Data Translator

Caso de Estudio Regresión

Equipo 3

José Jiménez Gonzalez

Iván López García

Data Translator - Resultados

Importamos las librerías a utilizar en jupyter y leemos nuestro archivo *“Estudycas/Usvideos.csv”.*

In [1]:

```
import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns

from pandas.plotting import scatter_matrix
import plotly.express as px
#from sklearn.preprocessing import PolynomialFeatures, StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split, KFold, GridSearchCV, cross_val_score
from sklearn.metrics import (r2_score, mean_squared_error, accuracy_score, precision_score, recall_score,
                             f1_score, roc_auc_score, roc_curve, precision_recall_curve, make_scorer)
#from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
from sklearn.linear_model import LinearRegression, Lasso, Ridge, ElasticNet#, LogisticRegression
%matplotlib inline
```

Data Translator - Resultados

Creamos la definición del error logarítmico cuadrático medio

Métrica de precisión

Error logarítmico cuadrático medio

In [2]:
```
def rmsle(actual, predictions):
    log_diff = np.log((predictions+1)/(actual+1))
    return np.sqrt(np.mean(log_diff**2))
```

Creamos una función de prueba de modelos

```
def mod_eval(xtrain,xtest,ytrain,ytest,regressor,parameters,score):
    preds=pd.DataFrame(index=list(range(1,25)))

    gs = GridSearchCV(regressor, parameters, cv=5, n_jobs=1, verbose=1, scoring = score)
    gs = GridSearchCV(regressor, parameters, cv=5, n_jobs=1, scoring = score)
    gs.fit(xtrain, ytrain)

    mdl=gs.best_estimator_
    fctest=gs.predict(xtest)

    mdl_metric=gs.best_score_
    tst_metric=rmsle(ytest,fctest)

    return mdl,mdl_metric,tst_metric
```

Data Translator - Resultados

Procedemos con la lectura e inspección de datos

```python
data = pd.read_csv("data/Train.csv")
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 401125 entries, 0 to 401124
Data columns (total 53 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   SalesID                   401125 non-null  int64
 1   SalePrice                 401125 non-null  int64
 2   MachineID                 401125 non-null  int64
 3   ModelID                   401125 non-null  int64
 4   datasource                401125 non-null  int64
 5   auctioneerID              380989 non-null  float64
 6   YearMade                  401125 non-null  int64
 7   MachineHoursCurrentMeter  142765 non-null  float64
 8   UsageBand                 69639 non-null   object
 9   saledate                  401125 non-null  object
 10  fiModelDesc               401125 non-null  object
 11  fiBaseModel               401125 non-null  object
 12  fiSecondaryDesc           263934 non-null  object
 13  fiModelSeries             56908 non-null   object
 14  fiModelDescriptor         71919 non-null   object
 15  ProductSize               190350 non-null  object
 16  fiProductClassDesc        401125 non-null  object
 17  state                     401125 non-null  object
 18  ProductGroup              401125 non-null  object
 19  ProductGroupDesc          401125 non-null  object
 20  Drive_System              104361 non-null  object
 21  Enclosure                 400800 non-null  object
 22  Forks                     192077 non-null  object
 23  Pad_Type                  79134 non-null   object
 24  Ride_Control              148606 non-null  object
 25  Stick                     79134 non-null   object
```

```
 26  Transmission              183230 non-null  object
 27  Turbocharged              79134 non-null   object
 28  Blade_Extension           25219 non-null   object
 29  Blade_Width               25219 non-null   object
 30  Enclosure_Type            25219 non-null   object
 31  Engine_Horsepower         25219 non-null   object
 32  Hydraulics                320570 non-null  object
 33  Pushblock                 25219 non-null   object
 34  Ripper                    104137 non-null  object
 35  Scarifier                 25230 non-null   object
 36  Tip_Control               25219 non-null   object
 37  Tire_Size                 94718 non-null   object
 38  Coupler                   213952 non-null  object
 39  Coupler_System            43458 non-null   object
 40  Grouser_Tracks            43362 non-null   object
 41  Hydraulics_Flow           43362 non-null   object
 42  Track_Type                99153 non-null   object
 43  Undercarriage_Pad_Width   99872 non-null   object
 44  Stick_Length              99218 non-null   object
 45  Thumb                     99288 non-null   object
 46  Pattern_Changer           99218 non-null   object
 47  Grouser_Type              99153 non-null   object
 48  Backhoe_Mounting          78672 non-null   object
 49  Blade_Type                79833 non-null   object
 50  Travel_Controls           79834 non-null   object
 51  Differential_Type         69411 non-null   object
 52  Steering_Controls         69369 non-null   object
dtypes: float64(2), int64(6), object(45)
```

