UNIVERSIDADE FEDERAL DE ITAJUBÁ
ENGENHARIA DE COMPUTAÇÃO

GABRIEL TAUCHEN FILGUEIRAS - 2022003880
MATEUS ALEXANDRE MARTINS DE SOUZA - 2021004023

INTELIGÊNCIA ARTIFICIAL
REGRESSÃO E CLASSIFICAÇÃO

ITAJUBÁ
2023

notebooks-intro

November 15, 2023

1 Regressão - Trabaho IA - 2023.2

Para a classificação com algoritmo de regressão linear, foi utilizada a seguinte base de dados: Experience Salary Dataset. Segue o código com os comentários nas células anteriores: Importando as bibliotecas e funções.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

2 Base de dados

[11]:
```
salary = pd.read_csv('ExperienceSalary.csv')
salary.head()
```

[11]:

	exp(in months)	salary(in thousands)
0	18.290293	16.521825
1	17.023407	11.666234
2	26.343613	23.167255
3	19.105834	20.877145
4	27.742516	23.166236

2.1 Pré-processamento dos dados

[12]:
```
salary.isnull().sum()
```

[12]:
```
exp(in months)          0
salary(in thousands)    0
dtype: int64
```

qual linha?

[13]:
```
salary[salary.isna().any(axis=1)]
```

[13]:
```
Empty DataFrame
Columns: [exp(in months), salary(in thousands)]
Index: []
```

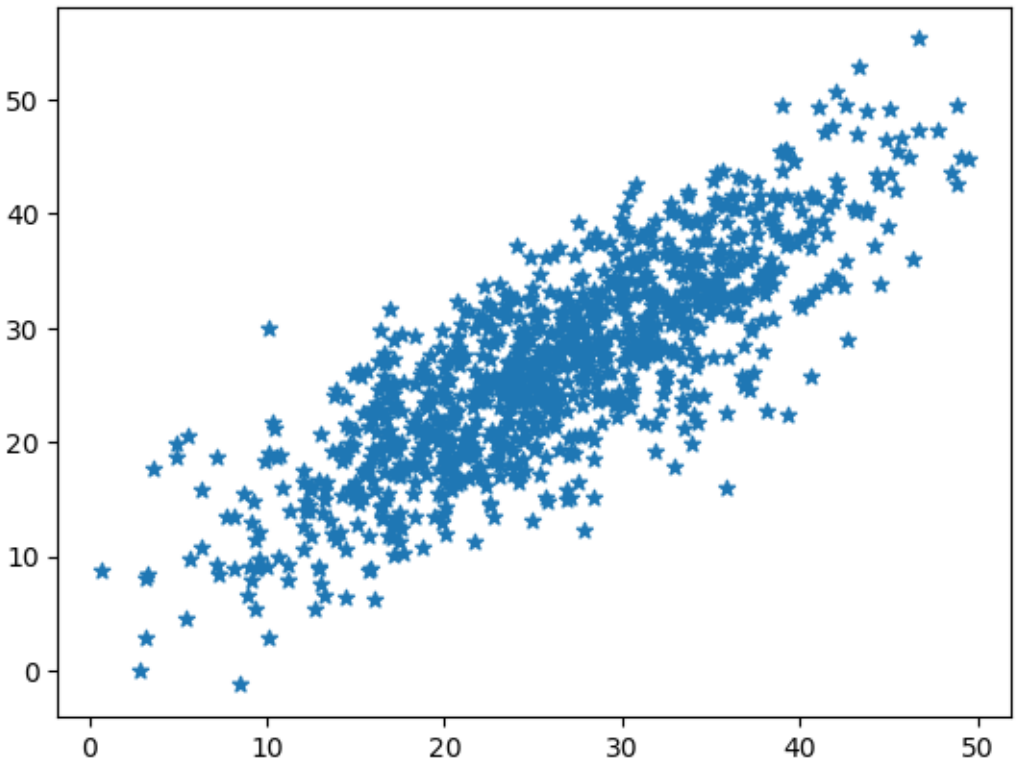
Remover linhas com dados faltantes

[14]:
```
salary = salary.dropna()
```

3 Visualizando os dados

[15]:
```
plt.scatter(salary['exp(in months)'], salary['salary(in thousands)'],␣
↪marker='*')
```

[15]:
```
<matplotlib.collections.PathCollection at 0xa3b2d78>
```



4 Dividindo em treino/teste

[16]:
```
from sklearn.model_selection import train_test_split
x_val = salary['exp(in months)'].values.reshape(-1,1)
y_val = salary['salary(in thousands)']
x_train, x_test, y_train, y_test = train_test_split(x_val, y_val, test_size=0.
↪3, random_state=42)
```

5 Fazendo a regressão

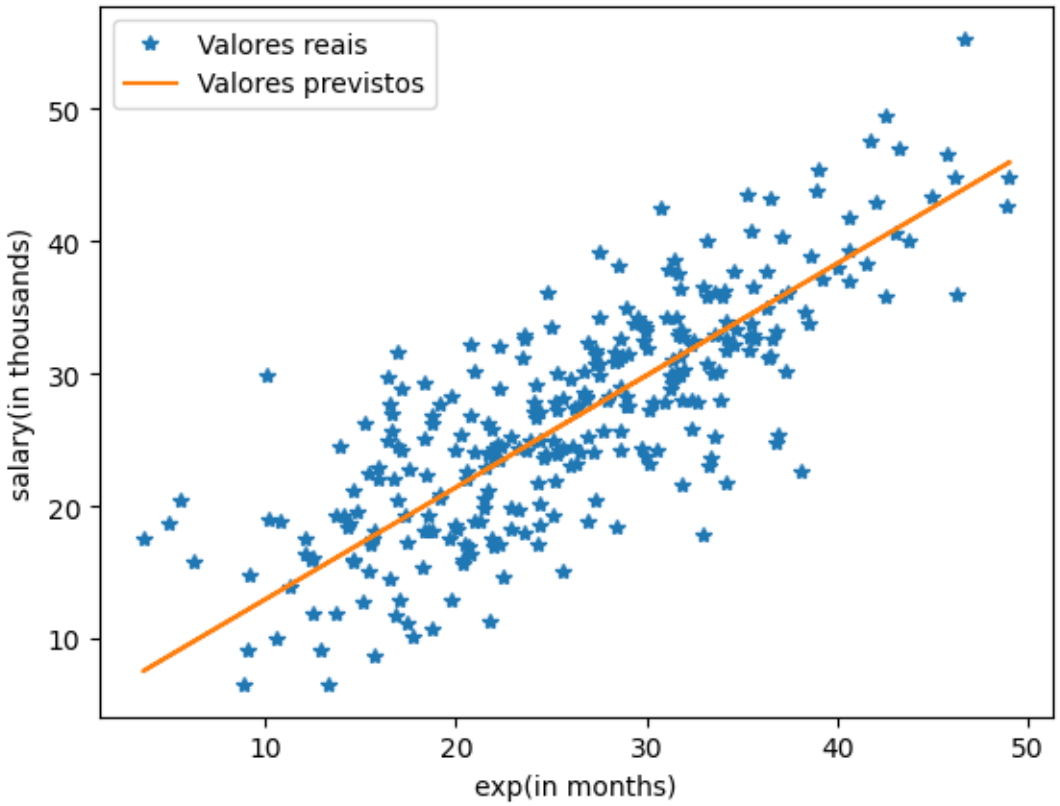
[17]:
```
from sklearn.linear_model import LinearRegression
# regressão (treinamento)
my_model = LinearRegression()
my_model.fit(x_train, y_train)
print("R_squared =", my_model.score(x_test, y_test)) # r2 squared
#regressão (teste)
test_pred = my_model.predict(x_test)
```

```
R_squared = 0.624443043862203
```

6 Analisando os resultados

Plotando o gráfico da regressão para dados de teste:

