Tarea 2

Topicos Aplicados en Estadistica

Nicolás Morales Nicolás Bustos

Pre-procesamiento de los datos

Para este trabajo lo primero es importar los datos:

```
Hitters = pd.read_csv("Hitters.csv")
Hitters = Hitters.rename(columns={"Unnamed: 0" : "Name"})
Hitters['Name'] = Hitters['Name'].apply(lambda x: x[1:])
```

A continuación se muestra una parte de las tres primeras filas de la tabla recién creada:

```
     Name      AtBat  Hits  HmRun  ...  Assists  Errors  Salary  NewLeague
Andy Allanson  293     66     1    ...    33       20      NaN      A
   Alan Ashby  315     81     7    ...    43       10    475.0      N
  Alvin Davis  479    130    18    ...    82       14    480.0      A
```

Acto seguido se realiza un conteo de valores faltantes (`NaN`). En la siguiente salida se aprecia que solo la columna *Salary* contiene estos eventos, con un total de 59 ocurrencias.

```
suma_nan = Hitters.isnull().sum(axis = 0)
print(suma_nan[suma_nan != 0])
```

```
Salary    59
dtype: int64
```

Se realiza un filtro, redefiniendo la base de datos dejando sólo aquellas filas libres de valores faltantes.

```
Hitters_NAN = Hitters[Hitters["Salary"].isnull()]
Hitters     = Hitters[Hitters["Salary"].isnull() == False]
```

Puesto que la distribución del salario es sesgada, se pide considerar en su lugar el logaritmo del salario:

```
Hitters['Salary'] = Hitters['Salary'].apply(lambda x: math.log(x))
```

Se verifica que las variables numéricas efectivamente tomen valores positivos de acuerdo a su construcción:

```
aux = Hitters[Hitters.columns.difference(['Name','League','Division','NewLeague'])] < 0
aux.sum(axis = 0)
```

	0
Assists	0
AtBat	0
CAtBat	0
CHits	0
CHmRun	0
CRBI	0
CRuns	0
CWalks	0
Errors	0
Hits	0
HmRun	0
PutOuts	0
RBI	0
Runs	0
Salary	0
Walks	0
Years	0

Para las variables numéricas, se examinan los posibles *outliers*:

	0
Assists	<- 5 outliers.
AtBat	
CAtBat	<- 4 outliers.
CHits	<- 5 outliers.
CHmRun	<- 23 outliers.
CRBI	<- 22 outliers.
CRuns	<- 8 outliers.
CWalks	<- 20 outliers.
Errors	<- 2 outliers.
Hits	
HmRun	<- 1 outliers.
PutOuts	<- 29 outliers.
RBI	
Runs	
Salary	
Walks	
Years	<- 3 outliers.

Escalar las variables para evitar problemas numéricos al ajustar la regresión:

```
def mi_scale(serie):
  mean = sum(serie) / len(serie)
  standard_deviation = math.sqrt( sum( (serie - mean)**2 ) / len(serie))
  return((serie - mean) / standard_deviation)

Hitters[Hitters.columns.difference(['Name','League', 'Division', 'NewLeague'])] = Hitters[
```

Para el manejo de variables categóricas, se generan variables *dummy* definidas como indicadoras 0-1:

```
Hitters = pd.get_dummies(Hitters, columns = ['League', 'Division', 'NewLeague'], drop_firs
```

¿Cuáles son las características más importantes para predecir el salario de los jugadores?

Primer modelo: Regresión lasso

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
y=Hitters["Salary"]
X=Hitters.drop(["Salary","Name"], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.20,
                                                   random_state=46)
from sklearn.linear_model import Ridge, Lasso
from sklearn.linear_model import RidgeCV, LassoCV

lasso_model = Lasso().fit(X_train, y_train)
y_pred=lasso_model.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))
```

```
1.0851027333102952
```

```
alphas = 10**np.linspace(10,-2,100)*0.5
lasso_cv_model = LassoCV(alphas = alphas, cv = 10).fit(X_train, y_train)
print(lasso_cv_model.alpha_)
```

```
0.03527401155359316
```

```
lasso_tuned = Lasso(alpha = lasso_cv_model.alpha_).fit(X_train, y_train)
y_pred = lasso_tuned.predict(X_test)
np.sqrt(mean_squared_error(y_pred,y_test))
```

```
0.9612798421259551
```

```
coeficientes = pd.DataFrame({'Names':X_train.columns.values.tolist(),'Coef':lasso_tuned.co
coeficientes[coeficientes["Coef"] != 0]
```

```
D:\Programas\Python\lib\site-packages\IPython\core\formatters.py:343: FutureWarning:

In future versions `DataFrame.to_latex` is expected to utilise the base implementation of `St
```

	Names	Coef
1	Hits	0.195024
2	HmRun	0.048231
3	Runs	0.131532
5	Walks	0.023773
6	Years	0.172494
8	CHits	0.320762
10	CRuns	0.048085
13	PutOuts	0.083241

Segundo modelo: Elastic Net

```
from sklearn.linear_model import Ridge, Lasso, ElasticNet
from sklearn.linear_model import RidgeCV, LassoCV,ElasticNetCV
```

```
enet_model = ElasticNet().fit(X_train, y_train)
y_pred = enet_model.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))
```

```
1.0278495783220163
```

```
alphas = 10**np.linspace(10,-2,100)*0.5
enet_cv_model = ElasticNetCV(alphas = alphas, cv = 10).fit(X_train, y_train)
enet_cv_model.alpha_
```

```
0.06164233697210317
```

```
enet_tuned = ElasticNet(alpha = enet_cv_model.alpha_).fit(X_train, y_train)
y_pred = enet_tuned.predict(X_test)
np.sqrt(mean_squared_error(y_test, y_pred))
```

```
0.9605580787178876
```

```
coeficientes = pd.DataFrame({'Names':X_train.columns.values.tolist(),'Coef':enet_tuned.coe
coeficientes[coeficientes["Coef"] != 0]
```

```
D:\Programas\Python\lib\site-packages\IPython\core\formatters.py:343: FutureWarning:

In future versions `DataFrame.to_latex` is expected to utilise the base implementation of `St
```

	Names	Coef
1	Hits	0.203052
2	HmRun	0.050585
3	Runs	0.119101
5	Walks	0.024919
6	Years	0.172095
7	CAtBat	0.051878
8	CHits	0.194369
10	CRuns	0.121675
13	PutOuts	0.086176