Clase7 IMA539

Alejandro Ferreira Vergara

April 10, 2023

1 Preprocesamiento de Datos

1.1 ¿Qué aprenderemos hoy?

- **Tratar los valores perdidos en un conjunto de datos**
- **Tratar con diferentes tipos de variables (contínuas y discretas)**
- **Estudiar subgrupos de variables incluidas en conjuntos de datos**
- **Librería Pandas**

2 Tratar con Valores Perdidos (datos)

- **Espacio en blanco o información inconsistente**
- **NaN (no hay número)**
- **Null (valores desconocidos)**

2.1 Datos Continuos

[ ]:
```
import pandas as pd

df = pd.read_csv("clase7/Clase7 CSV.csv", header=0)
df
```

[ ]:
```
df.dtypes
```

[ ]:
```
df.isnull().sum()
```

[ ]:
```
df.dropna(axis=0)
```

[ ]:
```
df.dropna(axis=1)
```

[ ]:
```
df.dropna(thresh=4)
```

[ ]:
```
df.dropna(subset=['D'])
```

[ ]:
```
df.values
```

¿Qué hacer con los valores perdido?

¿Simplemente los eliminamos?

[ ]:
```
import numpy as np
from sklearn.impute import SimpleImputer

imr = SimpleImputer(missing_values=np.nan, strategy='mean')
imr = imr.fit(df.values)
imputed_data = imr.transform(df.values)
imputed_data
```

[ ]:
```
df[['A','B','C','D']] = imputed_data
df
```

¿Cómo obtenemos la media de cada columna?

[ ]:
```

```

2.2 Procesando Datos Categóricos

- **Ordinales:** Valores categóricos que pueden ordenarse.
- **Nominales:** Valores categóricos que no pueden ordenarse.

[ ]:
```
df = pd.DataFrame([['green', 'M', 10.1, 'class2'],
                   ['red', 'L', 13.5, 'class1'],
                   ['blue', 'XL', 15.3, 'class2']])

df.columns = ['color', 'size', 'price', 'class_label']
df
```

Los algoritmos que conocemos ¿podríamos ajustarlos directamente a partir del Dataframe anterior?

[ ]:
```
size_mapping = {'XL': 3,'L': 2,'M': 1}

df['size'] = df['size'].map(size_mapping)
df
```

[ ]:
```
class_mapping = {label: idx for idx, label in enumerate(np.
 ↪unique(df['class_label']))}
class_mapping
```

[ ]:
```
df['class_label'] = df['class_label'].map(class_mapping)
df
```

[ ]:
```
y = df.iloc[:,3].values
y
```

[ ]:
```
X = df[['color', 'size', 'price']].values
X
```

Los algoritmos que conocemos ¿podríamos ajustarlos directamente a partir de X e y?

[ ]:
```
X = pd.get_dummies(df[['price', 'size', 'color']]).values
X
```

[ ]:
```
print('Conjunto de Variables o Características:\n',X)
print('Conjunto de Etiquetas de Clase:\n',y)
```

3 Trabajando con una Base de Datos real

[ ]:
```
df_wine = pd.read_csv('https://archive.ics.uci.edu/ml/
 ↪machine-learning-databases/wine/wine.data', header=None)

df_wine.columns = ['Class label', 'Alcohol', 'Malic acid', 'Ash',
                   'Alcalinity of ash', 'Magnesium', 'Total phenols',
                   'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyanins',
                   'Color intensity', 'Hue', 'OD280/OD315 of diluted wines',
                   'Proline']

df_wine
```

[ ]:
```
df_wine.describe()
```

[ ]:
```
print('Etiquetas de clase', np.unique(df_wine['Class label']))
```

[ ]:
```
df_wine.isnull().sum()
```

[ ]:
```
df_wine.describe(include='all')
```

[ ]:
```
df_wine.iloc[:,0] = df_wine.iloc[:,0].astype('category')
```

[ ]:
```
df_wine.describe(include='all')
```

[ ]:
```
from sklearn.model_selection import train_test_split

X, y = df_wine.iloc[:, 1:].values, df_wine.iloc[:, 0].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=0, stratify=y)
```

3.1 Reescalamiento de las variables

- **Normalización:**

$$x_{norm}^{(i)} = \frac{x^{(i)} - x_{min}}{x_{max} - x_{min}}$$

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

mms = MinMaxScaler()

