IMF Business School

Hoja de respuestas

Módulo	Aprendizaje Automático
Nombre y apellidos	Richard Douglas Grijalba
Fecha entrega	22 Abril 2024

Caso Practico

Trabajamos como científicos de datos para una empresa de retail que, debido al cambio en los hábitos de consumo de los clientes, está potenciando ampliamente el servicio de venta online. La empresa quiere realizar un modelo de aprendizaje automático para clasificar a los clientes en función de la probabilidad de generar ingresos al comprar en la web.

El objetivo es realizar una serie de acciones específicas para los clientes que es más probable que hagan compras en la web.

Información de los datos

1. El conjunto de datos consta de diez atributos numéricos y ocho categóricos.
2. El atributo **revenue** puede ser usado como la etiqueta de la clase.
3. **Administrative** , **administrative duration** , **informational**, **informational duration** , **product related** y **product related duration** representan el número de diferentes tipos de páginas visitadas por el visitante en esa sesión y el tiempo total dedicado a cada una de estas categorías de páginas. Los valores de estas características se derivan de la información del URL de las páginas visitadas por el usuario y se actualizan en tiempo real cuando el usuario realiza una acción, por ejemplo, pasar de una página a otra.

4. Las características **bounce rate** , **exit rate y page value** representan las métricas medidas por Google Analytics para cada página del sitio de comercio electrónico.

   El valor de la característica bounce rate de una página web se refiere al porcentaje de visitantes que entran en el sitio desde esa página y luego salen (rebote) sin activar ninguna otra solicitud al servidor de análisis durante esa sesión. El valor de la característica exit rate para una página web específica se calcula como para todas las visitas a la página, el porcentaje que fueron las últimas en la sesión. La función Page Value representa el valor medio de una página web que un usuario ha visitado antes de completar una transacción de comercio electrónico.

5. La característica de **special day** indica la cercanía de la hora de visita del sitio a un día especial específico (por ejemplo, el Día de la Madre, San Valentín) en el que es más probable que las sesiones finalicen con una transacción.
   El valor de este atributo se determina teniendo en cuenta la dinámica del comercio electrónico, como la duración entre la fecha del pedido y la fecha de entrega.
   Por ejemplo, para San Valentín, este valor toma un valor distinto de 0 entre el 2 y el 12 de febrero, 0 antes y después de esta fecha a menos que esté cerca de otro día especial, y su valor máximo de 1 el 8 de febrero.

6. El conjunto de datos también incluye el sistema operativo, el navegador, la región, el tipo de tráfico, el tipo de visitante como visitante que regresa o como nuevo visitante, un valor booleano que indica si la fecha de la visita es de fin de semana, y el mes del año.

Se pide : Tareas que nos pide la empresa

1. Realización de un análisis de las variables del dataset de Google Analytics como pueden ser histogramas, boxplots, etc. Cualquier otro análisis es bienvenido, siempre explicándolo y con un sentido de negocio.

2. Tratamiento de los valores faltantes, outliers, etc., en caso de que los hubiese. Si hay valores con missings, habrá que eliminarlos con el método de Pandas llamado Dropna().a.
3. Tratamiento de categóricas, pasándolas a numéricas por medio de dummies, mapeándolas o utilizando un label encoder. Hay que justificar las operaciones que se realizan.
4. Si existe alguna variable que se necesite borrar, habrá que borrara y justificarlo.
5. Estandariza los datos.
6. Dividir los datos en train y en test. Con los datos de train se pretende ajustar modelos con CrossValidation y GridSearch.

- Utilizar un modelo lineal. Entre los modelos lineales están las regresiones logísticas, las regresiones lineales, etc.
- Utilizar un modelo de redes neuronales.

- Utilizar cualquier otro modelo de clasificación.

7. Optimizar algún parámetro de cada modelo utilizando CrossValidation y GridSearch, o de la forma que se estime oportuna, siempre justificándolo.
8. Elegir el mejor modelo de los tres según la métrica ROC en CrossValidation. Predecir Test y obtener una métrica estimada.
9. Umbralizar las probabilidades utilizando el umbral que maximice el área bajo la curva ROC.
10. El entregable final será un Jupyter Notebook en el que se realicen todos los análisis y los modelos.

Carga de las librerias

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
import sklearn.metrics as metrics
from IPython.core.display import display, HTML
from sklearn.metrics import accuracy_score
from pandas.plotting import scatter_matrix
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

```
C:\Users\Rdouglas\AppData\Local\Temp\ipykernel_8500\306533062.py:9: DeprecationWarning: Importing display from IPython.core.dis
play is deprecated since IPython 7.14, please import from IPython display
  from IPython.core.display import display, HTML
```