[18]:
```
plt.plot(x_test, y_test, '*', label='Valores reais')
plt.plot(x_test, test_pred, '-', label='Valores previstos')
plt.xlabel('exp(in months)')
plt.ylabel('salary(in thousands)')
plt.legend()
plt.show()
```



6.1 Qual equação foi gerada durante a regressão?

[19]:
```
print(my_model.intercept_, my_model.coef_, my_model.score(x_test, y_test))
```

```
4.522271854340648 [0.84519042] 0.624443043862203
```

A equação que foi obtida pela regreção é aprocimadamente:

6.1.1 y = 4.522271854340648 + 0.84519042 * x

O coeficiente de determinação (R^2) do modelo é 62.44, o que significa que 62.44% da variação de salary pode ser explicada variação de exp.

Classificação - Trabalho IA - 2023.2

Para a classificação com algoritmo de regressão logística, foi utilizada a seguinte base de dados: Spam Email Classification

Segue o código com os comentários nas células anteriores:

Importando as bibliotecas e funções.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
```

Base de dados de spam

Esse dataset tem informações sobre e-mails e seu objetivo é identificar se a mensagem é ou não um spam.

```
spam = pd.read_csv('data/spambase.csv')
spam.head()
```

```
   word_freq_make  word_freq_address  word_freq_all  word_freq_3d  \
0            0.00               0.64           0.64           0.0
1            0.21               0.28           0.50           0.0
2            0.06               0.00           0.71           0.0
3            0.00               0.00           0.00           0.0
4            0.00               0.00           0.00           0.0

   word_freq_our  word_freq_over  word_freq_remove  word_freq_internet
\
0           0.32            0.00              0.00                0.00

1           0.14            0.28              0.21                0.07

2           1.23            0.19              0.19                0.12

3           0.63            0.00              0.31                0.63

4           0.63            0.00              0.31                0.63


   word_freq_order  word_freq_mail  ...  char_freq_%3B  char_freq_%28
\
0             0.00            0.00  ...           0.00          0.000
```

```
1              0.00              0.94  ...              0.00           0.132

2              0.64              0.25  ...              0.01           0.143

3              0.31              0.63  ...              0.00           0.137

4              0.31              0.63  ...              0.00           0.135


   char_freq_%5B  char_freq_%21  char_freq_%24  char_freq_%23  \
0            0.0          0.778          0.000          0.000
1            0.0          0.372          0.180          0.048
2            0.0          0.276          0.184          0.010
3            0.0          0.137          0.000          0.000
4            0.0          0.135          0.000          0.000

   capital_run_length_average  capital_run_length_longest  \
0                       3.756                          61
1                       5.114                         101
2                       9.821                         485
3                       3.537                          40
4                       3.537                          40

   capital_run_length_total  class
0                       278      1
1                      1028      1
2                      2259      1
3                       191      1
4                       191      1

[5 rows x 58 columns]
```

O conjunto de dados é baseado, principalmente na porcentagem da composição do e-mail por uma única palavra, de caracteres e no número de letras maíusculas consecutivas.

```
spam.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4601 entries, 0 to 4600
Data columns (total 58 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   word_freq_make              4601 non-null   float64
 1   word_freq_address           4601 non-null   float64
 2   word_freq_all               4601 non-null   float64
 3   word_freq_3d                4601 non-null   float64
 4   word_freq_our               4601 non-null   float64
 5   word_freq_over              4601 non-null   float64
 6   word_freq_remove            4601 non-null   float64
 7   word_freq_internet          4601 non-null   float64
```

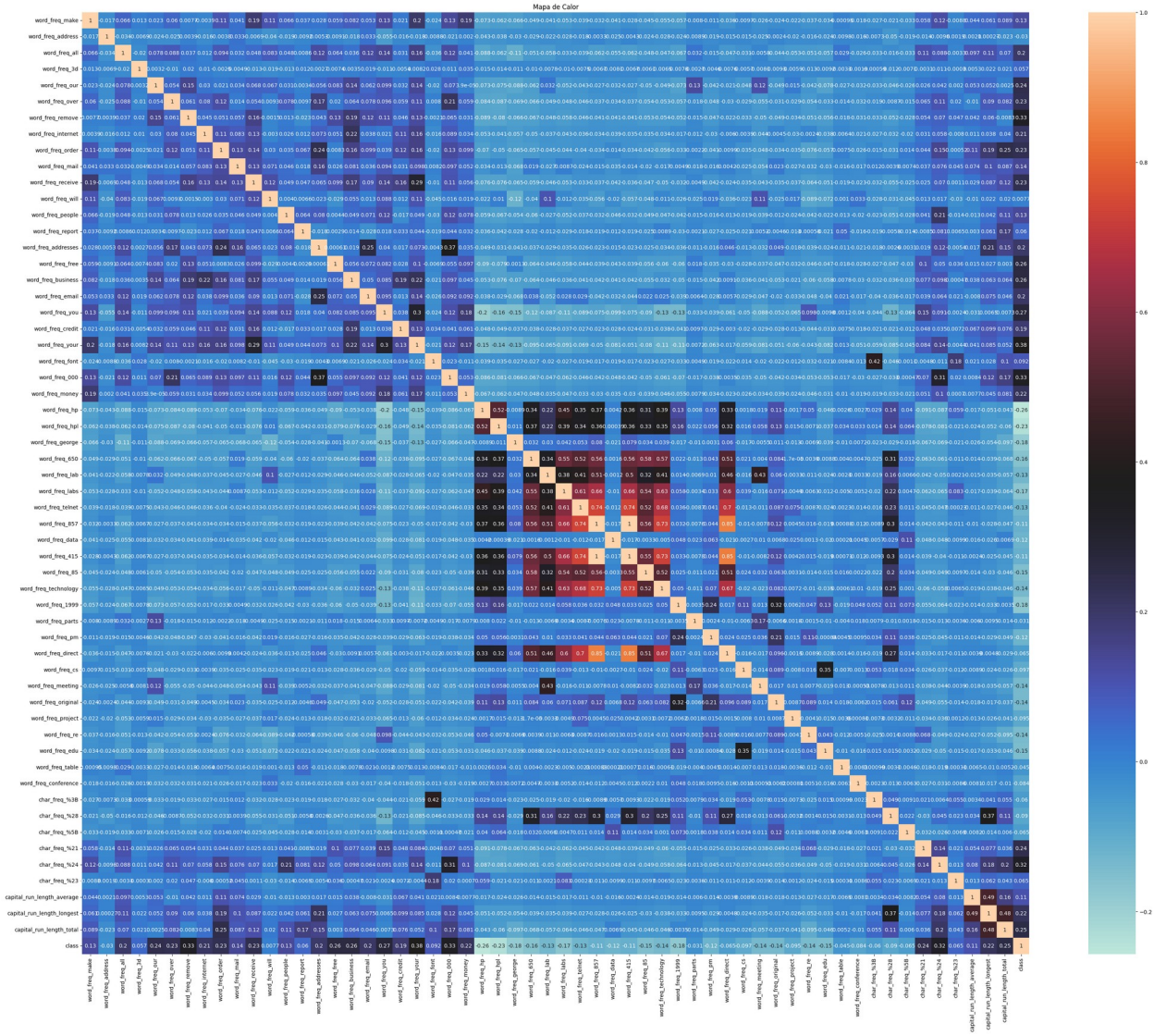
```
8   word_freq_order             4601 non-null   float64
9   word_freq_mail              4601 non-null   float64
10  word_freq_receive           4601 non-null   float64
11  word_freq_will              4601 non-null   float64
12  word_freq_people            4601 non-null   float64
13  word_freq_report            4601 non-null   float64
14  word_freq_addresses         4601 non-null   float64
15  word_freq_free              4601 non-null   float64
16  word_freq_business          4601 non-null   float64
17  word_freq_email             4601 non-null   float64
18  word_freq_you               4601 non-null   float64
19  word_freq_credit            4601 non-null   float64
20  word_freq_your              4601 non-null   float64
21  word_freq_font              4601 non-null   float64
22  word_freq_000               4601 non-null   float64
23  word_freq_money             4601 non-null   float64
24  word_freq_hp                4601 non-null   float64
25  word_freq_hpl               4601 non-null   float64
26  word_freq_george            4601 non-null   float64
27  word_freq_650               4601 non-null   float64
28  word_freq_lab               4601 non-null   float64
29  word_freq_labs              4601 non-null   float64
30  word_freq_telnet            4601 non-null   float64
31  word_freq_857               4601 non-null   float64
32  word_freq_data              4601 non-null   float64
33  word_freq_415               4601 non-null   float64
34  word_freq_85                4601 non-null   float64
35  word_freq_technology        4601 non-null   float64
36  word_freq_1999              4601 non-null   float64
37  word_freq_parts             4601 non-null   float64
38  word_freq_pm                4601 non-null   float64
39  word_freq_direct            4601 non-null   float64
40  word_freq_cs                4601 non-null   float64
41  word_freq_meeting           4601 non-null   float64
42  word_freq_original          4601 non-null   float64
43  word_freq_project           4601 non-null   float64
44  word_freq_re                4601 non-null   float64
45  word_freq_edu               4601 non-null   float64
46  word_freq_table             4601 non-null   float64
47  word_freq_conference        4601 non-null   float64
48  char_freq_%3B               4601 non-null   float64
49  char_freq_%28               4601 non-null   float64
50  char_freq_%5B               4601 non-null   float64
51  char_freq_%21               4601 non-null   float64
52  char_freq_%24               4601 non-null   float64
53  char_freq_%23               4601 non-null   float64
54  capital_run_length_average  4601 non-null   float64
55  capital_run_length_longest  4601 non-null   int64
56  capital_run_length_total    4601 non-null   int64
```

