task

May 28, 2021

1 Tarefa 1: Aprendizado supervisionado - MO432

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

Prof. Jacques Wainer, 2021s1

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

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

1.1 Leitura da base de dados

[1]:
```
import pandas as pd
import requests
import io

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

db = pd.read_csv(io.StringIO(s.decode('utf-8')),
                 skiprows = 1,
                 header = None,
                 delim_whitespace = True)
db.head()
```

[1]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	H	A	X	1	3	1	1	1	1	1	0	0	0
1	D	R	O	1	3	1	1	2	1	1	0	0	0
2	C	S	O	1	3	1	1	2	1	1	0	0	0
3	H	R	X	1	2	1	1	1	1	1	0	0	0
4	H	S	X	1	1	1	1	2	1	1	0	0	0

1.1.1 Remover colunas de predição da base de dados

[2]:
```
output_3 = db.pop(db.columns[-1])
output_2 = db.pop(db.columns[-1])
output_1 = db.pop(db.columns[-1])
```

1.2 Conversão de dados categóricos para numéricos

[3]:
```
db_formatted = pd.get_dummies(db)

db_formatted.head()
```

[3]:

	3	4	5	6	7	8	9	0_B	0_C	…	1_H	1_K	1_R	1_S	1_X	2_C	2_I	2_O	2_X
0	1	3	1	1	1	1	1	0	0	…	0	0	0	0	0	0	0	0	1
1	1	3	1	1	2	1	1	0	0	…	0	0	1	0	0	0	0	1	0
2	1	3	1	1	2	1	1	0	1	…	0	0	0	1	0	0	0	1	0
3	1	2	1	1	1	1	1	0	0	…	0	0	1	0	0	0	0	0	1
4	1	1	1	1	2	1	1	0	0	…	0	0	0	1	0	0	0	0	1

[5 rows x 23 columns]

Ao realizar a conversão dos valores categóricos para numéricos, pode-se perceber que as três primeiras columas, foram transformadas em colunas adicionais à direita. O padrão de formação resultate é i_C, onde i é a coluna na qual a categoria C foi extraída.

1.3 Centering e Scaling dos dados

[4]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler(copy=True, with_mean=True, with_std=True)
db_preprocessed = scaler.fit_transform(db_formatted)

db_preprocessed
```

[4]:
```
array([[-0.42640143,  0.96486532, -0.18458572, …, -0.5143262 ,
        -0.89991511,  1.49014892],
       [-0.42640143,  0.96486532, -0.18458572, …, -0.5143262 ,
         1.11121593, -0.67107387],
       [-0.42640143,  0.96486532, -0.18458572, …, -0.5143262 ,
         1.11121593, -0.67107387],
       …,
       [-0.42640143, -0.64727642, -0.18458572, …, -0.5143262 ,
```

```
         1.11121593, -0.67107387],
       [-0.42640143, -0.64727642, -0.18458572, …, -0.5143262 ,
        -0.89991511,  1.49014892],
       [-0.42640143, -2.25941816, -0.18458572, …, -0.5143262 ,
         1.11121593, -0.67107387]])
```

1.4 PCA

1.4.1 Número de dimensões mantendo 90% da variância dos dados

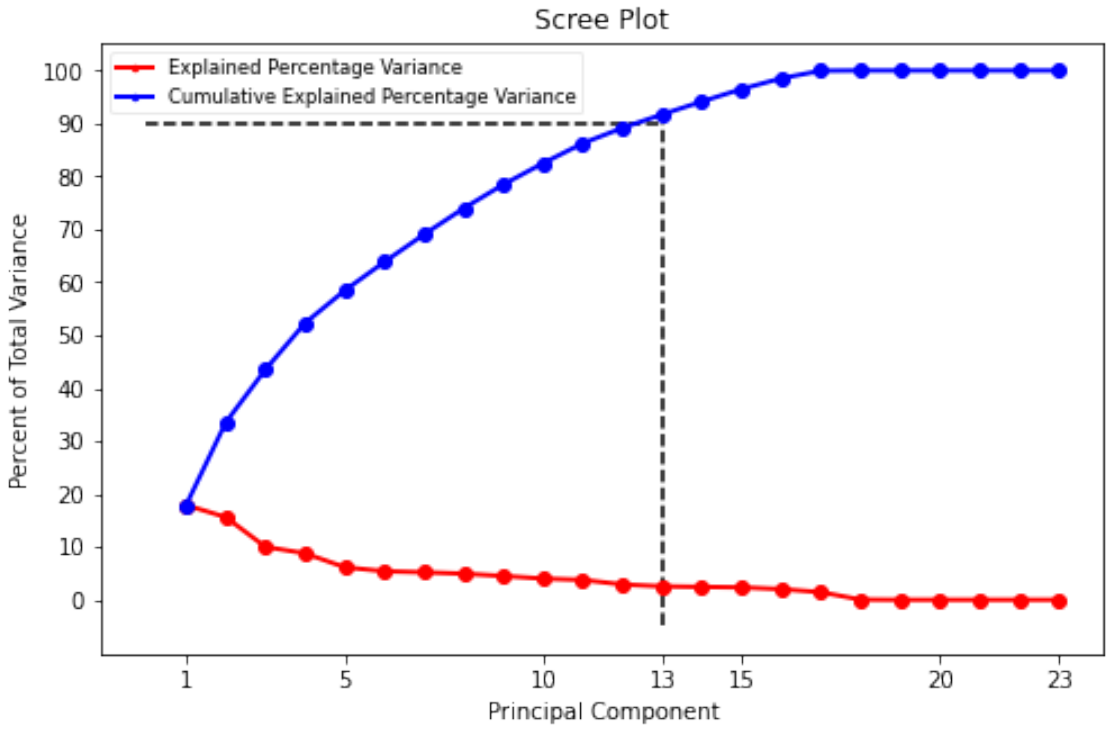
[12]:
```
from sklearn.decomposition import PCA