Definición de funciones: Esta funcion fue proporcionada durante el desarrollo de uno de los casos practicos, la cual es muy util para facilitar la generacion de los graficos

```
def relaciones_vs_target(X, Y, return_type='axes'):
    '''
    Función que representa gráficos de dispersión de las variables
    en X en función a la variable Y
    '''
    fig_tot = (len(X.columns))
    fig_por_fila = 5
    tamanio_fig = 4
    num_filas = int( np.ceil(fig_tot/fig_por_fila) )
    plt.figure( figsize=( fig_por_fila*tamanio_fig+5, num_filas*tamanio_fig+5 ) )
    c = 0
    for i, col in enumerate(X.columns):
        plt.subplot(num_filas, fig_por_fila, i+1)
        sns.scatterplot(x=X[col], y=Y)
        plt.title( '%s vs %s' % (col, 'target') )
        plt.ylabel('Target')
        plt.xlabel(col)
    plt.show()

def represento_doble_hist(x_1, x_0, n_bins=11, title='', label_1='Clase 1',
                          label_0='Clase 0', density=0):
    '''
    Función que recibe dos distribuciones de probabilidad y las representa
    en el mismo gráfico
    '''
    bins = n_bins
    plt.hist(x_1, bins, density = density, alpha=0.5, label=label_1, color='red')
    plt.hist(x_0, bins, density = density, alpha=0.5, label=label_0, color='green')
    plt.title(title)
    plt.legend(loc='best')

def hist_pos_neg_feat(x, y, density=0, nbins=11, targets=(0,1)):
    '''
    Representa las variables en x divididas en dos distribuciones
    según su valor de y sea 1 o 0
    '''
    fig_tot = len(x.columns)
    fig_tot_fila = 4; fig_tamanio = 4
    num_filas = int( np.ceil(fig_tot/fig_tot_fila) )
    plt.figure( figsize=( fig_tot_fila*fig_tamanio+2, num_filas*fig_tamanio+2 ) )
    target_neg, target_pos = targets
    for i, feat in enumerate(x.columns):
        plt.subplot(num_filas, fig_tot_fila, i+1);
        plt.title('%s' % feat)
        idx_pos = y == target_pos
        idx_neg= y == target_neg
        represento_doble_hist(x[feat][idx_pos].values, x[feat][idx_neg].values, nbins,
                  density = density, title=('%s' % feat))
```

Exploracion inicial de los datos

[4]:
```
#vamos a mostrar como se ven los datos en el dataset

df.head()
```

t[4]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues
0	0.0	0.0	0.0	0.0	1.0	0.000000	0.20	0.20	0.0
1	0.0	0.0	0.0	0.0	2.0	64.000000	0.00	0.10	0.0
2	0.0	-1.0	0.0	-1.0	1.0	-1.000000	0.20	0.20	0.0
3	0.0	0.0	0.0	0.0	2.0	2.666667	0.05	0.14	0.0
4	0.0	0.0	0.0	0.0	10.0	627.500000	0.02	0.05	0.0

[5]:
```
print(u'- El número de filas en el dataset es: {}'.format(df.shape[0]))
print(u'- El número de columnas en el dataset es: {}'.format(df.shape[1]))
print(u'- El tamaño dataset es: {}'.format(df.size))
```

```
- El número de filas en el dataset es: 12330
- El número de columnas en el dataset es: 18
- El tamaño dataset es: 221940
```

[6]:
```
print(u'- Los nombres de las variables son: {}'.format(list(df.columns)))
```

```
- Los nombres de las variables son: ['Administrative', 'Administrative_Duration', 'Informational', 'Informational_Duration', 'P
roductRelated', 'ProductRelated_Duration', 'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Month', 'OperatingSystems',
'Browser', 'Region', 'TrafficType', 'VisitorType', 'Weekend', 'Revenue']
```

Caracteristicas del Dataset Descripcion general, informacion que describe los datos

Caracteristicas del Dataset Descripcion general, informacion que describe los datos

El conjunto de datos contiene las siguientes caracteristicas:

- Administrative: El número de páginas administrativas visitadas por el usuario.
- Administrative Duration: La duración total de tiempo (en segundos) que el usuario pasó en páginas administrativas.
- Informational: El número de páginas informativas visitadas por el usuario.
- Informational Duration: La duración total de tiempo (en segundos) que el usuario pasó en páginas informativas.
- Product Related: El número de páginas relacionadas con productos visitadas por el usuario.
- Product Related Duration: La duración total de tiempo (en segundos) que el usuario pasó en páginas relacionadas con productos.
- Bounce Rate: El porcentaje de visitas de una sola página (visitantes que salieron inmediatamente).
- Exit Rate: El porcentaje de visitas que terminaron en esa página.
- Page Value: El valor promedio de una página vista antes de que el usuario abandone el sitio.
- Special Day: Indica la proximidad del tiempo de visita del sitio web a un día especial (por ejemplo, el Día de San Valentín, la Navidad).
- Month: El mes de la visita.
- Operating System: El sistema operativo del usuario.
- Browser: El navegador utilizado por el usuario.
- Region: La región del usuario.
- Traffic Type: El tipo de tráfico (por ejemplo, tráfico directo, motor de búsqueda).
- Visitor Type: El tipo de visitante (nuevo, recurrente).
- Weekend: Indica si la visita ocurrió en un fin de semana o no.
- Revenue: La variable objetivo que indica si el usuario realizó una compra o no (True/False).