```
 57  class                          4601 non-null   int64
dtypes: float64(55), int64(3)
memory usage: 2.0 MB
```

Heatmap

O mapa térmico (heatmap) de correlação é utilizado para verificar a força das relações entre as variáveis no conjunto de dados. Entretando, pela grande quantidade de colunas, o gráfico só pode ser bem visualizado com zoom ou ao ser limitado.

```
plt.figure(figsize=(40, 32))
sns.heatmap(spam.corr(), annot=True, cmap='icefire').set_title('Mapa de Calor')
plt.show()
```



Por isso, pode-se observar somente a última coluna, referente ao resultado e extrair seus cinco maiores valores, excluindo a correlação com a própria coluna.

```
correlacoes = spam.corr().loc[:, 'class'].drop('class')
maiores_correlacoes = correlacoes.abs().nlargest(10)
print(f"Os 5 maiores valores de correlação com o resultado:")
print(maiores_correlacoes)
```

```
Os 5 maiores valores de correlação com o resultado:
word_freq_your              0.383234
word_freq_000               0.334787
word_freq_remove            0.332117
char_freq_%24               0.323629
word_freq_you               0.273651
word_freq_free              0.263215
word_freq_business          0.263204
word_freq_hp                0.256723
capital_run_length_total    0.249164
word_freq_our               0.241920
Name: class, dtype: float64
```

Assim, foram escolhidas as colunas referntes às palavras "your" e "free", a primeira e sexta na lista, respectivamente.

