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



#Cargar archivo csv desde equipo
from google.colab import files
files.upload()
```

```
#Carga desde un archivo .csv sin indice y con decodificación. Ejemplos: 'ascii', 'UTF-8'
df = pd.read_csv("cuentas_credicel.csv", encoding= 'latin')
df.head(10)
```

```
#Corroboramos valores nulos
valores_nulos=df.isnull().sum()
valores_nulos
```

```
folio                   0
tag                     0
folio_solicitud         0
fecha                   0
marca                   0
modelo                  0
plazo                   0
precio                  0
enganche                0
descuento               0
semana                  0
monto_financiado        0
costo_total             0
monto_accesorios        0
agente_venta            0
```

```
dis_venta                0
status                   0
fraude                   0
empresa                 13
inversion                0
pagos_realizados         0
reautorizacion           0
fecha_ultimo_pago     3356
fecha_pago_proximo    3356
status_cuenta         4197
puntos                3595
riesgo                2473
porc_enganche         2473
porc_tasa                0
score_buro               0
razones_buro          4505
semana_actual         4505
codigo_postal          685
Unnamed: 33          21876
Unnamed: 34          21532
Unnamed: 35          20262
Unnamed: 36          22734
dtype: int64
```

```
df1=df.copy()
```

```
#Quinto método de sustitución de valores nulos
#Sustituir valores nulos por valores no nulos hacia adelante  "forward fill" ("ffill")
#Filtro por columnas
df1["empresa"] =df1["empresa"].fillna(method="ffill")
```

```
#Quinto método de sustitución de valores nulos
#Sustituir valores nulos por valores no nulos hacia adelante  "forward fill" ("ffill")
#Filtro por columnas
df1["fecha_ultimo_pago"] =df1["fecha_ultimo_pago"].fillna(method="ffill")
```

```
#Sexto método de sustitución de valores nulos
#Sustituir valores nulos por valores no nulos hacia atrás backward fill" ("bfill")
#Filtro por columnas
df1["fecha_ultimo_pago"] =df1["fecha_ultimo_pago"].fillna(method="bfill")
```

```
#Quinto método de sustitución de valores nulos
#Sustituir valores nulos por valores no nulos hacia adelante  "forward fill" ("ffill")
#Filtro por columnas
df1["fecha_pago_proximo"] =df1["fecha_pago_proximo"].fillna(method="ffill")
```

```
#Sexto método de sustitución de valores nulos
#Sustituir valores nulos por valores no nulos hacia atrás backward fill" ("bfill")
#Filtro por columnas
df1["fecha_pago_proximo"] =df1["fecha_pago_proximo"].fillna(method="bfill")
```

```
#Quinto método de sustitución de valores nulos
#Sustituir valores nulos por valores no nulos hacia adelante  "forward fill" ("ffill")
#Filtro por columnas
df1["status_cuenta"] =df1["status_cuenta"].fillna(method="ffill")
```

```
#Sexto método de sustitución de valores nulos
#Sustituir valores nulos por valores no nulos hacia atrás backward fill" ("bfill")
#Filtro por columnas
df1["status_cuenta"] =df1["status_cuenta"].fillna(method="bfill")
```

```
#Primer método de sustitución de valores nulos
#Sustituir valores nulos con promedio o media
df1["puntos"]=df1["puntos"].fillna(round(df["puntos"].mean(),1))
```

```
#Quinto método de sustitución de valores nulos
#Sustituir valores nulos por valores no nulos hacia adelante  "forward fill" ("ffill")
#Filtro por columnas
df1["riesgo"] =df1["riesgo"].fillna(method="ffill")
```

```
#Sexto método de sustitución de valores nulos
#Sustituir valores nulos por valores no nulos hacia atrás backward fill" ("bfill")
#Filtro por columnas
df1["riesgo"] =df1["riesgo"].fillna(method="bfill")
```

```
#Primer método de sustitución de valores nulos
#Sustituir valores nulos con promedio o media
df1["porc_enganche"]=df1["porc_enganche"].fillna(round(df["porc_enganche"].mean(),1))


#Quinto método de sustitución de valores nulos
#Sustituir valores nulos por valores no nulos hacia adelante  "forward fill" ("ffill")
#Filtro por columnas
df1["razones_buro"] =df1["razones_buro"].fillna(method="ffill")


#Quinto método de sustitución de valores nulos
#Sustituir valores nulos por valores no nulos hacia adelante  "forward fill" ("ffill")
#Filtro por columnas
df1["semana_actual"] =df1["semana_actual"].fillna(method="ffill")


#Quinto método de sustitución de valores nulos
#Sustituir valores nulos por valores no nulos hacia adelante  "forward fill" ("ffill")
#Filtro por columnas
df1["codigo_postal"] =df1["codigo_postal"].fillna(method="ffill")


#Quinto método de sustitución de valores nulos
#Sustituir valores nulos por valores no nulos hacia adelante  "forward fill" ("ffill")
#Filtro por columnas
df1["Unnamed: 33"] =df1["Unnamed: 33"].fillna(method="ffill")


#Sexto método de sustitución de valores nulos
#Sustituir valores nulos por valores no nulos hacia atrás backward fill" ("bfill")
#Filtro por columnas
df1["Unnamed: 33"] =df1["Unnamed: 33"].fillna(method="bfill")