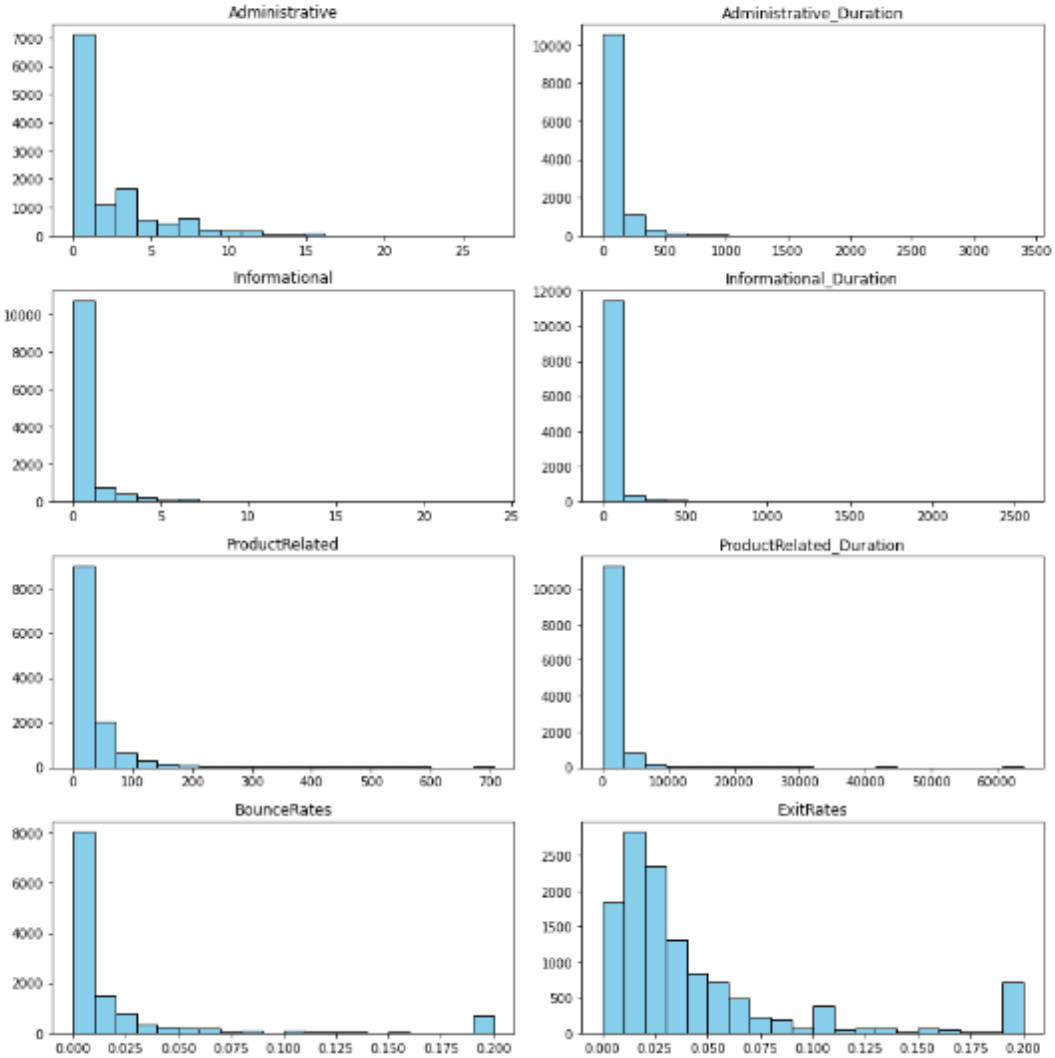
Procede a Graficar los Histogramas

[11]:
```
# Crear una figura y ejes para los 8 histogramas
fig, axs = plt.subplots(4, 2, figsize=(12, 12))

# Nombres de las columnas
columnas = ['Administrative', 'Administrative_Duration', 'Informational', 'Informational_Duration',
            'ProductRelated', 'ProductRelated_Duration', 'BounceRates', 'ExitRates']

# Iterar sobre cada columna y crear un histograma en el eje correspondiente
for i, columna in enumerate(columnas):
    fila = i // 2
    columna_grafico = i % 2
    axs[fila, columna_grafico].hist(df[columna], bins=20, color='skyblue', edgecolor='black')
    axs[fila, columna_grafico].set_title(columna)

# Ajustar el diseño de los subgráficos
plt.tight_layout()
plt.show()
```



Categóricas a numéricas:

[15]:
```
df.select_dtypes(exclude=['number']).columns
```

t[15]:
```
Index(['Month', 'VisitorType', 'Weekend', 'Revenue'], dtype='object')
```

[16]:
```
# realizar los cambios o transformaciones de las variables
valores_mes = df['Month'].unique()
valores_tipo_visita = df['VisitorType'].unique()
valores_revenue = df['Revenue'].unique()
valores_fin_de_semana = df['Weekend'].unique()

# Imprimir los valores únicos de cada columna
print("Valores únicos de la columna 'Month':", valores_mes)
print("Valores únicos de la columna 'VisitorType':", valores_tipo_visita)
print("Valores únicos de la columna 'Revenue':", valores_revenue)
print("Valores únicos de la columna 'Weekend':", valores_fin_de_semana)
```

```
Valores únicos de la columna 'Month': ['Feb' 'Mar' 'May' 'Oct' 'June' 'Jul' 'Aug' 'Nov' 'Sep' 'Dec']
Valores únicos de la columna 'VisitorType': ['Returning_Visitor' 'New_Visitor' 'Other']
Valores únicos de la columna 'Revenue': [False  True]
Valores únicos de la columna 'Weekend': [False  True]
```

[17]:
```
from sklearn.preprocessing import LabelEncoder
```

[18]:
```
 # se llama el encoder
lb_encoder = LabelEncoder()
```

[19]:
```
#antes de realizar esta transformacion se va a realizar una copia del dataset en adelante el codificado sera df1
df1 = df.copy()
```

[20]:
```
df1.head(3)
```

t[20]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues
0	0.0	0.0	0.0	0.0	1.0	0.0	0.2	0.2	0.0
1	0.0	0.0	0.0	0.0	2.0	64.0	0.0	0.1	0.0
2	0.0	-1.0	0.0	-1.0	1.0	-1.0	0.2	0.2	0.0

Dividir el Dataset entre las caracteristicas X, vs la caracteristica Target

n [26]:
```
X = df1.drop('Revenue', axis=1)
Y = df1['Revenue']
```

Obersevar el contenido de la variable Objetivo

n [27]:
```
Y.value_counts()  # 0 false   1 true
```

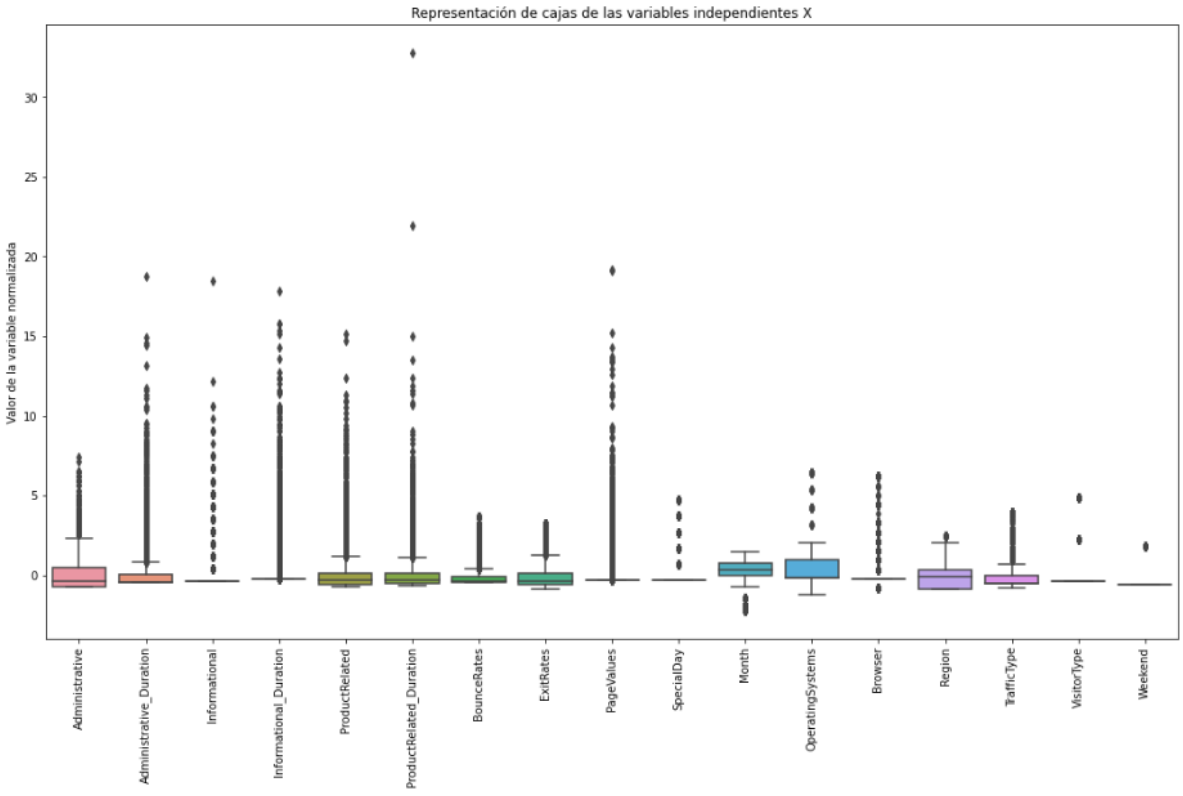
ut[27]:
```
0    10422
1     1908
Name: Revenue, dtype: int64
```

In [29]:
```
X.describe().T
```

Out[29]:

	count	mean	std	min	25%	50%	75%	max
Administrative	12330.0	2.316302	3.321163	0.0	0.000000	1.000000	4.000000	27.000000
Administrative_Duration	12330.0	80.823396	176.777041	-1.0	0.000000	8.000000	93.256250	3398.750000
Informational	12330.0	0.503406	1.270093	0.0	0.000000	0.000000	0.000000	24.000000
Informational_Duration	12330.0	34.467207	140.750298	-1.0	0.000000	0.000000	0.000000	2549.375000
ProductRelated	12330.0	31.748256	44.467488	0.0	7.000000	18.000000	38.000000	705.000000
ProductRelated_Duration	12330.0	1195.360026	1913.390668	-1.0	185.287500	599.766190	1464.157214	63973.522230
BounceRates	12330.0	0.022131	0.048404	0.0	0.000000	0.003119	0.016667	0.200000
ExitRates	12330.0	0.042982	0.048503	0.0	0.014286	0.025124	0.050000	0.200000
PageValues	12330.0	5.889258	18.568437	0.0	0.000000	0.000000	0.000000	361.763742
SpecialDay	12330.0	0.061427	0.198917	0.0	0.000000	0.000000	0.000000	1.000000
Month	12330.0	6.973885	2.719474	1.0	7.000000	8.000000	9.000000	11.000000
OperatingSystems	12330.0	2.124006	0.911325	1.0	2.000000	2.000000	3.000000	8.000000
Browser	12330.0	2.357097	1.717277	1.0	2.000000	2.000000	2.000000	13.000000
Region	12330.0	3.147364	2.401591	1.0	1.000000	3.000000	4.000000	9.000000
TrafficType	12330.0	4.069586	4.025169	1.0	2.000000	2.000000	4.000000	20.000000
VisitorType	12330.0	0.151176	0.376989	0.0	0.000000	0.000000	0.000000	2.000000
Weekend	12330.0	0.232603	0.422509	0.0	0.000000	0.000000	0.000000	1.000000

[34]:
```
#Se generan los graficos de caja/bigotes  Boxplots para ver los datos
plt.figure(figsize=(18,10))
ax = sns.boxplot(data=X_normalizado)
ax.set_xticklabels(ax.get_xticklabels(),rotation=90)
plt.title(u'Representación de cajas de las variables independientes X')
plt.ylabel('Valor de la variable normalizada')
_ = plt.xlabel('Nombre de la variable')
```



```
hist_pos_neg_feat(X,Y)
```

Variables con mayor correlación con la target:

Este paso es importante en vista que las variables con alta o mediana correlación tanto postiva como negativa, con la variable objetivo (target) a menudo se utilizan como características importantes en el modelo, esto para utilizar aquellas que mejor apoyen al proceso de calculo, ya que pueden tener un mayor impacto en la capacidad del modelo para hacer predicciones precisas y mejores resultados.

```
correlaciones_target = matriz_correlaciones.values[ -1, : -1]
indices_inversos =  abs(correlaciones_target[ : ]).argsort()[ : : -1]
diccionario = {}
for nombre, correlacion in zip( X.columns[indices_inversos], list(correlaciones_target[indices_inversos] ) ):
    diccionario[nombre] = correlacion
pd.DataFrame.from_dict(diccionario, orient='index', columns=['Correlación con la target'])
```

	Correlación con la target
PageValues	0.492569
ExitRates	-0.206669
ProductRelated	0.158405
ProductRelated_Duration	0.152258
BounceRates	-0.150401
Administrative	0.138797
VisitorType	0.098485
Informational	0.095260
Administrative_Duration	0.093576
SpecialDay	-0.082305
Month	0.075441
Informational_Duration	0.070360
Weekend	0.029295
Browser	0.023984
OperatingSystems	-0.014668
Region	-0.011595
TrafficType	-0.005113

Basado en la tabla anterior

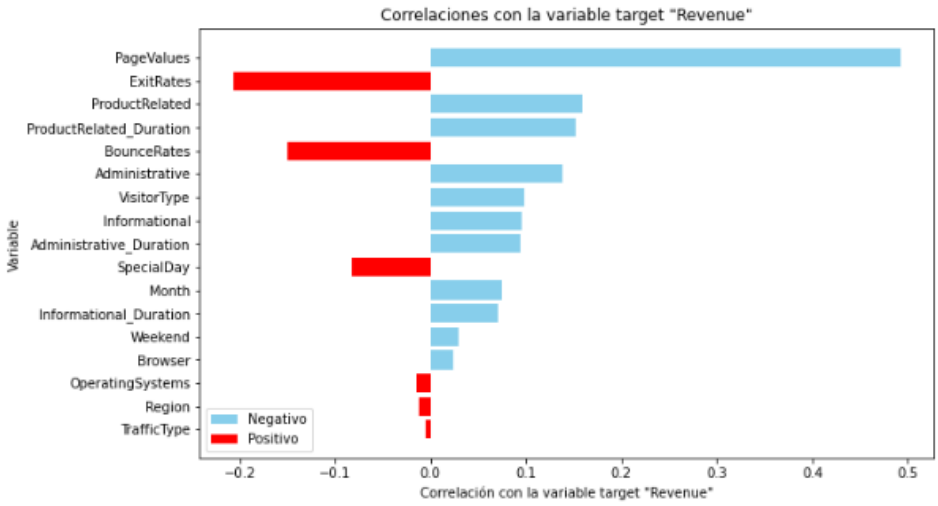
- **Variables con baja correlación absoluta:** Variables como 'Weekend', 'Browser', 'OperatingSystems', 'Region' y 'TrafficType' tienen correlaciones cercanas a cero con la variable objetivo.
- **Variables con correlación negativa moderada:** 'ExitRates', 'BounceRates' y 'SpecialDay' tienen correlaciones negativas con la variable objetivo.
- **Variables con correlación negativa moderada:** 'ExitRates', 'BounceRates' y 'SpecialDay' tienen correlaciones negativas con la variable objetivo.

```
# Definir los datos para el gráfico de barras
nombres_columnas = list(diccionario.keys())
correlaciones = list(diccionario.values())

# Crear el gráfico de barras
plt.figure(figsize=(10, 6))
bars = plt.barh(nombres_columnas, correlaciones, color=['red' if c < 0 else 'skyblue' for c in correlaciones])
plt.xlabel('Correlación con la variable target "Revenue"')
plt.ylabel('Variable')
plt.title('Correlaciones con la variable target "Revenue"')
plt.gca().invert_yaxis()  # Invertir el eje y para mostrar la variable más correlacionada en la parte superior

# Añadir una leyenda para las barras de color rojo y azul
plt.legend(handles=[bars[0], bars[-1]], labels=['Negativo', 'Positivo'])

plt.show()
```

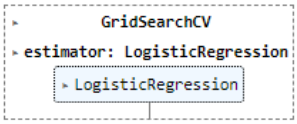


Model 1 : Aplicamos un modelo de regresión logística / Aplicamos un modelo de clasificación lineal

In [45]:
```
modelo = LogisticRegression()
parametros = {"C": [0., 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08,0.09],
              "class_weight":['balanced', None]}
```

In [46]:
```
modelo_gs = GridSearchCV(modelo, param_grid=parametros,
                         cv = 5, scoring='roc_auc')
modelo_gs.fit(X_train, Y_train)
```

Out[46]:

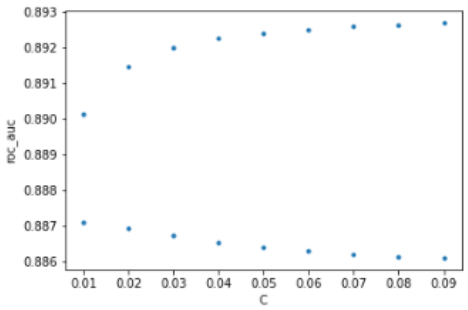


In [47]:
```
print(modelo_gs.best_params_, "\nROC AUC: {}".format(round(modelo_gs.best_score_,2)))
```

```
{'C': 0.09, 'class_weight': 'balanced'}
ROC AUC: 0.89
```

In [48]:
```
df_search = pd.DataFrame.from_dict(modelo_gs.cv_results_)
```

In [49]:
```
plt.xlabel('C')
plt.ylabel('roc_auc')
_ = plt.plot( df_search['param_C'], df_search['mean_test_score'], '.')
```



]:
```
umbral = 0.6
y_umbralizadas = 1*(y_test_pred_prob[:, 1] > umbral)
```

]:
```
print(u"Matriz de confusión\n", metrics.confusion_matrix(Y_test, y_umbralizadas))
print("\nAccuracy\t{}".format(round(metrics.accuracy_score(Y_test, y_umbralizadas),2)))
print("Sensitividad\t{}".format(round(metrics.recall_score(Y_test, y_umbralizadas),2)))
print(u"Precisión\t{}".format(round(metrics.precision_score(Y_test, y_umbralizadas),2)))
```

```
Matriz de confusión
 [[1898  146]
 [ 151  271]]

Accuracy        0.88
Sensitividad    0.64
Precisión       0.65
```

Modelo #2 Aplicación de un modelo de redes neuronales

```
modelo = MLPClassifier()
parametros = {'solver': ['lbfgs'],
              'max_iter': [50,100,150,200], # Iteraciones máximas en cada red
              'alpha': 10.0 ** -np.arange(1, 10), # Parámetro de regularización L2 para evitar sobreajuste
              'hidden_layer_sizes':np.arange(30, 35), # Número de neuronas en cada capa
              'random_state':[0]}
```

```
modelo_gs = GridSearchCV(modelo, param_grid=parametros, cv = 3,
                         scoring='roc_auc', n_jobs=-1, verbose=10)
modelo_gs.fit(X_train, Y_train)
```

```
Fitting 3 folds for each of 180 candidates, totalling 540 fits
```

GridSearchCV

```
GridSearchCV(cv=3, estimator=MLPClassifier(), n_jobs=-1,
             param_grid={'alpha': array([1.e-01, 1.e-02, 1.e-03, 1.e-04, 1.e-05, 1.e-06, 1.e-07, 1.e-08,
       1.e-09]),
                         'hidden_layer_sizes': array([30, 31, 32, 33, 34]),
                         'max_iter': [50, 100, 150, 200], 'random_state': [0],
                         'solver': ['lbfgs']},
             scoring='roc_auc', verbose=10)
```

estimator: MLPClassifier

MLPClassifier()

Umbralizo las predicciones:

In [65]:
```
umbral = 0.5
y_umbralizadas = 1*(y_test_pred_prob[:, 1] > umbral)
```

In [66]:
```
print(u"Matriz de confusión\n", metrics.confusion_matrix(Y_test, y_umbralizadas))
print("\nAccuracy\t{}".format(round(metrics.accuracy_score(Y_test, y_umbralizadas),2)))
print("Sensitividad\t{}".format(round(metrics.recall_score(Y_test, y_umbralizadas),2)))
print(u"Precisión\t{}".format(round(metrics.precision_score(Y_test, y_umbralizadas),2)))
```

```
Matriz de confusión
 [[1958   86]
 [ 183  239]]

Accuracy        0.89
Sensitividad    0.57
Precisión       0.74
```

Modelo #3 Árboles de decisión

:
```
df1.columns
```

:
```
Index(['Administrative', 'Administrative_Duration', 'Informational',
       'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration',
       'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Month',
       'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType',
       'Weekend', 'Revenue'],
      dtype='object')
```

[73]:
```
#entrenamiento del modelo
clf.fit(X_train,y_train)
```

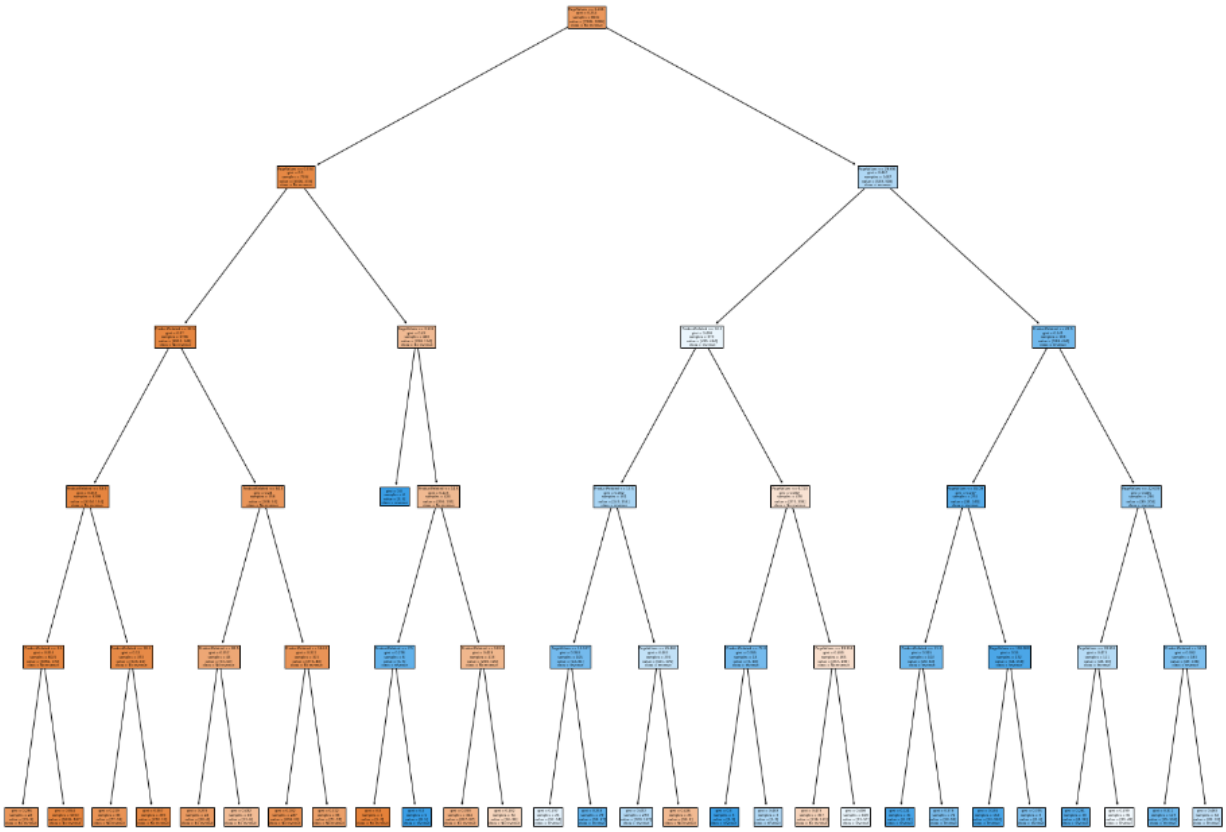
t[73]:
```
DecisionTreeClassifier(max_depth=5, random_state=0)
```

[74]:
```
from sklearn import tree   # este codigo tomado de una asignacion de arboles de decision Richard Douglas G.
text_representation = tree.export_text(clf)
print(text_representation)
```

```
|--- feature_0 <= 6.46
|   |--- feature_0 <= 0.03
|   |   |--- feature_1 <= 80.50
|   |   |   |--- feature_1 <= 53.50
|   |   |   |   |--- feature_1 <= 0.50
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature_1 >  0.50
|   |   |   |   |   |--- class: 0
|   |   |   |--- feature_1 >  53.50
|   |   |   |   |--- feature_1 <= 58.50
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature_1 >  58.50
|   |   |   |   |   |--- class: 0
|   |   |--- feature_1 >  80.50
|   |   |   |--- feature_1 <= 84.50
|   |   |   |   |--- feature_1 <= 82.50
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature_1 >  82.50
|   |   |   |   |   |--- class: 0
|   |   |   |--- feature_1 >  84.50
|   |   |   |   |--- feature_1 <= 144.00
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature_1 >  144.00
|   |   |   |   |   |--- class: 0
|   |--- feature_0 >  0.03
|   |   |--- feature_0 <= 0.14
|   |   |   |--- class: 1
|   |   |--- feature_0 >  0.14
|   |   |   |--- feature_1 <= 14.50
|   |   |   |   |--- feature_1 <= 7.50
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature_1 >  7.50
|   |   |   |   |   |--- class: 1
|   |   |   |--- feature_1 >  14.50
|   |   |   |   |--- feature_1 <= 128.50
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature_1 >  128.50
|   |   |   |   |   |--- class: 0
|--- feature_0 >  6.46
|   |--- feature_0 <= 29.89
|   |   |--- feature_1 <= 32.50
|   |   |   |--- feature_1 <= 15.50
|   |   |   |   |--- feature_0 <= 13.54
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_0 >  13.54
|   |   |   |   |   |--- class: 1
|   |   |   |--- feature_1 >  15.50
|   |   |   |   |--- feature_0 <= 25.46
```

```
import matplotlib.pyplot as plt  # este codigo ejemplo tomado de una asignacion realizada sobre arboles decision
fig = plt.figure(figsize=(25,20))
feature=['PageValues','ProductRelated']
target=["No revenue","revenue"]

arbol= tree.plot_tree(clf, feature_names=feature,class_names=target,filled=True)
```



```
print(u"Matriz de confusión\n", metrics.confusion_matrix(y_test, y_pred))  # se presenta los resultados del arbol decision
print("\nAccuracy\t{}".format(round(metrics.accuracy_score(y_test, y_pred),2)))
print("Sensitividad\t{}".format(round(metrics.recall_score(y_test, y_pred),2)))
print(u"Precisión\t{}".format(round(metrics.precision_score(y_test, y_pred),2)))
```

```
Matriz de confusión
 [[2918  159]
 [ 284  338]]

Accuracy        0.88
Sensitividad    0.54
Precisión       0.68
```

Resultados

- **Variables con baja correlación absoluta:** Variables como 'Weekend', 'Browser', 'OperatingSystems', 'Region' y 'TrafficType' tienen correlaciones cercanas a cero con la variable objetivo.
- **Variables con correlación negativa moderada:** 'ExitRates', 'BounceRates' y 'SpecialDay' tienen correlaciones negativas con la variable objetivo.
- **Variables con correlación negativa moderada:** 'ExitRates', 'BounceRates' y 'SpecialDay' tienen correlaciones negativas con la variable objetivo.

Accuracy de Modelos

- Modelo 1 - **Model 1 : Aplicamos un modelo de regresión logística**
    - Accuracy 0.88
    - Sensitividad 0.64
    - Precisión 0.65
- Modelo 2 - **Modelo 2 Aplicación de un modelo de redes Neuronales**
    - Accuracy 0.89
    - Sensitividad 0.57
    - Precisión 0.74
- Modelo 3 - **Modelo 3 Árboles de decisión**
    - Accuracy 0.88
    - Sensitividad 0.54
    - Precisión 0.68

Cual modelo elegir? La decision que todo Analista, Datascientist, Empresa debe enfrentar.

La decisión de cual modelo usar o elegir no solo se debe basar en los resultados de accuracy, sino también las posibilidades económicas de una empresa, puede que un modelo brinde el mejor accurasy pero su aplicación en términos de costo sea imposible, po lo que se debe buscar un balance entre el costo/beneficio, en lo que respecta al uso o eleccion de un modelo en si, en mi preferencia me quedo con la regresion logistica, por un tema de aplicacion, costo-beneficio, y facil interpretacion, el mismo puede dar un resultado que se acerque a lo que una empresa busca y con dicho resultado decidir que otro modelo puede aplicar segun la disponibilida de los recursos, esto debido a que en función de costo/beneficio, cuando pensamos en la regresión logística esta nos resenta una opción más eficiente en términos de recursos computacionales (costo) y tiempo de entrenamiento (facilidad), por otra parte los árboles de decisión son un buen equilibrio entre resultados y rendimiento. Finalmente tenemos las redes neuronales (NeuralNetwork) las cuales pueden ofrecer el mejor rendimiento en términos de precisión, pero requieren más recursos y esfuerzo para su implementación y ajuste.

IMF Master en Business Analytics & Data Science Estudiante: **Richard Douglas Grijalba. modalidad virtual**. Costa Rica, pura vida.

Asignacion realizada utilizando codigo ejemplo y material suministrado por IMF, asi tambien investigacion propia sobre casos y aplicaciones de modelos de clasificacion, asignaciones previas o cursos introductorios a Modelos de Clasificacion tomados por el estudiante.

El datset suministrado por IMF, funciones casos y ejemplos vistos en el curso de Aprendizajes Automatico Modulo VI.