Data Translator - Resultados

Continuamos con la selección de features y limpieza de datos

```
data4preds= data[['YearMade','MachineHoursCurrentMeter','UsageBand','Transmission','ProductGroup','Drive_System','SalePrice']]
data4preds.head()
#scatter_matrix(data4preds, figsize=(15,9));
```

	YearMade	MachineHoursCurrentMeter	UsageBand	Transmission	ProductGroup	Drive_System	SalePrice
0	2004	68.0	Low	NaN	WL	NaN	66000
1	1996	4640.0	Low	NaN	WL	NaN	57000
2	2001	2838.0	High	NaN	SSL	NaN	10000
3	2001	3486.0	High	NaN	TEX	NaN	38500
4	2007	722.0	Medium	NaN	SSL	NaN	11000

```
data4preds['UsageBand'] =data4preds.UsageBand.map({'Low':1, 'High':3, 'Medium':2})
data4preds
```

	YearMade	MachineHoursCurrentMeter	UsageBand	Transmission	ProductGroup	Drive_System	SalePrice
0	2004	68.0	1.0	NaN	WL	NaN	66000
1	1996	4640.0	1.0	NaN	WL	NaN	57000
2	2001	2838.0	3.0	NaN	SSL	NaN	10000
3	2001	3486.0	3.0	NaN	TEX	NaN	38500
4	2007	722.0	2.0	NaN	SSL	NaN	11000
...	...	...	...	...	...	...	...
401120	2005	NaN	NaN	NaN	TEX	NaN	10500
401121	2005	NaN	NaN	NaN	TEX	NaN	11000
401122	2005	NaN	NaN	NaN	TEX	NaN	11500
401123	2005	NaN	NaN	NaN	TEX	NaN	9000
401124	2005	NaN	NaN	NaN	TEX	NaN	7750

Data Translator - Resultados

Continuamos con la selección de features y limpieza de datos

```
dummies = pd.get_dummies(data4preds[['Transmission','ProductGroup','Drive_System']])
data4preds=pd.concat([data4preds,dummies],axis=1).drop(['Transmission','ProductGroup','Drive_System','Transmission_AutoShift'],axis=1)
data4preds
```

	YearMade	MachineHoursCurrentMeter	UsageBand	SalePrice	Transmission_Autoshift	Transmission_Direct Drive	Transmission_Hydrostatic	Transmission_None or Unspecified	Transmission_Powe
0	2004	68.0	1.0	66000	0	0	0	0	
1	1996	4640.0	1.0	57000	0	0	0	0	
2	2001	2838.0	3.0	10000	0	0	0	0	
3	2001	3486.0	3.0	38500	0	0	0	0	
4	2007	722.0	2.0	11000	0	0	0	0	
...	...	...	...	...	...	...	...	...	
401120	2005	NaN	NaN	10500	0	0	0	0	
401121	2005	NaN	NaN	11000	0	0	0	0	
401122	2005	NaN	NaN	11500	0	0	0	0	
401123	2005	NaN	NaN	9000	0	0	0	0	
401124	2005	NaN	NaN	7750					

```
data4preds['UsageBand'].fillna(0,inplace=True)
data4preds
```

	YearMade	MachineHoursCurrentMeter	UsageBand	SalePrice	Transmission_Autoshift	Transmission_Direct Drive	Transmission_Hydrostatic	Transmission_None or Unspecified	Transmission_Powe
0	2004	68.0	1.0	66000	0	0	0	0	
1	1996	4640.0	1.0	57000	0	0	0	0	
2	2001	2838.0	3.0	10000	0	0	0	0	
3	2001	3486.0	3.0	38500	0	0	0	0	
4	2007	722.0	2.0	11000	0	0	0	0	
...	...	...	...	...	...	...	...	...	
401120	2005	NaN	0.0	10500	0	0	0	0	
401121	2005	NaN	0.0	11000	0	0	0	0	
401122	2005	NaN	0.0	11500	0	0	0	0	
401123	2005	NaN	0.0	9000	0	0	0	0	
401124	2005	NaN	0.0	7750	0	0	0	0	

Data Translator - Resultados

Creamos el conjunto de entrenamiento y prueba

```
X_data = data4preds.drop('SalePrice',axis=1)
y_data = data4preds['SalePrice']
```

```
X_train, X_test, y_train, y_test = train_test_split(X_data, y_data, test_size=0.33)
score =make_scorer(rmsle,greater_is_better=False)
```

Data Translator - Resultados

Realizamos un Grid Search de modelos de regresión: LR, Lasso,Ridge, Elastic Net.

Se prueban parámetros separados en escala logarítmica para los últimos tres modelos.

```
mdl_list=[LinearRegression(),Lasso(),Ridge(),ElasticNet()]
param_array=list(np.logspace(-2,4,num=100))
mdl_params=[{},{'alpha':param_array},{'alpha':param_array},{'alpha':param_array,'l1_ratio':list(np.linspace(0.01,0.03,10))}]
best_mdl=[]
scores=[]
for m in range(0,len(mdl_list)):
    res=mod_eval(X_train,X_test,y_train,y_test,mdl_list[m],mdl_params[m],score)
    best_mdl.append(res[0])
    scores.append(res[2])

summary=pd.DataFrame(list(zip(best_mdl,scores)),columns=['Mejor Modelo','Scores'])
```

```
summary
```

	Mejor Modelo	Scores
0	LinearRegression()	0.567405
1	Lasso(alpha=305.38555088334186)	0.555447
2	Ridge(alpha=1232.8467394420659)	0.559299
3	ElasticNet(alpha=0.02009233002565047, l1_ratio...	0.554539

Data Translator - Resultados

Realizamos un Grid Search de modelos de regresión: LR, Lasso,Ridge, Elastic Net.

Se prueban parámetros separados en escala logarítmica para los últimos tres modelos.

```
mdl_list=[LinearRegression(),Lasso(),Ridge(),ElasticNet()]
param_array=list(np.logspace(-2,4,num=100))
mdl_params=[{},{'alpha':param_array},{'alpha':param_array},{'alpha':param_array,'l1_ratio':list(np.linspace(0.01,0.03,10))}]
best_mdl=[]
scores=[]
for m in range(0,len(mdl_list)):
    res=mod_eval(X_train,X_test,y_train,y_test,mdl_list[m],mdl_params[m],score)
    best_mdl.append(res[0])
    scores.append(res[2])

summary=pd.DataFrame(list(zip(best_mdl,scores)),columns=['Mejor Modelo','Scores'])
```

```
summary
```

	Mejor Modelo	Scores
0	LinearRegression()	0.567405
1	Lasso(alpha=305.38555088334186)	0.555447
2	Ridge(alpha=1232.8467394420659)	0.559299
3	ElasticNet(alpha=0.02009233002565047, l1_ratio...	0.554539

Data Translator - Resultados

Selección del mejor modelo con datos de entrenamiento y su conjunto de validación

```
best_mdl=summary[summary['Scores']==min(summary.Scores)].loc[:,'Mejor Modelo'].reset_index()
best_mdl
```

	index	Mejor Modelo
0	3	ElasticNet(alpha=0.02009233002565047, l1_ratio...

```
feats4val = pd.read_csv('data/test.csv').set_index('SalesID')
y4val = pd.read_csv('data/test_actual.csv').set_index('SalesID')
#data4val = data4val.set_index('SalesID')
feats4val.info()
feats4val.describe().T
```

	count	mean	std	min	25%	50%	75%	max
MachineID	11573.0	1.651495e+06	652248.533150	150.0	1067304.0	1862151.0	2270530.0	2485252.0
ModelID	11573.0	8.940136e+03	7807.393696	28.0	3362.0	4763.0	14303.0	37197.0
datasource	11573.0	1.526227e+02	14.872064	121.0	149.0	149.0	172.0	173.0
auctioneerID	11573.0	7.547481e+00	22.307077	0.0	1.0	1.0	3.0	99.0
YearMade	11573.0	1.895332e+03	305.481901	1000.0	1993.0	2001.0	2005.0	2014.0
MachineHoursCurrentMeter	4739.0	5.482141e+03	6391.097182	0.0	1268.0	3786.0	7793.0	89200.0

Data Translator - Resultados

Se lleva a cabo la limpieza de datos de validación

Limpieza de datos de validación

```
data4val = pd.concat([feats4val[['YearMade','MachineHoursCurrentMeter','UsageBand','Transmission','ProductGroup','Drive_System']],
                   y4val['SalePrice']],axis=1)
data4val['UsageBand'] =data4val.UsageBand.map({'Low':1, 'High':3, 'Medium':2})
data4val=data4val[data4val['YearMade']>1960]
dummies = pd.get_dummies(data4val[['Transmission','ProductGroup','Drive_System']])
data4val=pd.concat([data4val,dummies],axis=1).drop(['Transmission','ProductGroup','Drive_System'],axis=1)
data4val['UsageBand'].fillna(0,inplace=True)
data4val.dropna(inplace=True)
```

```
X_val = data4val.drop('SalePrice',axis=1)
y_val = data4val['SalePrice']
```

Data Translator - Resultados

Finalmente se lleva acabo la predicción del mejor modelo

```
mdl_val=best_mdl.loc[:,'Mejor Modelo'][0]
y_hat = mdl_val.predict(X_val)
rmsle(y_val, y_hat)
```

```
0.599499603598123
```

¡Gracias!