#Quinto método de sustitución de valores nulos
#Sustituir valores nulos por valores no nulos hacia adelante  "forward fill" ("ffill")
#Filtro por columnas
df1["Unnamed: 34"] =df1["Unnamed: 34"].fillna(method="ffill")


#Sexto método de sustitución de valores nulos
#Sustituir valores nulos por valores no nulos hacia atrás backward fill" ("bfill")
#Filtro por columnas
df1["Unnamed: 34"] =df1["Unnamed: 34"].fillna(method="bfill")


#Primer método de sustitución de valores nulos
#Sustituir valores nulos con promedio o media
df1["Unnamed: 35"]=df1["Unnamed: 35"].fillna(round(df["Unnamed: 35"].mean(),1))


#Primer método de sustitución de valores nulos
#Sustituir valores nulos con promedio o media
df1["Unnamed: 36"]=df1["Unnamed: 36"].fillna(round(df["Unnamed: 36"].mean(),1))


#Corroboramos valores nulos
valores_nulos=df1.isnull().sum()
valores_nulos
```

```
folio                0
tag                  0
folio_solicitud      0
fecha                0
marca                0
modelo               0
plazo                0
precio               0
enganche             0
descuento            0
semana               0
monto_financiado     0
costo_total          0
monto_accesorios     0
agente_venta         0
dis_venta            0
status               0
fraude               0
empresa              0
inversion            0
pagos_realizados     0
reautorizacion       0
fecha_ultimo_pago    0
fecha_pago_proximo   0
status_cuenta        0
```

```
puntos               0
riesgo               0
porc_enganche        0
porc_tasa            0
score_buro           0
razones_buro         0
semana_actual        0
codigo_postal        0
Unnamed: 33          0
Unnamed: 34          0
Unnamed: 35          0
Unnamed: 36          0
dtype: int64
```

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22735 entries, 0 to 22734
Data columns (total 37 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   folio               22735 non-null  int64  
 1   tag                 22735 non-null  object 
 2   folio_solicitud     22735 non-null  int64  
 3   fecha               22735 non-null  object 
 4   marca               22735 non-null  object 
 5   modelo              22735 non-null  object 
 6   plazo               22735 non-null  object 
 7   precio              22735 non-null  float64
 8   enganche            22735 non-null  float64
 9   descuento           22735 non-null  float64
 10  semana              22735 non-null  int64  
 11  monto_financiado    22735 non-null  float64
 12  costo_total         22735 non-null  int64  
 13  monto_accesorios    22735 non-null  float64
 14  agente_venta        22735 non-null  object 
 15  dis_venta           22735 non-null  object 
 16  status              22735 non-null  int64  
 17  fraude              22735 non-null  int64  
 18  empresa             22735 non-null  object 
 19  inversion           22735 non-null  int64  
 20  pagos_realizados    22735 non-null  int64  
 21  reautorizacion      22735 non-null  int64  
 22  fecha_ultimo_pago   22735 non-null  object 
 23  fecha_pago_proximo  22735 non-null  object 
 24  status_cuenta       22735 non-null  object 
 25  puntos              22735 non-null  float64
 26  riesgo              22735 non-null  object 
 27  porc_enganche       22735 non-null  float64
 28  porc_tasa           22735 non-null  float64
 29  score_buro          22735 non-null  float64
 30  razones_buro        22735 non-null  object 
 31  semana_actual       22735 non-null  object 
 32  codigo_postal       22735 non-null  object 
 33  Unnamed: 33         22735 non-null  object 
 34  Unnamed: 34         22735 non-null  object 
 35  Unnamed: 35         22735 non-null  float64
 36  Unnamed: 36         22735 non-null  float64
dtypes: float64(11), int64(9), object(17)
memory usage: 6.4+ MB
```

Convertir columna plazo a entero, quitando el carácter "S"

```
#Eliminar un signo de una columna
df1['plazo']=df1['plazo'].str.replace('S', '')
```

```
#Conversión de tipo de dato de columna de tipo Object a int
df1['plazo']= df1['plazo'].astype(int)
```

```
#Compruebo que la columna plazo ya se halla convertido a tipo numérico
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22735 entries, 0 to 22734
Data columns (total 37 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   folio               22735 non-null  int64  
 1   tag                 22735 non-null  object 
 2   folio_solicitud     22735 non-null  int64  
 3   fecha               22735 non-null  object 
 4   marca               22735 non-null  object 
 5   modelo              22735 non-null  object 
 6   plazo               22735 non-null  int64
```

```
 7   precio              22735 non-null  float64
 8   enganche            22735 non-null  float64
 9   descuento           22735 non-null  float64
 10  semana              22735 non-null  int64
 11  monto_financiado    22735 non-null  float64
 12  costo_total         22735 non-null  int64
 13  monto_accesorios    22735 non-null  float64
 14  agente_venta        22735 non-null  object
 15  dis_venta           22735 non-null  object
 16  status              22735 non-null  int64
 17  fraude              22735 non-null  int64
 18  empresa             22735 non-null  object
 19  inversion           22735 non-null  int64
 20  pagos_realizados    22735 non-null  int64
 21  reautorizacion      22735 non-null  int64
 22  fecha_ultimo_pago   22735 non-null  object
 23  fecha_pago_proximo  22735 non-null  object
 24  status_cuenta       22735 non-null  object
 25  puntos              22735 non-null  float64
 26  riesgo              22735 non-null  object
 27  porc_enganche       22735 non-null  float64
 28  porc_tasa           22735 non-null  float64
 29  score_buro          22735 non-null  float64
 30  razones_buro        22735 non-null  object
 31  semana_actual       22735 non-null  object
 32  codigo_postal       22735 non-null  object
 33  Unnamed: 33         22735 non-null  object
 34  Unnamed: 34         22735 non-null  object
 35  Unnamed: 35         22735 non-null  float64
 36  Unnamed: 36         22735 non-null  float64
dtypes: float64(11), int64(10), object(16)
memory usage: 6.4+ MB
```

Convertir columna riesgo a entero, quitando los carácteres existentes

```
#Sustituyo valores nulos por el numero "0"
#Tercer método de sustitución de valores nulos
#Sustituir valores nulos por un valor numérico en  concreto
df1["riesgo"]= df1["riesgo"].fillna(0)
```

```
#Analizar categorias de una columna
riesgo_categorias = df1.groupby(['riesgo'])['riesgo'].count()
riesgo_categorias
```

```
riesgo
-35.0           1
-34.7           2
-34.4           1
-33.7           1
-33.35          1
               ...
9.95            5
91              1
Atraso          1
Cancelado     125
Fraude       1125
Name: riesgo, Length: 2743, dtype: int64
```

```
#Sustituyo los strings Atraso, Cancelado y Fraude por la constante "0"
df1['riesgo']=df1['riesgo'].str.replace('Atraso', '0')
df1['riesgo']=df1['riesgo'].str.replace('Cancelado', '0')
df1['riesgo']=df1['riesgo'].str.replace('Fraude', '0')
```

```
#Conversión de tipo de dato de columna de tipo Object a int
df1['riesgo']= df1['riesgo'].astype(float)
```

```
#Compruebo que la columna riesgo ya se halla convertido a tipo numérico
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22735 entries, 0 to 22734
Data columns (total 37 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   folio               22735 non-null  int64
 1   tag                 22735 non-null  object
 2   folio_solicitud     22735 non-null  int64
 3   fecha               22735 non-null  object
 4   marca               22735 non-null  object
 5   modelo              22735 non-null  object
 6   plazo               22735 non-null  int64
 7   precio              22735 non-null  float64
 8   enganche            22735 non-null  float64
```

```
 9   descuento           22735 non-null  float64
 10  semana              22735 non-null  int64
 11  monto_financiado    22735 non-null  float64
 12  costo_total         22735 non-null  int64
 13  monto_accesorios    22735 non-null  float64
 14  agente_venta        22735 non-null  object
 15  dis_venta           22735 non-null  object
 16  status              22735 non-null  int64
 17  fraude              22735 non-null  int64
 18  empresa             22735 non-null  object
 19  inversion           22735 non-null  int64
 20  pagos_realizados    22735 non-null  int64
 21  reautorizacion      22735 non-null  int64
 22  fecha_ultimo_pago   22735 non-null  object
 23  fecha_pago_proximo  22735 non-null  object
 24  status_cuenta       22735 non-null  object
 25  puntos              22735 non-null  float64
 26  riesgo              16384 non-null  float64
 27  porc_enganche       22735 non-null  float64
 28  porc_tasa           22735 non-null  float64
 29  score_buro          22735 non-null  float64
 30  razones_buro        22735 non-null  object
 31  semana_actual       22735 non-null  object
 32  codigo_postal       22735 non-null  object
 33  Unnamed: 33         22735 non-null  object
 34  Unnamed: 34         22735 non-null  object
 35  Unnamed: 35         22735 non-null  float64
 36  Unnamed: 36         22735 non-null  float64
dtypes: float64(12), int64(10), object(15)
memory usage: 6.4+ MB
```

```
#Corroboramos valores nulos
valores_nulos=df1.isnull().sum()
valores_nulos
```

```
folio                   0
tag                     0
folio_solicitud         0
fecha                   0
marca                   0
modelo                  0
plazo                   0
precio                  0
enganche                0
descuento               0
semana                  0
monto_financiado        0
costo_total             0
monto_accesorios        0
agente_venta            0
dis_venta               0
status                  0
fraude                  0
empresa                 0
inversion               0
pagos_realizados        0
reautorizacion          0
fecha_ultimo_pago       0
fecha_pago_proximo      0
status_cuenta           0
puntos                  0
riesgo               6351
porc_enganche           0
porc_tasa               0
score_buro              0
razones_buro            0
semana_actual           0
codigo_postal           0
Unnamed: 33             0
Unnamed: 34             0
Unnamed: 35             0
Unnamed: 36             0
dtype: int64
```

```
#Primer método de sustitución de valores nulos
#Sustituir valores nulos con promedio o media
#df1["riesgo"]=df1["riesgo"].fillna(round(df["riesgo"].mean(),1))
df1["riesgo"]=df1["riesgo"].fillna(method="ffill")
```

```
#Corroboramos valores nulos
valores_nulos=df1.isnull().sum()
valores_nulos
```

```
folio                0
tag                  0
folio_solicitud      0
```

```
fecha                0
marca                0
modelo               0
plazo                0
precio               0
enganche             0
descuento            0
semana               0
monto_financiado     0
costo_total          0
monto_accesorios     0
agente_venta         0
dis_venta            0
status               0
fraude               0
empresa              0
inversion            0
pagos_realizados     0
reautorizacion       0
fecha_ultimo_pago    0
fecha_pago_proximo   0
status_cuenta        0
puntos               0
riesgo               0
porc_enganche        0
porc_tasa            0
score_buro           0
razones_buro         0
semana_actual        0
codigo_postal        0
Unnamed: 33          0
Unnamed: 34          0
Unnamed: 35          0
Unnamed: 36          0
dtype: int64
```

```
#Filtro para obtener variables cuantitativas
Cuantitativas=df1.iloc[ : , [0,2,6,7,8,9,10,11,12,13,16,17,19,20,21,25,26,27,28,29,35,36]]
Cuantitativas
```

```
#Filtro para obtener variables cualitativas
Cualitativas=df1.iloc[ : , [1,3,4,5,14,15,18,22,23,24,30,31,32,33,34]]
Cualitativas
```

PROCEDIMIENTO "DESVIACIÓN ESTÁNDAR" PARA ELIMINAR OUTLIERS EN DATAFRAME

```
#Método aplicando desviación estandar. Encuentro los valores extremos
y=Cuantitativas
Limite_Superior= y.mean() + 3*y.std()
Limite_Inferior= y.mean() - 3*y.std()
print("Limite superior permitido", Limite_Superior)
print("Limite inferior permitido", Limite_Inferior)
```

```
Limite superior permitido folio               31323.107698
folio_solicitud     127723.906961
plazo                   55.874879
precio                8081.558082
enganche              2356.496845
descuento              478.215020
semana                 541.287392
monto_financiado      6366.317810
costo_total          12857.588246
monto_accesorios       191.260880
status                   2.611178
fraude                   0.864131
inversion                1.511599
pagos_realizados        39.306132
reautorizacion           0.578086
puntos                  42.386431
riesgo                  58.047951
porc_enganche           18.135840
porc_tasa               24.630537
score_buro              13.931194
Unnamed: 35          36884.839233
Unnamed: 36          93080.000000
dtype: float64
Limite inferior permitido folio              -8301.055796
folio_solicitud    -36023.715538
plazo                  -4.055569
precio                -87.484900
enganche             -503.622066
descuento            -321.263535
semana                -65.792692
monto_financiado     -214.044583
costo_total         -1458.203597
monto_accesorios     -164.333344
status                 -0.100423
fraude                 -0.714494
inversion              -1.038408
pagos_realizados      -18.944575
reautorizacion         -0.509998
puntos                -14.217221
riesgo                -42.470220
porc_enganche         -16.334824
porc_tasa             -21.606103
score_buro            -14.442358
```

```
Unnamed: 35        -16962.239048
Unnamed: 36         93080.000000
dtype: float64
```

```
# Ajustar maximo de filas
pd.options.display.max_rows = None
```

```
#Obtenemos datos y los outliers se convierten en nulos en el DataFrame
Datos_sin_Outliers= Cuantitativas[(y<=Limite_Superior)&(y>=Limite_Inferior)]
Datos_sin_Outliers
```

```
#Corroboramos valores nulos
valores_nulos=Datos_sin_Outliers.isnull().sum()
valores_nulos
```

```
folio                  0
folio_solicitud        0
plazo                  0
precio               110
enganche             292
descuento             17
semana               362
monto_financiado     182
costo_total          290
monto_accesorios     617
status               154
fraude              1701
inversion              0
pagos_realizados     184
```

```
reautorizacion       774
puntos               954
riesgo                82
porc_enganche        486
porc_tasa            899
score_buro           582
Unnamed: 35          297
Unnamed: 36            0
dtype: int64
```

```
#Reemplazamos valores atípicos (nulos) del dataframe con "mean"
#Realizamos una copia del dataframe
data_clean=Datos_sin_Outliers.copy()
data_clean=data_clean.fillna(round(Datos_sin_Outliers.mean(),1))
data_clean
```

```
#Corroboramos valores nulos
valores_nulos=data_clean.isnull().sum()
valores_nulos
```

```
folio               0
folio_solicitud     0
plazo               0
precio              0
enganche            0
descuento           0
semana              0
monto_financiado    0
costo_total         0
```

```
monto_accesorios    0
status              0
fraude              0
inversion           0
pagos_realizados    0
reautorizacion      0
puntos              0
riesgo              0
porc_enganche       0
porc_tasa           0
score_buro          0
Unnamed: 35         0
Unnamed: 36         0
dtype: int64
```

**Análisis de correlación

“Enganche”

```
#Imprimimos el scatter plot entre la variable dependiente (total) e independiente (alchool)
#para observar el comportamiento en su dispersión
from turtle import color
sns.scatterplot(x='porc_enganche', y='enganche', color="blue", data=Cuantitativas)
sns.scatterplot(x='score_buro', y='enganche', color="red", data=Cuantitativas)
```

```
#Declaramos las variables dependientes e independientes para la regresión lineal
#Vars_Indep= df[['alcohol', 'speeding']]
Vars_Indep= Cuantitativas[['score_buro']]
Var_Dep= Cuantitativas['enganche']
```

```
#Se define model como la función de regresión lineal
from sklearn.linear_model import LinearRegression
model1= LinearRegression()
```

```
#Ajustamos el modelo con las variables antes declaradas
model1.fit(X=Vars_Indep, y=Var_Dep)
```

```
#Verificamos los coeficientes obtenidos para el modelo ajustado
model1.__dict__
```

```
{'fit_intercept': True,
 'copy_X': True,
 'n_jobs': None,
 'positive': False,
 'feature_names_in_': array(['score_buro'], dtype=object),
 'n_features_in_': 1,
 'coef_': array([8.30362837]),
 'rank_': 1,
```

```
      'singular_': array([713.01781912]),
      'intercept_': 928.5596445591957}
```

Modelo matemático: y= 8.30362837x + 928.5596445591957

```
#Predecimos los valores de total de accidentes a partir de la variable "alcohol"
#y_pred= model.predict(X=df[['alcohol', 'speeding']])
y_pred= model1.predict(X=Cuantitativas[['score_buro']])
y_pred
```

```
     array([928.55964456, 928.55964456, 928.55964456, ..., 928.55964456,
            928.55964456, 928.55964456])
```

```
#Insertamos la columna de predicciones en el DataFrame
Cuantitativas.insert(0, 'Prediccion', y_pred)
Cuantitativas
```

```
#Visualizamos la gráfica comparativa entre el total real y el total predecido

sns.scatterplot(x='score_buro', y='enganche', color="blue", data=Cuantitativas)
sns.scatterplot(x='score_buro', y='Prediccion', color="red", data=Cuantitativas)
#sns.lineplot(x='alcohol', y='Predicciones', color="red", data=df)
```

```
#Corroboramos cual es el coeficiente de Determinación de nuestro modelo
coef_Deter=model1.score(X=Vars_Indep, y=Var_Dep)
coef_Deter
```

```
0.006785706027080618
```

```
#Corroboramos cual es el coeficiente de Correlación de nuestro modelo
coef_Correl=np.sqrt(coef_Deter)
coef_Correl
```

```
0.08237539697677103
```

“Riesgo”

```
#Imprimimos el scatter plot entre la variable dependiente (total) e independiente (alchool)
#para observar el comportamiento en su dispersión
from turtle import color
sns.scatterplot(x='descuento', y='riesgo', color="blue", data=Cuantitativas)
sns.scatterplot(x='precio', y='riesgo', color="red", data=Cuantitativas)
```

```
#Declaramos las variables dependientes e independientes para la regresión lineal
#Vars_Indep= df[['alcohol', 'speeding']]
Vars_Indep1= Cuantitativas[['descuento']]
Var_Dep1= Cuantitativas['riesgo']
```

```
#Se define model como la función de regresión lineal
from sklearn.linear_model import LinearRegression
model2= LinearRegression()
```

```
#Ajustamos el modelo con las variables antes declaradas
model2.fit(X=Vars_Indep1, y=Var_Dep1)
```

```
#Verificamos los coeficientes obtenidos para el modelo ajustado
model2.__dict__
```

```
{'fit_intercept': True,
 'copy_X': True,
 'n_jobs': None,
 'positive': False,
 'feature_names_in_': array(['descuento'], dtype=object),
 'n_features_in_': 1,
 'coef_': array([0.0092446]),
 'rank_': 1,
 'singular_': array([20090.62726427]),
 'intercept_': 7.063387975837218}
```

Modelo matemático: y= 0.0092446x + 7.063387975837218

```
#Predecimos los valores de total de accidentes a partir de la variable "alcohol"
#y_pred= model.predict(X=df[['alcohol', 'speeding']])
y_pred2= model2.predict(X=Cuantitativas[['descuento']])
y_pred2
```

```
array([7.06338798, 7.06338798, 7.06338798, ..., 7.06338798, 7.06338798,
       7.06338798])
```

```
#Insertamos la columna de predicciones en el DataFrame
Cuantitativas.insert(0, 'Predicciones2', y_pred2)
Cuantitativas
```

```
#Visualizamos la gráfica comparativa entre el total real y el total predecido

sns.scatterplot(x='descuento', y='riesgo', color="blue", data=Cuantitativas)
sns.scatterplot(x='descuento', y='Predicciones2', color="red", data=Cuantitativas)
#sns.lineplot(x='alcohol', y='Predicciones', color="red", data=df)
```

```
#Corroboramos cual es el coeficiente de Determinación de nuestro modelo
coef_Deter2=model2.score(X=Vars_Indep1, y=Var_Dep1)
coef_Deter2
```

```
0.005406312743220121
```

```
#Corroboramos cual es el coeficiente de Correlación de nuestro modelo
coef_Correl2=np.sqrt(coef_Deter2)
coef_Correl2
```

```
0.0735276325147228
```

“Precio”

```
#Imprimimos el scatter plot entre la variable dependiente (total) e independiente (alchool)
#para observar el comportamiento en su dispersión
from turtle import color
sns.scatterplot(x='descuento', y='precio', color="blue", data=Cuantitativas)
sns.scatterplot(x='monto_financiado', y='precio', color="red", data=Cuantitativas)
```

```
#Declaramos las variables dependientes e independientes para la regresión lineal
#Vars_Indep= df[['alcohol', 'speeding']]
Vars_Indep2= Cuantitativas[['monto_financiado']]
Var_Dep2= Cuantitativas['precio']


#Se define model como la función de regresión lineal
from sklearn.linear_model import LinearRegression
model3= LinearRegression()


#Ajustamos el modelo con las variables antes declaradas
model3.fit(X=Vars_Indep2, y=Var_Dep2)
```

```
#Verificamos los coeficientes obtenidos para el modelo ajustado
model3.__dict__
```

```
{'fit_intercept': True,
 'copy_X': True,
 'n_jobs': None,
 'positive': False,
 'feature_names_in_': array(['monto_financiado'], dtype=object),
 'n_features_in_': 1,
 'coef_': array([1.17573073]),
 'rank_': 1,
 'singular_': array([165362.29426739]),
 'intercept_': 380.32825850329664}
```

Modelo matemático: y= 1.17573073x + 380.32825850329664

```
#Predecimos los valores de total de accidentes a partir de la variable "alcohol"
#y_pred= model.predict(X=df[['alcohol', 'speeding']])
y_pred3= model3.predict(X=Cuantitativas[['monto_financiado']])
y_pred3
```

```
array([1754.75747689, 2671.82744297, 2107.47669462, ..., 4727.00475156,
       8610.44333868, 5451.25487862])
```

```
#Insertamos la columna de predicciones en el DataFrame
Cuantitativas.insert(0, 'Predicciones3', y_pred3)
Cuantitativas
```

```
#Visualizamos la gráfica comparativa entre el total real y el total predecido

sns.scatterplot(x='monto_financiado', y='precio', color="blue", data=Cuantitativas)
sns.scatterplot(x='monto_financiado', y='Predicciones3', color="red", data=Cuantitativas)
#sns.lineplot(x='alcohol', y='Predicciones', color="red", data=df)
```

```
#Corroboramos cual es el coeficiente de Determinación de nuestro modelo
coef_Deter3=model3.score(X=Vars_Indep2, y=Var_Dep2)
coef_Deter3
```

```
0.8969598364305345
```

```
#Corroboramos cual es el coeficiente de Correlación de nuestro modelo
coef_Correl3=np.sqrt(coef_Deter3)
coef_Correl3
```

```
0.9470796357384813
```

“Plazo”

```
#Imprimimos el scatter plot entre la variable dependiente (total) e independiente (alchool)
#para observar el comportamiento en su dispersión
from turtle import color
sns.scatterplot(x='descuento', y='plazo', color="blue", data=Cuantitativas)
sns.scatterplot(x='porc_enganche', y='plazo', color="red", data=Cuantitativas)
```

```
#Declaramos las variables dependientes e independientes para la regresión lineal
#Vars_Indep= df[['alcohol', 'speeding']]
Vars_Indep3= Cuantitativas[['porc_enganche']]
Var_Dep3= Cuantitativas['plazo']
```

```
#Se define model como la función de regresión lineal
from sklearn.linear_model import LinearRegression
model4= LinearRegression()
```

```
#Ajustamos el modelo con las variables antes declaradas
model4.fit(X=Vars_Indep3, y=Var_Dep3)
```

```
#Verificamos los coeficientes obtenidos para el modelo ajustado
model4.__dict__
```

```
{'fit_intercept': True,
 'copy_X': True,
 'n_jobs': None,
 'positive': False,
 'feature_names_in_': array(['porc_enganche'], dtype=object),
 'n_features_in_': 1,
 'coef_': array([-0.07611609]),
 'rank_': 1,
 'singular_': array([866.23618546]),
 'intercept_': 25.97819788407601}
```

Modelo matemático: y= -0.07611609x + 25.97819788407601

```
#Predecimos los valores de total de accidentes a partir de la variable "alcohol"
#y_pred= model.predict(X=df[['alcohol', 'speeding']])
y_pred4= model4.predict(X=Cuantitativas[['porc_enganche']])
y_pred4
```

```
     array([25.9096934 , 25.9096934 , 25.97819788, ..., 23.94970411,
            27.29881203, 24.31886714])
```

```
#Insertamos la columna de predicciones en el DataFrame
Cuantitativas.insert(0, 'Predicciones4', y_pred4)
Cuantitativas
```

```
#Visualizamos la gráfica comparativa entre el total real y el total predecido

sns.scatterplot(x='porc_enganche', y='plazo', color="blue", data=Cuantitativas)
sns.scatterplot(x='porc_enganche', y='Predicciones4', color="red", data=Cuantitativas)
#sns.lineplot(x='alcohol', y='Predicciones', color="red", data=df)
```

```
#Corroboramos cual es el coeficiente de Determinación de nuestro modelo
coef_Deter4=model4.score(X=Vars_Indep3, y=Var_Dep3)
coef_Deter4
```

```
0.0019167133465273212
```

```
#Corroboramos cual es el coeficiente de Correlación de nuestro modelo
coef_Correl4=np.sqrt(coef_Deter4)
coef_Correl4
```

```
0.04378028490687699
```

“Costo total ”

```
#Imprimimos el scatter plot entre la variable dependiente (total) e independiente (alchool)
#para observar el comportamiento en su dispersión
from turtle import color
sns.scatterplot(x='monto_accesorios', y='costo_total', color="blue", data=Cuantitativas)
sns.scatterplot(x='enganche', y='costo_total', color="red", data=Cuantitativas)
```

```
#Declaramos las variables dependientes e independientes para la regresión lineal
#Vars_Indep= df[['alcohol', 'speeding']]
Vars_Indep4= Cuantitativas[['monto_accesorios']]
Var_Dep4= Cuantitativas['costo_total']
```

```
#Se define model como la función de regresión lineal
from sklearn.linear_model import LinearRegression
model5= LinearRegression()
```

```
#Ajustamos el modelo con las variables antes declaradas
model5.fit(X=Vars_Indep4, y=Var_Dep4)
```

```
#Verificamos los coeficientes obtenidos para el modelo ajustado
model5.__dict__
```

```
{'fit_intercept': True,
 'copy_X': True,
 'n_jobs': None,
 'positive': False,
 'feature_names_in_': array(['monto_accesorios'], dtype=object),
 'n_features_in_': 1,
 'coef_': array([2.5403712]),
 'rank_': 1,
 'singular_': array([8935.96326915]),
 'intercept_': 5665.489355660351}
```

Modelo matemático: y= 2.5403712x + 5665.489355660351

```
#Predecimos los valores de total de accidentes a partir de la variable "alcohol"
#y_pred= model.predict(X=df[['alcohol', 'speeding']])
y_pred5= model5.predict(X=Cuantitativas[['monto_accesorios']])
y_pred5
```

```
array([5665.48935566, 5665.48935566, 5665.48935566, ..., 5665.48935566,
       5665.48935566, 5665.48935566])
```

```
#Insertamos la columna de predicciones en el DataFrame
Cuantitativas.insert(0, 'Predicciones5', y_pred5)
Cuantitativas
```

```
#Visualizamos la gráfica comparativa entre el total real y el total predecido
sns.scatterplot(x='monto_accesorios', y='costo_total', color="blue", data=Cuantitativas)
sns.scatterplot(x='monto_accesorios', y='Predicciones5', color="red", data=Cuantitativas)
#sns.lineplot(x='alcohol', y='Predicciones', color="red", data=df)
```

```
#Corroboramos cual es el coeficiente de Determinación de nuestro modelo
coef_Deter5=model5.score(X=Vars_Indep4, y=Var_Dep4)
coef_Deter5
```

```
0.0039817409854288055
```

```
#Corroboramos cual es el coeficiente de Correlación de nuestro modelo
coef_Correl5=np.sqrt(coef_Deter5)
coef_Correl5
```

```
0.06310103791086803
```

“Monto_financiado”

```
#Imprimimos el scatter plot entre la variable dependiente (total) e independiente (alchool)
#para observar el comportamiento en su dispersión
from turtle import color
sns.scatterplot(x='inversion', y='monto_financiado', color="blue", data=Cuantitativas)
sns.scatterplot(x='porc_tasa', y='monto_financiado', color="red", data=Cuantitativas)
```

```
#Declaramos las variables dependientes e independientes para la regresión lineal
#Vars_Indep= df[['alcohol', 'speeding']]
Vars_Indep5= Cuantitativas[['porc_tasa']]
Var_Dep5= Cuantitativas['monto_financiado']
```

```
#Se define model como la función de regresión lineal
from sklearn.linear_model import LinearRegression
model6= LinearRegression()


#Ajustamos el modelo con las variables antes declaradas
model6.fit(X=Vars_Indep5, y=Var_Dep5)
```

```
#Verificamos los coeficientes obtenidos para el modelo ajustado
model6.__dict__
```

```
{'fit_intercept': True,
 'copy_X': True,
 'n_jobs': None,
 'positive': False,
 'feature_names_in_': array(['porc_tasa'], dtype=object),
 'n_features_in_': 1,
 'coef_': array([-5.3397735]),
 'rank_': 1,
 'singular_': array([1161.91123173]),
 'intercept_': 3084.211509627098}
```

Modelo matemático: y= -5.3397735x + 3084.211509627098

```
#Predecimos los valores de total de accidentes a partir de la variable "alcohol"
#y_pred= model.predict(X=df[['alcohol', 'speeding']])
y_pred6= model6.predict(X=Cuantitativas[['porc_tasa']])
y_pred6
```

```
array([3084.21150963, 3084.21150963, 3084.21150963, ..., 3080.2066795 ,
       3084.95907792, 3099.59005731])
```

```
#Insertamos la columna de predicciones en el DataFrame
Cuantitativas.insert(0, 'Predicciones6', y_pred6)
Cuantitativas
```

```
#Visualizamos la gráfica comparativa entre el total real y el total predecido

sns.scatterplot(x='porc_tasa', y='monto_financiado', color="blue", data=Cuantitativas)
sns.scatterplot(x='porc_tasa', y='Predicciones6', color="red", data=Cuantitativas)
#sns.lineplot(x='alcohol', y='Predicciones', color="red", data=df)
```

```
#Corroboramos cual es el coeficiente de Determinación de nuestro modelo
coef_Deter6=model6.score(X=Vars_Indep5, y=Var_Dep5)
coef_Deter6
```

```
0.0014077274944149787
```

```
#Corroboramos cual es el coeficiente de Correlación de nuestro modelo
coef_Correl6=np.sqrt(coef_Deter6)
coef_Correl6
```

```
0.03751969475375538
```

“Pagos_realizados”

```
#Imprimimos el scatter plot entre la variable dependiente (total) e independiente (alchool)
#para observar el comportamiento en su dispersión
from turtle import color
sns.scatterplot(x='semana', y='pagos_realizados', color="blue", data=Cuantitativas)
sns.scatterplot(x='status', y='pagos_realizados', color="red", data=Cuantitativas)
```

```
#Declaramos las variables dependientes e independientes para la regresión lineal
#Vars_Indep= df[['alcohol', 'speeding']]
Vars_Indep6= Cuantitativas[['semana']]
Var_Dep6= Cuantitativas['pagos_realizados']


#Se define model como la función de regresión lineal
from sklearn.linear_model import LinearRegression
model7= LinearRegression()


#Ajustamos el modelo con las variables antes declaradas
model7.fit(X=Vars_Indep6, y=Var_Dep6)
```

```
#Verificamos los coeficientes obtenidos para el modelo ajustado
model7.__dict__
```

```
{'fit_intercept': True,
 'copy_X': True,
 'n_jobs': None,
 'positive': False,
 'feature_names_in_': array(['semana'], dtype=object),
 'n_features_in_': 1,
 'coef_': array([-0.01281814]),
 'rank_': 1,
 'singular_': array([15255.71836322]),
 'intercept_': 13.228256219508708}
```

Modelo matemático: y= -0.01281814x + 13.228256219508708

```
#Predecimos los valores de total de accidentes a partir de la variable "alcohol"
#y_pred= model.predict(X=df[['alcohol', 'speeding']])
y_pred7= model7.predict(X=Cuantitativas[['semana']])
y_pred7
```

```
array([12.18998726, 10.34417579, 11.04917323, ...,  9.9468136 ,
        8.28045602,  9.39563378])
```

```
#Insertamos la columna de predicciones en el DataFrame
Cuantitativas.insert(0, 'Predicciones7', y_pred7)
Cuantitativas
```

```
#Visualizamos la gráfica comparativa entre el total real y el total predecido

sns.scatterplot(x='semana', y='pagos_realizados', color="blue", data=Cuantitativas)
sns.scatterplot(x='semana', y='Predicciones7', color="red", data=Cuantitativas)
#sns.lineplot(x='alcohol', y='Predicciones', color="red", data=df)
```

```
#Corroboramos cual es el coeficiente de Determinación de nuestro modelo
coef_Deter7=model7.score(X=Vars_Indep6, y=Var_Dep6)
coef_Deter7
```

```
0.017845933234437727
```

```
#Corroboramos cual es el coeficiente de Correlación de nuestro modelo
coef_Correl7=np.sqrt(coef_Deter7)
coef_Correl7
```

```
0.1335886718043028
```

“Porcentaje_enganche”

```
#Imprimimos el scatter plot entre la variable dependiente (total) e independiente (alchool)
#para observar el comportamiento en su dispersión
from turtle import color
sns.scatterplot(x='score_buro', y='porc_enganche', color="blue", data=Cuantitativas)
sns.scatterplot(x='status', y='porc_enganche', color="red", data=Cuantitativas)
```

```
#Declaramos las variables dependientes e independientes para la regresión lineal
#Vars_Indep= df[['alcohol', 'speeding']]
Vars_Indep7= Cuantitativas[['score_buro']]
Var_Dep7= Cuantitativas['porc_enganche']
```

```
#Se define model como la función de regresión lineal
from sklearn.linear_model import LinearRegression
model8= LinearRegression()
```

```
#Ajustamos el modelo con las variables antes declaradas
model8.fit(X=Vars_Indep7, y=Var_Dep7)
```

```
#Verificamos los coeficientes obtenidos para el modelo ajustado
model8.__dict__
```

```
{'fit_intercept': True,
 'copy_X': True,
 'n_jobs': None,
 'positive': False,
 'feature_names_in_': array(['score_buro'], dtype=object),
 'n_features_in_': 1,
 'coef_': array([-0.07932367]),
```

```
'rank_': 1,
'singular_': array([713.01781912]),
'intercept_': 0.8802345802939936}
```

Modelo matemático: y= -0.07932367x + 0.8802345802939936

```
#Predecimos los valores de total de accidentes a partir de la variable "alcohol"
#y_pred= model.predict(X=df[['alcohol', 'speeding']])
y_pred8= model8.predict(X=Cuantitativas[['score_buro']])
y_pred8
```

```
array([0.88023458, 0.88023458, 0.88023458, ..., 0.88023458, 0.88023458,
       0.88023458])
```

```
#Insertamos la columna de predicciones en el DataFrame
Cuantitativas.insert(0, 'Predicciones8', y_pred8)
Cuantitativas
```

```
#Visualizamos la gráfica comparativa entre el total real y el total predecido

sns.scatterplot(x='semana', y='pagos_realizados', color="blue", data=Cuantitativas)
sns.scatterplot(x='semana', y='Predicciones7', color="red", data=Cuantitativas)
#sns.lineplot(x='alcohol', y='Predicciones', color="red", data=df)
```

```
#Corroboramos cual es el coeficiente de Determinación de nuestro modelo
coef_Deter8=model8.score(X=Vars_Indep7, y=Var_Dep7)
coef_Deter8
```

```
0.004263180903821717
```

```
#Corroboramos cual es el coeficiente de Correlación de nuestro modelo
coef_Correl8=np.sqrt(coef_Deter8)
coef_Correl8
```

```
0.06529303870874534
```