Lab Multi Layer Perceptron

Neste exercício você vai empregar redes MLP do `scikit-learn` para resolver um problema de classificação sobre a base de dados `penguins`.
Siga o modelo e em seguida responda o questionário no Moodle.

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

Modelo MLP

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report
```

Preparação dos dados

1. Aquisição dos Dados
2. Seleção de Atributos e Instâncias (*nulos?*)
3. Hot encode dos dados
4. Normalização

```
# Carregando os dados
df = pd.read_csv('https://vincentarelbundock.github.io/Rdatasets/csv/MASS/Cars93.csv',index_col=0)
df = df.reset_index(drop=True)
display(df.head())

# Separando preditores e classe objetivo
X = df[['MPG.city', 'Horsepower', 'Type', 'Price', 'Width', 'Length', 'Weight', 'EngineSize']]
y = df['Origin']

# Aplicando one-hot encoding em 'Origin'
encoder = OneHotEncoder()
X_encoded = encoder.fit_transform(X[['Type']]).toarray()
X_encoded = pd.DataFrame(X_encoded, columns=encoder.get_feature_names_out())
X = pd.concat([X.drop(columns='Type'), X_encoded], axis=1)
display(X.head())

# Normalizando os dados
scaler = StandardScaler()
X = scaler.fit_transform(X)
display(pd.DataFrame(X).head())
```

	Manufacturer	Model	Type	Min.Price	Price	Max.Price	MPG.city	MPG.highway	AirBags	DriveTrain	...	Passengers	Length
0	Acura	Integra	Small	12.9	15.9	18.8	25	31	None	Front	...	5	177
1	Acura	Legend	Midsize	29.2	33.9	38.7	18	25	Driver & Passenger	Front	...	5	195
2	Audi	90	Compact	25.9	29.1	32.3	20	26	Driver only	Front	...	5	180
3	Audi	100	Midsize	30.8	37.7	44.6	19	26	Driver & Passenger	Front	...	6	193
4	BMW	535i	Midsize	23.7	30.0	36.2	22	30	Driver only	Rear	...	4	186

5 rows × 27 columns

	MPG.city	Horsepower	Price	Width	Length	Weight	EngineSize	Type_Compact	Type_Large	Type_Midsize	Type_Small	Type_Sporty	T
0	25	140	15.9	68	177	2705	1.8	0.0	0.0	0.0	1.0	0.0	
1	18	200	33.9	71	195	3560	3.2	0.0	0.0	1.0	0.0	0.0	

Treinamento do Modelo

1. Separação dos conjuntos de Treinamento e Teste
2. Definição do modelo e seus *Hiperparâmetros*
3. Treinamento do Modelo

1 -0.781032 1.078322 1.497844 0.431983 0.812171 0.830208 0.515869 -0.455842 -0.36626 1.796461 -0.540062 -0.420969 -0.327327

```
# Separando os dados de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

# Define o modelo de classificação (MLP)
clf = MLPClassifier(hidden_layer_sizes=(8,16,8), random_state=1, max_iter=5000)

# Treinando o modelo
clf.fit(X_train, y_train)
```

```
MLPClassifier(hidden_layer_sizes=(8, 16, 8), max_iter=5000, random_state=1)
```

Calculo da Eficiência do Modelo

1. Acuracidade
2. Classification Report
    1. Precision
    2. Recall
    3. F1

```
# Calculando métricas de avaliação
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print("Acurácia:", accuracy)
print("Classification Report:\n", classification_rep)
```

```
Acurácia: 0.8571428571428571
Classification Report:
               precision    recall  f1-score   support

          USA       0.82      0.82      0.82        11
      non-USA       0.88      0.88      0.88        17

     accuracy                           0.86        28
    macro avg       0.85      0.85      0.85        28
 weighted avg       0.86      0.86      0.86        28
```

Predição

Faça a predição dos seguintes veículos em `X_new`:

Execute para gerar `X_new`

Mostrar código

```
display(X_new)
```

	MPG.city	Horsepower	Type	Price	Width	Length	Weight	EngineSize
0	22	105	Sporty	15.9	68	180	2850	2.3
1	24	140	Compact	17.5	67	185	3040	2.2

Atenção: Antes de aplicar a predição é necessário aplicar as mesmas transformações empregadas antes durante o treinamento.

