Tarefa#4 - Aprendizado supervisionado - MO432

July 1, 2021

Universidade Estadual de Campinas (UNICAMP), Instituto de Computação (IC)

Prof. Jacques Wainer, 2021s1

[ ]:
```
# RA & Name
print('264965: ' + 'Décio Luiz Gazzoni Filho')
print('265673: ' + 'Gabriel Luciano Gomes')
print('192880: ' + 'Lucas Borges Rondon')
```

```
264965: Décio Luiz Gazzoni Filho
265673: Gabriel Luciano Gomes
192880: Lucas Borges Rondon
```

1 Leitura da base de dados

[ ]:
```
import pandas as pd
import requests
import io

url = "https://www.ic.unicamp.br/~wainer/cursos/1s2021/432/dados4.csv"
s = requests.get(url).content

db = pd.read_csv(io.StringIO(s.decode('utf-8')))
db.head()
```

[ ]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
0	1	22.08	11.46	2	k	bb	1.585	0	0	0	1	g	100	1213	0
1	0	22.67	7.00	2	c	bb	0.165	0	0	0	0	g	160	1	0
2	0	29.58	1.75	1	k	bb	1.250	0	0	0	1	g	280	1	0
3	0	21.67	11.50	1	j	j	0.000	1	1	11	1	g	0	1	1
4	1	20.17	8.17	2	aa	bb	1.960	1	1	14	0	g	60	159	1

[ ]:
```
print(f'A base de dados possui {db.shape[0]} instâncias com {db.shape[1]}␣
↪atributos.')
```

```
A base de dados possui 690 instâncias com 15 atributos.
```

2 Pré-processamento dos dados

2.1 Identificando e removendo valores faltantes (caso exista)

[ ]:
```
# Verificar se existem valores desconhecidos na base de dados
db.isna().sum()
```

[ ]:
```
V1      0
V2      0
V3      0
V4      0
V5      0
V6      0
V7      0
V8      0
V9      0
V10     0
V11     0
V12     0
V13     0
V14     0
V15     0
dtype: int64
```

2.2 Separação em X/y

[ ]:
```
y = db['V15']
db.drop('V15', axis=1, inplace=True)
X = db
```

[ ]:
```
X.shape, y.shape
```

[ ]:
```
((690, 14), (690,))
```

2.3 Conversão dos dados categórios para numéricos

[ ]:
```
X = pd.get_dummies(X, prefix=['V5', 'V6', 'V12'])
X.head()
```

[ ]:

	V1	V2	V3	V4	V7	V8	…	V6_o	V6_v	V6_z	V12_g	V12_p	V12_s
0	1	22.08	11.46	2	1.585	0	…	0	0	0	1	0	0
1	0	22.67	7.00	2	0.165	0	…	0	0	0	1	0	0
2	0	29.58	1.75	1	1.250	0	…	0	0	0	1	0	0
3	0	21.67	11.50	1	0.000	1	…	0	0	0	1	0	0
4	1	20.17	8.17	2	1.960	1	…	0	0	0	1	0	0

[5 rows x 36 columns]

2.4 Scaling e Centering dos dados

[ ]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled
```

[ ]:
```
array([[ 0.68873723, -0.80105183,  1.34711063, …,  0.32249031,
        -0.10830607, -0.30007898],
       [-1.45193254, -0.75124044,  0.45054795, …,  0.32249031,
        -0.10830607, -0.30007898],
       [-1.45193254, -0.16785619, -0.60482292, …,  0.32249031,
        -0.10830607, -0.30007898],
       …,
       [-1.45193254, -1.07543661,  0.96114643, …,  0.32249031,
        -0.10830607, -0.30007898],
       [-1.45193254, -0.35021653,  1.95822062, …,  0.32249031,
        -0.10830607, -0.30007898],
       [ 0.68873723,  0.79628971, -0.94857229, …, -3.10086836,
        -0.10830607,  3.33245602]])
```

3 Códigos auxiliares

[ ]:
```
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold,␣
 ↪train_test_split
from sklearn.utils._testing import ignore_warnings
from sklearn.exceptions import ConvergenceWarning
import numpy as np

def find_hyperparams(model, space, X, y, cv, n_iter = 10):
  search = RandomizedSearchCV(model,
                         space,
                         n_iter = n_iter,
                         scoring = 'roc_auc',
                         n_jobs = -1,
                         cv = cv)
  return search.fit(X,y)

def best_results(model, X, y, space, n_iter):

  score_outer = []

  for i in range(4):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
                                                        random_state=i)
```