```
x = spam[['word_freq_free', 'word_freq_your']]
y = spam['class']
```

Regressão Logística

É realizada chamada da função de Regressão Logística. O desempenho atende ao requisito de, no mínimo, 70% de acerto.

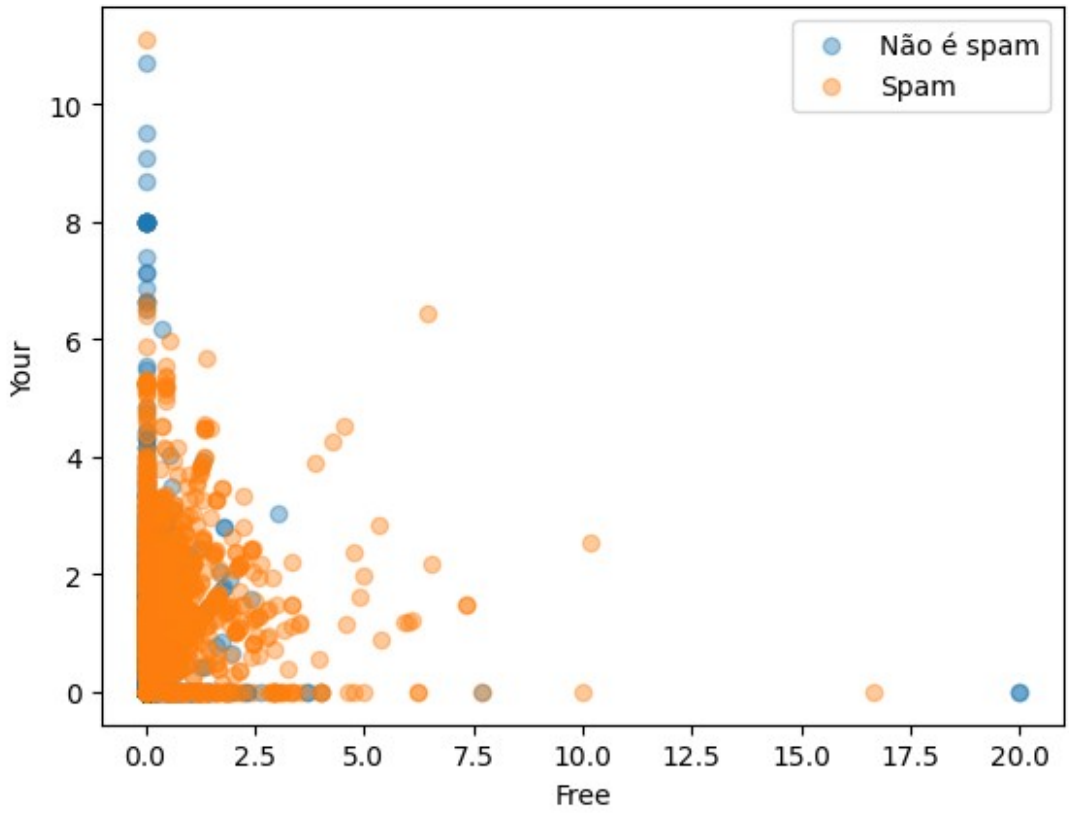
```
lr = LogisticRegression()
scores = cross_val_score(lr, x, y, cv=10, scoring='accuracy') #
validação cruzada
print('Acurácia média:', scores.mean())
```

```
Acurácia média: 0.7630953503725361
```

Aqui, é mostrado o gráfico da regressão.

```
def plot_scatter():
    plt.scatter(x['word_freq_free'][y == 0], x['word_freq_your'][y ==
0], alpha=.4, label='Não é spam')
    plt.scatter(x['word_freq_free'][y == 1], x['word_freq_your'][y ==
1], alpha=.4, label='Spam')
    plt.legend()
    plt.xlabel('Free')
```

```
    plt.ylabel('Your')
plot_scatter()
```



Teste

Ao testar o modelo com e-mails aleatórios, ele prediz que o primeiro, cuja frequência da palavra "free" é 0.85 e a frequência de "your"´é 0.15, é um spam. Já o segundo, cujas frequências são 0.2 e 0.6, respectivamente, não foi considerado spam. A probabilidade do primeiro ser spam foi de 51,7% e do segundo de 35,76%.

```
df_emails = pd.DataFrame([[0.85, 0.15], # email 1
[0.2, 0.6]]) # email 2
df_emails.columns = ["word_freq_free", "word_freq_your"]
lr.fit(x, y)
print('Classificações:\n', lr.predict(df_emails))
print('Probabilidades:\n', lr.predict_proba(df_emails))
```

```
Classificações:
 [1 0]
Probabilidades:
 [[0.48294689 0.51705311]
 [0.64236466 0.35763534]]
```