pca = PCA(0.9)
pca_fit = pca.fit_transform(db_preprocessed)
S_90 = pca_fit.shape[1]

print(f'Quantidade de dimensões mantendo 90% da variância: {S_90}')
```

```
Quantidade de dimensões mantendo 90% da variância: 13
```

1.4.2 Número de dimensões mantidas utilizando Scree Plot

[31]:
```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

num_vars = db_preprocessed.shape[1]
pca_full = PCA(num_vars)
pca_full.fit_transform(db_preprocessed)
eigvals = pca_full.explained_variance_ratio_ * 100

eigvals_sum = np.cumsum(eigvals)

fig = plt.figure(figsize=(8,5))
sing_vals = np.arange(num_vars) + 1
plt.plot(sing_vals, eigvals, 'ro-', linewidth=2)
plt.plot(sing_vals, eigvals_sum, 'bo-', linewidth=2)
plt.hlines(90, 0, 13, 'k', 'dashed')
plt.vlines(13, -5, 90, 'k', 'dashed')
plt.yticks(np.arange(0, 110, step=10))
plt.xticks([1, 5, 10, 13, 15, 20, 23])
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Percent of Total Variance')
leg = plt.legend(['Explained Percentage Variance', 'Cumulative Explained␣
↪Percentage Variance'], loc='best', borderpad=0.3,
                 shadow=False, prop=matplotlib.font_manager.
↪FontProperties(size='small'),
```

```
                  markerscale=0.4)
leg.get_frame().set_alpha(0.4)
plt.show()

# cumsum
```



1.4.3 Conversão dos dados utiizando PCA com 90% das variâncias

[32]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=13)
reduced_db_preprocessed = pca.fit_transform(db_preprocessed)

# Shape of reduced_db_preprocessed
print(f'Reduced Data shape: {reduced_db_preprocessed.shape}')
```

```
Reduced Data shape: (1066, 13)
```

1.5 Validação Cruzada e Regressão Linear

[65]:
```
from statistics import mean
from sklearn.model_selection import ShuffleSplit
from sklearn.linear_model import LinearRegression
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error

rs = ShuffleSplit(n_splits=5, test_size=.3, random_state=0)

mae_outputs = [[]] * 3
rmse_outputs = [[]] * 3

iterator = 0
for train_index, test_index in rs.split(reduced_db_preprocessed):
  # aux lists
  split_outputs   = []
  reg_outputs     = []
  outputs_predict = []

  # train and test splits
  split_train = reduced_db_preprocessed[train_index]
  split_test = reduced_db_preprocessed[test_index]

  # train and test 'label'
  split_outputs.append((output_1[train_index], output_1[test_index]))
  split_outputs.append((output_2[train_index], output_2[test_index]))
  split_outputs.append((output_3[train_index], output_3[test_index]))

  for i in range(0, 3):
    # Training
    reg_outputs.append(LinearRegression().fit(split_train, split_outputs[i][0]))

    # Testing
    outputs_predict.append(reg_outputs[i].predict(split_test))

    # MAE scores
    mae_outputs[0].append(
      mean_absolute_error(split_outputs[i][1], output1_predict))

    # RMSE scores
    rmse_outputs[0].append(
      np.sqrt(mean_squared_error(split_outputs[i][1], output1_predict)))

  print(f'({iterator+1}/5) MAE \t Output1: {mae_outputs[0][-1]:5f}'
        + f' Output2: {mae_outputs[1][-1]:5f} Output3: {mae_outputs[2][-1]:5f}')

  print(f'({iterator+1}/5) RMSE \t Output1: {rmse_outputs[0][-1]:5f}'
        + f' Output2: {rmse_outputs[1][-1]:5f} Output3: {rmse_outputs[2][-1]:
↪5f}')

  iterator += 1
```

```
print('=' * 70)
print(f'Mean MAE \t Output1: {mean(mae_outputs[0]):5f}'
      + f' Output2: {mean(mae_outputs[1]):5f} Output3: {mean(mae_outputs[2]):
 ↪5f}')
print(f'Mean RMSE \t Output1: {mean(rmse_outputs[0]):5f}'
      + f' Output2: {mean(rmse_outputs[1]):5f} Output3: {mean(rmse_outputs[2]):
 ↪5f}')
```

```
(1/5) MAE        Output1: 0.349921 Output2: 0.349921 Output3: 0.349921
(1/5) RMSE       Output1: 0.372058 Output2: 0.372058 Output3: 0.372058
(2/5) MAE        Output1: 0.348277 Output2: 0.348277 Output3: 0.348277
(2/5) RMSE       Output1: 0.369843 Output2: 0.369843 Output3: 0.369843
(3/5) MAE        Output1: 0.353466 Output2: 0.353466 Output3: 0.353466
(3/5) RMSE       Output1: 0.381818 Output2: 0.381818 Output3: 0.381818
(4/5) MAE        Output1: 0.348572 Output2: 0.348572 Output3: 0.348572
(4/5) RMSE       Output1: 0.370241 Output2: 0.370241 Output3: 0.370241
(5/5) MAE        Output1: 0.354527 Output2: 0.354527 Output3: 0.354527
(5/5) RMSE       Output1: 0.378465 Output2: 0.378465 Output3: 0.378465
======================================================================
Mean MAE         Output1: 0.416672 Output2: 0.416672 Output3: 0.416672
Mean RMSE        Output1: 0.553123 Output2: 0.553123 Output3: 0.553123
```