X_train_norm = mms.fit_transform(X_train)
X_test_norm = mms.transform(X_test)

print(X_train_norm.min())
print(X_train_norm.max())
print(X_test_norm.min())
print(X_test_norm.max())
```

- **Estandarización:**

$$x_{std}^{(i)} = \frac{x^{(i)} - \mu_x}{\sigma_x}$$

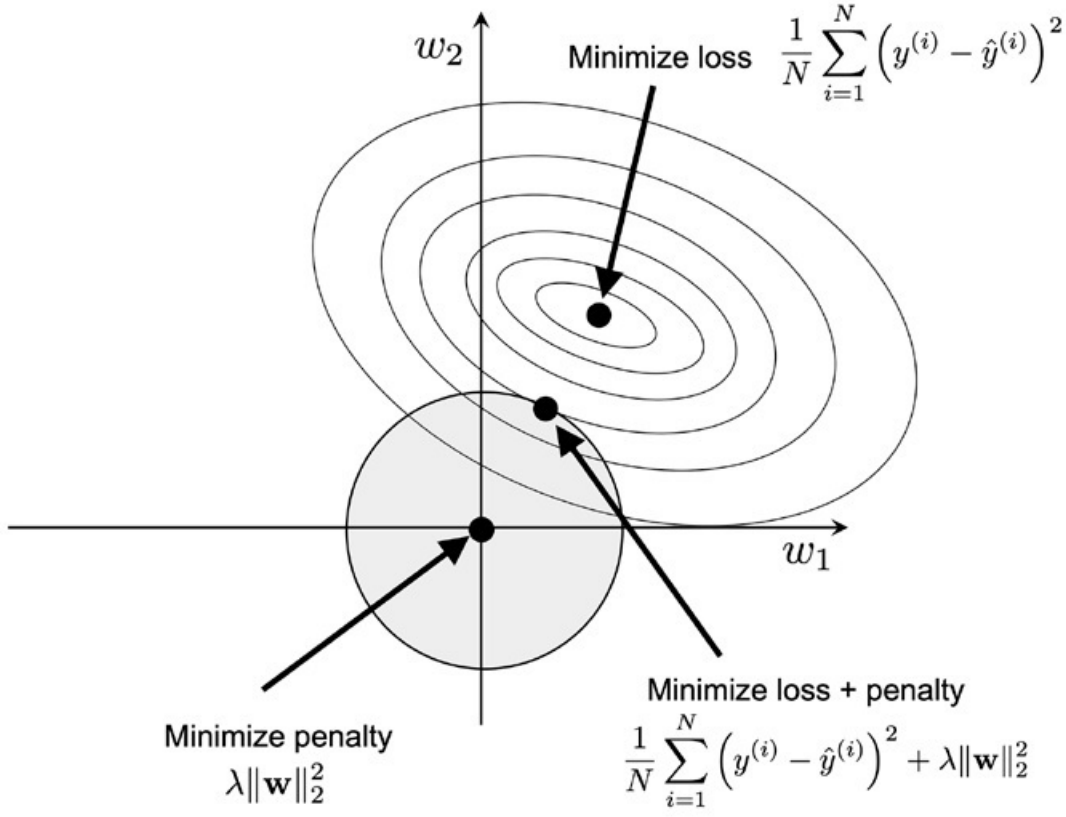
[ ]:
```
from sklearn.preprocessing import StandardScaler

stdsc = StandardScaler()

X_train_std = stdsc.fit_transform(X_train)
X_test_std = stdsc.transform(X_test)

print(X_train_std.min())
print(X_train_std.max())
print(X_test_std.min())
print(X_test_std.max())
```

3.2 Estudiando las variables para reducir la complejidad del Modelo

- **Regularización L2:** $\|w\|_2^2 = \sum_{i=1}^{m} w_i^2$

[1]:
```
from IPython.display import Image

Image(filename=r'clase7/7_1.jpg', width=540)
```

[1]:



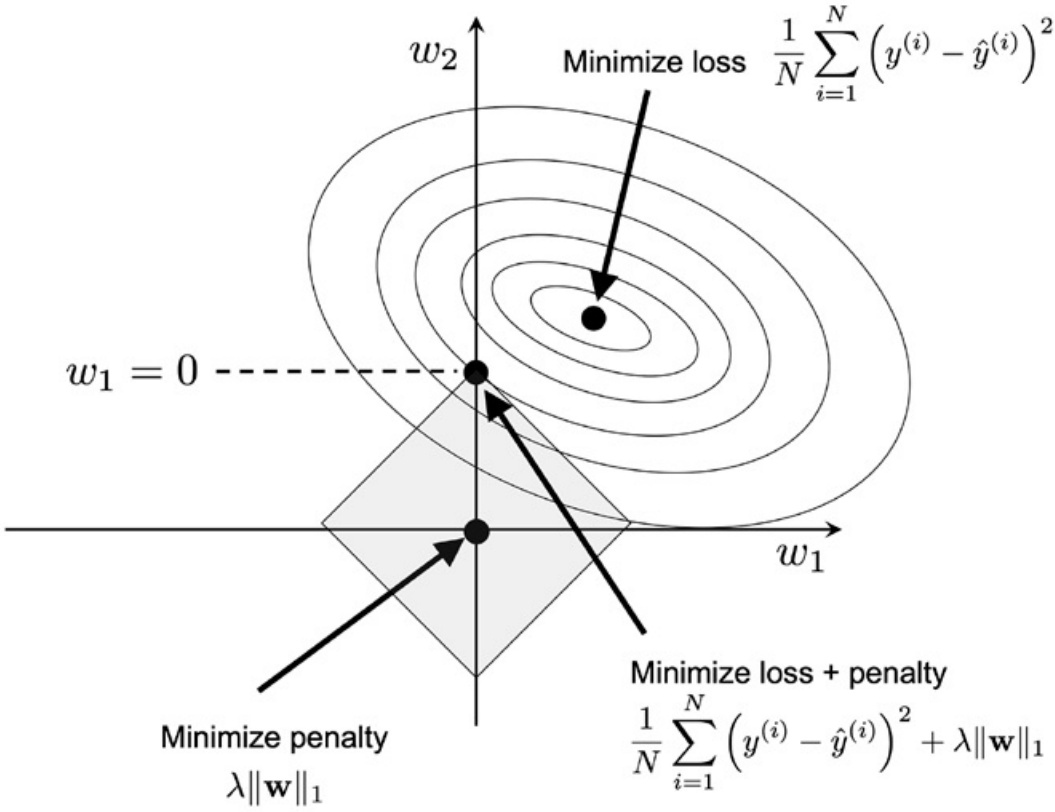
[ ]:
```
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt

fig = plt.figure()
ax = plt.subplot(111)

colors = ['blue', 'green', 'red', 'cyan', 'magenta', 'yellow', 'black',
          'pink', 'lightgreen', 'lightblue', 'gray', 'indigo', 'orange']

weights, params = [], []
for c in np.arange(-4., 6.):
    lr = LogisticRegression(penalty='l2', C=10.**c, random_state=0)
    lr.fit(X_train_std, y_train)
    weights.append(lr.coef_[1])
    params.append(10**c)

weights = np.array(weights)

for column, color in zip(range(weights.shape[1]), colors):
    plt.plot(params,weights[:, column],label=df_wine.columns[column +␣
 ↪1],color=color)
```

```
plt.axhline(0, color='black', linestyle='--', linewidth=3)
plt.xlim([10**(-5), 10**5])
plt.ylabel('weight coefficient')
plt.xlabel('C')
plt.xscale('log')
plt.legend(loc='upper left')
ax.legend(loc='upper center',bbox_to_anchor=(1.38,1.03),ncol=1,fancybox=True)
#plt.savefig('images/04_07.png', dpi=300,
#            bbox_inches='tight', pad_inches=0.2)
plt.show()
```

- **Regularización L1:** $\|w\|_1 = \sum_{i=1}^{m} |w_i|$

[2]:
```
Image(filename=r'clase7/7_2.jpg', width=540)
```

[2]:



[ ]:
```
fig = plt.figure()
ax = plt.subplot(111)

colors = ['blue', 'green', 'red', 'cyan', 'magenta', 'yellow', 'black',
          'pink', 'lightgreen', 'lightblue', 'gray', 'indigo', 'orange']

weights, params = [], []
```

```
for c in np.arange(-4., 6.):
    lr = LogisticRegression(penalty='l1', C=10.**c, random_state=0,␣
 ↪solver='saga')
    lr.fit(X_train_std, y_train)
    weights.append(lr.coef_[1])
    params.append(10**c)

weights = np.array(weights)

for column, color in zip(range(weights.shape[1]), colors):
    plt.plot(params,weights[:, column],label=df_wine.columns[column +␣
 ↪1],color=color)
plt.axhline(0, color='black', linestyle='--', linewidth=3)
plt.xlim([10**(-5), 10**5])
plt.ylabel('weight coefficient')
plt.xlabel('C')
plt.xscale('log')
plt.legend(loc='upper left')
ax.legend(loc='upper center',bbox_to_anchor=(1.38,1.03),ncol=1,fancybox=True)
#plt.savefig('images/04_07.png', dpi=300,
#            bbox_inches='tight', pad_inches=0.2)
plt.show()
```

4 Actividad Final

Desde el siguiente enlace descargar la BBDD "Asignaciones y Créditos 2022" para realizar las siguientes actividades:

Enlace: https://datosabiertos.mineduc.cl/asignaciones-de-becas-y-creditos-en-educacion-superior/

Cargar la BBDD:

[ ]:

Pregunta 1 ¿Cuántas columnas y registros tiene la BBDD?

[ ]:

Pregunta 2 ¿Cuántos tipos de Beneficios diferentes existen?

[ ]:

Pregunta 3 ¿Cuántos tipos de alumnos existen en la BBDD?

[ ]:

Pregunta 4 ¿Cuántos estudiantes tuvieron beneficios el año 2022?

[ ]:

Pregunta 5 Realizar un mapeo de lo tipos de beneficios y transformar la columna a valores discretos.

[ ]: