

Preprocesado

September 15, 2025

1 Importación de librerías

```
[1]: import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
from sklearn.preprocessing import OrdinalEncoder
```

2 Importación del dataset

```
[2]: df = pd.read_csv('../Data/data_raw.csv')
```

3 Preprocesado

Comienza el preprocesado del dataset a partir del EDA realizado.

3.1 Limpieza de datos

Eliminar filas con:

- **DayOfWeekClaimed** = 0
- **MonthClaimed** = 0

```
[3]: df = df[df['DayOfWeekClaimed'] != '0']
df = df[df['MonthClaimed'] != '0']
```

3.2 Eliminación de variables no útiles

Eliminar:

- **PolicyNumber** y **RepNumber** (son IDs que no aportan información útil).
- **PolicyType** y **Age** (por redundancia extrema).
- **DriverRating** (por falta de información).
- **Year** (por no poder generalizar a años futuros).

```
[4]: df.drop(columns=['PolicyNumber', 'RepNumber', 'PolicyType', 'DriverRating',
↳ 'Age', 'Year'], inplace=True)
```

3.3 Conversión de variables categóricas a simples booleanas

Variable	Cómo transformar
AccidentArea	Urban = 1, Rural = 0
Sex	Male = 1, Female = 0
Fault	Third Party = 1, Policy Holder = 0
PoliceReportFiled	Yes = 1, No = 0
WitnessPresent	Yes = 1, No = 0
AgentType	Internal = 1, External = 0

```
[5]: binary_cols = ['AccidentArea', 'Sex', 'Fault', 'PoliceReportFiled',
↳ 'WitnessPresent', 'AgentType']

# Diccionario para guardar los encoders
encoders = {}

for col in binary_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    encoders[col] = le
    print(f"{col} mapping: {dict(zip(le.classes_, le.transform(le.classes_)))}")
```

AccidentArea mapping: {'Rural': 0, 'Urban': 1}

Sex mapping: {'Female': 0, 'Male': 1}

Fault mapping: {'Policy Holder': 0, 'Third Party': 1}

PoliceReportFiled mapping: {'No': 0, 'Yes': 1}

WitnessPresent mapping: {'No': 0, 'Yes': 1}

AgentType mapping: {'External': 0, 'Internal': 1}

3.4 Conversión de variables con rangos a valores medios

```
[6]: col_map = [
    {'PastNumberOfClaims': {
        'none': 0,
        '1': 1,
        '2 to 4': 3,
        'more than 4': 5
    }},
    {'AgeOfPolicyHolder': {
        '16 to 17': 16.5,
        '18 to 20': 19,
        '21 to 25': 23,
        '26 to 30': 28,
```

```

    '31 to 35': 33,
    '36 to 40': 38,
    '41 to 50': 45.5,
    '51 to 65': 58,
    'over 65': 66
  }},
  {'NumberOfSuppliments': {
    'none': 0,
    '1 to 2': 1.5,
    '3 to 5': 4,
    'more than 5': 6
  }},
  {'VehiclePrice': {
    '20000 to 29000': 24500,
    '30000 to 39000': 34500,
    '40000 to 59000': 49500,
    '60000 to 69000': 64500,
    'less than 20000': 15000,
    'more than 69000': 70000
  }},
  {'Days_Policy_Accident': {
    'none': 0,
    '1 to 7': 4,
    '8 to 15': 11.5,
    '15 to 30': 22.5,
    'more than 30': 35
  }},
  {'Days_Policy_Claim': {
    'none': 0,
    '8 to 15': 11.5,
    '15 to 30': 22.5,
    'more than 30': 35
  }},
  {'AgeOfVehicle': {
    '2 years': 2,
    '3 years': 3,
    '4 years': 4,
    '5 years': 5,
    '6 years': 6,
    '7 years': 7,
    'more than 7': 8,
    'new': 0.5
  }},
  {'NumberOfCars': {
    '1 vehicle': 1,
    '2 vehicles': 2,
    '3 to 4': 3.5,