```
X_encoded = encoder.transform(X_new[['Type']]).toarray()
X_encoded = pd.DataFrame(X_encoded, columns=encoder.get_feature_names_out())
X_new = pd.concat([X_new.drop(columns='Type'), X_encoded], axis=1)
display(X_new.head())

X_new = scaler.transform(X_new)

y_pred = clf.predict(X_new)

print(y_pred)
```

	MPG.city	Horsepower	Price	Width	Length	Weight	EngineSize	Type_Compact	Type_Large	Type_Midsize	Type_Small	Type_Sporty	T
0	22	105	15.9	68	180	2850	2.3	0.0	0.0	0.0	0.0	1.0	
1	24	140	17.5	67	185	3040	2.2	1.0	0.0	0.0	0.0	0.0	

```
['USA' 'non-USA']
```

Exercício

Empregue os modelos de rede neural MLP abaixo para classifique a origem, ilha do Pinguim, baseado nos demais atributos.

```
df = sns.load_dataset('penguins')
df.head()
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female

Preparação dos Dados

Exclua os dados ausentes se houverem. Para o atributo `species` aplique o One Hot encode. Para o atributo `sex` faça o Label encode, atribuindo 0 para o masculino e 1 para femino.

`Dropna`

```
df.isnull().sum()
```

```
species              0
island               0
bill_length_mm       2
bill_depth_mm        2
flipper_length_mm    2
body_mass_g          2
sex                 11
dtype: int64
```

```
df = df.dropna().reset_index(drop=True)
```

```
len(df)
```

```
333
```

Hot Encode

Faça o hot enconde de `species` empregando:

```
from sklearn.preprocessing import OneHotEncoder
```

```
from sklearn.preprocessing import OneHotEncoder, LabelEncoder

hot_encode = OneHotEncoder(handle_unknown='ignore')
hot_encode = hot_encode.fit(df[['species']])
transformed = hot_encode.transform(df[['species']]).toarray()
# print(transformed)

transformed_df = pd.DataFrame(transformed, columns=hot_encode.get_feature_names_out())
display(transformed_df.head())

df = pd.concat([df, transformed_df],axis=1)
df.head()
```

	species_Adelie	species_Chinstrap	species_Gentoo
0	1.0	0.0	0.0
1	1.0	0.0	0.0
2	1.0	0.0	0.0
3	1.0	0.0	0.0
4	1.0	0.0	0.0

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0
3	Adelie	Torgersen	36.7	19.3	193.0	3450.0
4	Adelie	Torgersen	39.3	20.6	190.0	3650.0

```
df.species_Adelie.sum()
```

```
146.0
```

Label Encode

Faça o Label Encode de `sex` empregando a função `replace()` do `Pandas`. Atribua 0 para `Male` e 1 para `Female`.

```
df.sex = df.sex.replace('Male',0)
df.sex = df.sex.replace('Female',1)
```

```
df.head()
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0
3	Adelie	Torgersen	36.7	19.3	193.0	3450.0
4	Adelie	Torgersen	39.3	20.6	190.0	3650.0

```
df.sex.sum()
```

```
165
```

Normalização

Empregue:

```
from sklearn.preprocessing import MinMaxScaler
```

```
from sklearn.preprocessing import MinMaxScaler

# Entradas e Saídas
X = df.drop(columns=['island','species'])
y = df['island']

columns = X.columns

scaler = MinMaxScaler()
scaler.fit(X)
X = scaler.transform(X)
```

```
pd.DataFrame(X, columns = columns)
```

	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	species_Adel
0	0.254545	0.666667	0.152542	0.291667	0.0	
1	0.269091	0.511905	0.237288	0.305556	1.0	
2	0.298182	0.583333	0.389831	0.152778	1.0	
3	0.167273	0.738095	0.355932	0.208333	1.0	
4	0.261818	0.892857	0.305085	0.263889	0.0	
...	...	...	...	...	...	
328	0.549091	0.071429	0.711864	0.618056	1.0	
329	0.534545	0.142857	0.728814	0.597222	1.0	
330	0.665455	0.309524	0.847458	0.847222	0.0	
331	0.476364	0.202381	0.677966	0.694444	1.0	
332	0.647273	0.357143	0.694915	0.750000	0.0	

333 rows × 8 columns