```
    # inner score
    inner_cv = StratifiedKFold(n_splits=3)
    clf = find_hyperparams(model, space, X_train, y_train, inner_cv, n_iter)

    # outer score
    score = clf.score(X_test, y_test)

    score_outer.append(score)

  return np.mean(score_outer)

@ignore_warnings(category=ConvergenceWarning)
def compute_outer_score(models, X, y):

  model_results = []

  for name, model, space, n_iter in models:
    result = best_results(model, X, y, space, n_iter)
    model_results.append({'model': name,
                          'AUC'  : result})
  return pd.DataFrame(model_results)
```

4 Treino dos modelos

[ ]:
```
from sklearn.linear_model import LogisticRegression, RidgeClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis,␣
↪QuadraticDiscriminantAnalysis
from sklearn.svm import SVC
from scipy.stats import loguniform, uniform
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier

models = []

# MODEL LOGISTIC REGRESSION
models.append(('Logistic Regression',
              LogisticRegression(penalty='none'),
              {},
              1))

# MODEL LOGISTIC REGRESSION WITH L2 PENALTY
models.append(('Logistic Regression with L2 penalty',
              LogisticRegression(),
```

```
              {'C': loguniform(10**-3, 10**3)},
              10))

# MODEL LDA
models.append(('LDA',
              LinearDiscriminantAnalysis(),
              {},
              1))

# MODEL QDA
models.append(('QDA',
              QuadraticDiscriminantAnalysis(),
              {},
              1))

# MODEL SVC Linear
models.append(('SVC Linear',
              SVC(kernel='linear', max_iter=5000),
              {'C' : loguniform(2**-5, 2**15)},
              10))

# MODEL SVC RBF
models.append(('SVC RBF',
              SVC(kernel='rbf'),
              {'C' : loguniform(2**-5, 2**15),
               'gamma': loguniform(2**-9, 2**3)},
              10))

# MODEL Naive Bayes
models.append(('Naive Bayes',
              GaussianNB(),
              {},
              1))

# MODEL KNN
models.append(('KNN',
              KNeighborsClassifier(),
              {'n_neighbors': list(range(1, 302, 2))},
              10))

# MODEL MLP
models.append(('MLP',
              MLPClassifier(),
              {'hidden_layer_sizes': list(range(5, 21, 3))},
              6))

# MODEL DECISION TREE
```

```python
model = DecisionTreeClassifier()
models.append(('Decision Tree',
              DecisionTreeClassifier(),
              {'ccp_alpha': uniform(0.0, 0.04)},
              10))

# MODEL RANDOM FOREST
models.append(('Random Forest',
              RandomForestClassifier(),
              {'n_estimators': [10, 100, 1000],
               'max_features': [5, 8, 10]},
              9))

# MODEL GBM
models.append(('GBM',
              GradientBoostingClassifier(),
              {'n_estimators': list(range(5, 101)),
               'learning_rate': uniform(0.01, 0.3),
               'max_depth': [2, 3]},
              10))

results_df = compute_outer_score(models, X_scaled, y)

results_df
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/discriminant_analysis.py:691:
UserWarning: Variables are collinear
  warnings.warn("Variables are collinear")
/usr/local/lib/python3.7/dist-packages/sklearn/discriminant_analysis.py:691:
UserWarning: Variables are collinear
  warnings.warn("Variables are collinear")
/usr/local/lib/python3.7/dist-packages/sklearn/discriminant_analysis.py:691:
UserWarning: Variables are collinear
  warnings.warn("Variables are collinear")
/usr/local/lib/python3.7/dist-packages/sklearn/discriminant_analysis.py:691:
UserWarning: Variables are collinear
  warnings.warn("Variables are collinear")
```

[ ]:

	model	AUC
0	Logistic Regression	0.899217
1	Logistic Regression with L2 penalty	0.927371
2	LDA	0.928781
3	QDA	0.825241
4	SVC Linear	0.922557
5	SVC RBF	0.918747
6	Naive Bayes	0.838479
7	KNN	0.910143

8	MLP	0.911183
9	Decision Tree	0.903408
10	Random Forest	0.937497
11	GBM	0.937722