```

```

        '5 to 8': 6.5,
        'more than 8': 9
    }}
]

# Aplicar los reemplazos
for mapping in col_map:
    df.replace(mapping, inplace=True)

# Convertir a float
for mapping in col_map:
    col = list(mapping.keys())[0]
    df[col] = df[col].astype(float)

print(df[[list(m.keys())[0] for m in col_map]].dtypes)

```

```

PastNumberOfClaims      float64
AgeOfPolicyHolder       float64
NumberOfSupplements     float64
VehiclePrice            float64
Days_Policy_Accident    float64
Days_Policy_Claim       float64
AgeOfVehicle            float64
NumberOfCars            float64
dtype: object

```

3.5 Codificación de variables categóricas

El **OrdinalEncoder** es más adecuado que el **One-Hot Encoding** por las siguientes razones:

1. **Número limitado de categorías:** las variables listadas tienen un número limitado de categorías que son fundamentales para la predicción, pero no son necesariamente independientes entre sí. Por ejemplo, los meses del año (**Month**), o el día de la semana (**DayOfWeek**), no deberían ser tratadas como variables independientes, ya que el orden tiene un impacto en la relación entre las categorías.
2. **Evitar la expansión de la dimensionalidad:** utilizar **One-Hot Encoding** en variables con muchas categorías (como **Make**, que tiene muchas marcas de vehículos diferentes) puede aumentar de manera considerable el número de características del conjunto de datos. Esto podría resultar en un modelo más complejo y un mayor riesgo de sobreajuste, especialmente si las categorías tienen poca representación en los datos.

```

[7]: categorical_cols = df.select_dtypes(include=['object']).columns

cat_summary = pd.DataFrame({
    "Variable": categorical_cols,
    "Nº Valores únicos": [df[col].nunique() for col in categorical_cols],
    "Valores": [df[col].unique().tolist() for col in categorical_cols]
})

```

```
print(cat_summary)
```

	Variable	Nº Valores únicos \
0	Month	12
1	DayOfWeek	7
2	Make	19
3	DayOfWeekClaimed	7
4	MonthClaimed	12
5	MaritalStatus	4
6	VehicleCategory	3
7	AddressChange_Claim	5
8	BasePolicy	3

	Valores
0	[Dec, Jan, Oct, Jun, Feb, Nov, Apr, Mar, Aug, ...
1	[Wednesday, Friday, Saturday, Monday, Tuesday,...
2	[Honda, Toyota, Ford, Mazda, Chevrolet, Pontia...
3	[Tuesday, Monday, Thursday, Friday, Wednesday,...
4	[Jan, Nov, Jul, Feb, Mar, Dec, Apr, Aug, May, ...
5	[Single, Married, Widow, Divorced]
6	[Sport, Utility, Sedan]
7	[1 year, no change, 4 to 8 years, 2 to 3 years...
8	[Liability, Collision, All Perils]

```
[8]: # Diccionarios con el orden secuencial
month_order = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
dayofweek_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
make_order = ['Accura', 'BMW', 'Chevrolet', 'Dodge', 'Ferrari', 'Ford', 'Honda', 'Jaguar', 'Lexus', 'Mazda', 'Mecce
marital_order = ['Divorced', 'Married', 'Single', 'Widow']
vehiclecat_order = ['Sedan', 'Sport', 'Utility']
basepolicy_order = ['All Perils', 'Collision', 'Liability']
addresschange_order = ['under 6 months', 'no change', '1 year', '2 to 3 years', '4 to 8 years']

# Lista de variables categóricas
cat_vars = [
    'Month',
    'DayOfWeek',
    'Make',
    'DayOfWeekClaimed',
    'MonthClaimed',
    'MaritalStatus',
    'VehicleCategory',
```

```

    'BasePolicy',
    'AddressChange_Claim'
]
categories_list = [
    month_order,
    dayofweek_order,
    make_order,
    dayofweek_order,
    month_order,
    marital_order,
    vehiclecat_order,
    basepolicy_order,
    addresschange_order
]

encoder = OrdinalEncoder(categories=categories_list)

df[cat_vars] = encoder.fit_transform(df[cat_vars])

print("Categorías asignadas a cada columna:")
for feature, categories in zip(cat_vars, encoder.categories_):
    print(f"{feature}: {dict(zip(categories, range(len(categories))))}")

