos para essa classe?

M, 0.06

Q5. Quantos casos benignos e malignos classificados nos novos casos?

71 e 37

```
           B       0.96      1.00      0.98        82
```

$$Precisão = \frac{TP}{TP + FP}$$

```
   macro avg       0.98      0.97      0.97       130
```

```
print(f'Precisão de B = {cm_df.loc["B"]["B"].values[0][0]} / ({cm_df.loc["B"]["B"].values
```

```
Precisão de B = 82 / (82 + 3) = 0.96
```

```
print(f'Precisão de M = {cm_df.loc["M"]["M"].values[0][0]} / ({cm_df.loc["M"]["M"].values
```

```
Precisão de M = 45 / (45 + 0) = 1.00
```

$$Recall = \frac{TP}{TP + FN}$$

```
print(f'Recall de B = {cm_df.loc["B"]["B"].values[0][0]} / ({cm_df.loc["B"]["B"].values[0
```

```
Recall de B = 82 / (82 + 0) = 1.00
```

```
print(f'Recall de M = {cm_df.loc["M"]["M"].values[0][0]} / ({cm_df.loc["M"]["M"].values[0
```

```
Recall de M = 45 / (45 + 3) = 0.94
```

$$F1 = \frac{2}{\frac{1}{Precisão} + \frac{1}{Recall}}$$

```
print(f'F1 de B = 2 / (1/0.96 + 1/1) = { 2 / (1/0.96 + 1/1) :.2f}')
```

```
F1 de B = 2 / (1/0.96 + 1/1) = 0.98
```

```
print(f'F1 de M = 2 / (1/1 + 1/0.94) = { 2 / (1/1 + 1/0.94) :.2f}')
```

```
F1 de M = 2 / (1/1 + 1/0.94) = 0.97
```

Atenção, aplique a mesma transformação empregada sobre o primeiro conjunto!

```
# Novos casos
X_new = new_breast.drop(columns=['id','diagnosis'])
X_new = X_new / breast.drop(columns=['id','diagnosis']).max()
y_pred = clf.predict(X_new)

new_breast.diagnosis = y_pred

new_breast.diagnosis.value_counts()
```

```
B    71
M    37
Name: diagnosis, dtype: int64
```

Exercício

Empregue agora o modelo de K-vizinhos mais próximos, para k=3 e k=5 e analise o desempenho de cada modelo. Em seguida empregue o melhor modelo Knn para determinar o diagnóstico provável dos novos casos.

- Empregue os mesmos dados de treinamento e teste empregados antes.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# Entradas e Saídas
# X = breast.drop(columns=['id','diagnosis'])
# X = X / X.max()
# y = breast['diagnosis']

# Separando conjuntos de Treinamento e Teste
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y, ra

# Definição do modelo
clf = KNeighborsClassifier(n_neighbors = 3)

# Treinamento
clf.fit(X_train,y_train)

# Avaliação
y_pred = clf.predict(X_test)

print(clf)
print()

cm = confusion_matrix(y_test, y_pred)
print('\nMatriz de Confusão:\n')
print(cm)
print()

cm_df = pd.DataFrame(cm,index=[clf.classes_],columns=[clf.classes_])
display(cm_df)
print()

accuracy = clf.score(X_test, y_test)
print('\nScore de Acuracidade:\n')
print(f'{accuracy:.4f}')

print('\nClassification Report:\n')
print(classification_report(y_test, y_pred))
```

Q6. Qual o melhor k e a acuracidade do melhor modelo Knn?

3 e 0.9923

Q7. Para k=5, há quantos Falso Positivos e Falsos Negativos da classe B?

1 e 1

Q8. Para k=3, há quantos Falso Positivos e Falsos Negativos da classe B no melhor modelo Knn?

1 e 0

Q9. Para k=3, há quantos Falso Positivos e Falsos Negativos da classe M no melhor modelo Knn?

0 e 1

Q10. Podemos concluir desses testes que modelos K-vizinhos mais Próximos são melhores que modelos de Regressão Logística.

FALSO

Q11. Podemos concluir desses testes que quanto maior o K, melhores os modelos de K-vizinhos mais Próximos.

FALSO

Q12. Quantos casos benignos e malignos classificados nos novos casos com o melhor modelo de Knn?

67 e 41

```
# Novos casos
X_new = new_breast.drop(columns=['id','diagnosis'])
X_new = X_new / breast.drop(columns=['id','diagnosis']).max()
y_pred = clf.predict(X_new)

new_breast.diagnosis = y_pred

new_breast.diagnosis.value_counts()
```

```
B    67
M    41
Name: diagnosis, dtype: int64
```