```
X.sum()
```

```
1106.048069435273
```

Treinamento 1

Empregue os atributos normalizados da seção anterior como variáveis preditoras para implemente um algoritmo MLP para classificação da ilha (`island`) de origem dos pinguins.

1. Train/Test Split. Empregue 30% dos dados para teste, `random_state=1` e dados estratificados pelo atributo classe (variável objetivo).
2. MLP. Empregue `random_state=1`, uma rede com camadas ocultas de 8, 16 e 8 neurônios respectivamente e número máximo de iterações 1000.

Não empregue outros parâmetros além dos solicitados.

Obtenha ao final a acuracidade do modelo, o `classification report` e a matriz de confusão para responder as questões.

```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split

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

# Separação Treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.3, random_state=1)

# Definição
clf = MLPClassifier(random_state=1, hidden_layer_sizes=(8,16,8), max_iter=1000)

# Treinamento
clf.fit(X_train, y_train)

# Avaliação
y_pred = clf.predict(X_test)

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

print('\nConfusion Matrix:\n')
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:

              precision    recall  f1-score   support

      Biscoe       0.73      0.78      0.75        49
       Dream       0.61      0.76      0.67        37
   Torgersen       0.50      0.07      0.12        14

    accuracy                           0.67       100
   macro avg       0.61      0.53      0.52       100
weighted avg       0.65      0.67      0.64       100


Confusion Matrix:

[[38 10  1]
 [ 9 28  0]
 [ 5  8  1]]
```

```
clf.coefs_[0].sum()
```

```
-2.244783703253475
```

Treinamento 2

Altere o modelo anterior para uma rede com 8, 4 elementos de camadas internas, a função de ativação `relu` e o máximo número de iterações 5000. Empregue o mesmo conjunto de Treinamento e Teste anterior.

```
# Definição
clf = MLPClassifier(random_state=1, hidden_layer_sizes=(8,4), max_iter=5000, activation='relu')
# Treinamento
clf.fit(X_train, y_train)

# Avaliação
y_pred = clf.predict(X_test)

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

print('\nConfusion Matrix:\n')
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:

              precision    recall  f1-score   support

      Biscoe       0.92      0.73      0.82        49
       Dream       0.61      0.89      0.73        37
   Torgersen       0.29      0.14      0.19        14

    accuracy                           0.71       100
   macro avg       0.61      0.59      0.58       100
weighted avg       0.72      0.71      0.70       100
```

```
Confusion Matrix:

[[36 11  2]
 [ 1 33  3]
 [ 2 10  2]]
```

```
clf.coefs_[0].sum()
```

```
-0.8648101761509308
```

Predição

Faça a predição dos seguinte pinguim:

```
species                Adelie
bill_length_mm           40.0
bill_depth_mm            18.0
flipper_length_mm       185.0
body_mass_g            3900.0
sex                      Male
```

```
X_new = pd.DataFrame()

X_new['species']=['Adelie']

X_new['bill_length_mm']=40.0
X_new['bill_depth_mm']=18.0
X_new['flipper_length_mm']=185.0
X_new['body_mass_g']=3900.0
X_new['sex']='Male'

X_new.head()
```

	species	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	40.0	18.0	185.0	3900.0	Male

```
X_encoded = hot_encode.transform(X_new[['species']]).toarray()
X_encoded = pd.DataFrame(X_encoded, columns=hot_encode.get_feature_names_out())
X_new = pd.concat([X_new.drop(columns='species'), X_encoded], axis=1)
display(X_new.head())

X_new.sex = X_new.sex.replace('Male',0)
X_new.sex = X_new.sex.replace('Female',1)

X_new = scaler.transform(X_new)

y_pred = clf.predict(X_new)

print(y_pred)
```

	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	species_Adelie	species_Chinstrap	species_Gentoo
0	40.0	18.0	185.0	3900.0	Male	1.0	0.0	0.0

```
['Dream']
```

```
clf.classes_
```

```
array(['Biscoe', 'Dream', 'Torgersen'], dtype='<U9')
```

```
clf.predict_proba(X_new)
```

```
array([[0.36003115, 0.38667105, 0.25329781]])
```