```

Categorías asignadas a cada columna:

Month: {'Jan': 0, 'Feb': 1, 'Mar': 2, 'Apr': 3, 'May': 4, 'Jun': 5, 'Jul': 6, 'Aug': 7, 'Sep': 8, 'Oct': 9, 'Nov': 10, 'Dec': 11}

DayOfWeek: {'Monday': 0, 'Tuesday': 1, 'Wednesday': 2, 'Thursday': 3, 'Friday': 4, 'Saturday': 5, 'Sunday': 6}

Make: {'Accura': 0, 'BMW': 1, 'Chevrolet': 2, 'Dodge': 3, 'Ferrari': 4, 'Ford': 5, 'Honda': 6, 'Jaguar': 7, 'Lexus': 8, 'Mazda': 9, 'Mecedes': 10, 'Mercury': 11, 'Nissan': 12, 'Pontiac': 13, 'Porche': 14, 'Saab': 15, 'Saturn': 16, 'Toyota': 17, 'VW': 18}

DayOfWeekClaimed: {'Monday': 0, 'Tuesday': 1, 'Wednesday': 2, 'Thursday': 3, 'Friday': 4, 'Saturday': 5, 'Sunday': 6}

MonthClaimed: {'Jan': 0, 'Feb': 1, 'Mar': 2, 'Apr': 3, 'May': 4, 'Jun': 5, 'Jul': 6, 'Aug': 7, 'Sep': 8, 'Oct': 9, 'Nov': 10, 'Dec': 11}

MaritalStatus: {'Divorced': 0, 'Married': 1, 'Single': 2, 'Widow': 3}

VehicleCategory: {'Sedan': 0, 'Sport': 1, 'Utility': 2}

BasePolicy: {'All Perils': 0, 'Collision': 1, 'Liability': 2}

AddressChange_Claim: {'under 6 months': 0, 'no change': 1, '1 year': 2, '2 to 3 years': 3, '4 to 8 years': 4}

```
[9]: df.head()
```

```

[9]:   Month  WeekOfMonth  DayOfWeek  Make  AccidentArea  DayOfWeekClaimed  \
0    11.0           5         2.0    6.0              1              1.0
1     0.0           3         2.0    6.0              1              0.0
2     9.0           5         4.0    6.0              1              3.0

```

3	5.0	2	5.0	17.0	0	4.0
4	0.0	5	0.0	6.0	1	1.0

	MonthClaimed	WeekOfMonthClaimed	Sex	MaritalStatus	...	\
0	0.0		1	0	2.0	...
1	0.0		4	1	2.0	...
2	10.0		2	1	1.0	...
3	6.0		1	1	1.0	...
4	1.0		2	0	2.0	...

	PastNumberOfClaims	AgeOfVehicle	AgeOfPolicyHolder	PoliceReportFiled	\
0	0.0	3.0	28.0	0	
1	0.0	6.0	33.0	1	
2	1.0	7.0	45.5	0	
3	1.0	8.0	58.0	1	
4	0.0	5.0	33.0	0	

	WitnessPresent	AgentType	NumberOfSuppliments	AddressChange_Claim	\
0	0	0	0.0	2.0	
1	0	0	0.0	1.0	
2	0	0	0.0	1.0	
3	0	0	6.0	1.0	
4	0	0	0.0	1.0	

	NumberOfCars	BasePolicy
0	3.5	2.0
1	1.0	1.0
2	1.0	1.0
3	1.0	2.0
4	1.0	1.0

[5 rows x 27 columns]

4 Guardado de dataset preprocesado

```
[10]: df.to_csv("../Data/data_processed.csv", index=False)
```