Módulo 2 - Práctica 2

Descubrimientos

El objetivo de esta práctica era el poder realizar un modelo de clasificación, para ello, al momento de poder leer los datos, nos pudimos dar cuenta que la variable *email* no nos funcionaría como unidad muestral, en su lugar, la variable a tomar sería el télefono de cada reservación dado que eran valores únicos.

```
In [9]: #We will be using uses' phone number as um
        df.shape, len(df.email.unique()), len(df["phone-number"].unique())
Out[9]: ((119390, 36), 115889, 119390)
```

Además, dado que las variables *name*, *email*, *credit_card* y *reservation_status_date* no ayudaría al modelo dada su naturaleza, las eliminamos. Para la selección de variables continuas y discretas, terminamos con un total de: 12 variables continuas y 18 variables discretas.

Una vez seleccionadas las variables discretas, podemos determinar que no nos sería de mucha ayuda crear variables dummies dado que son varias;por lo que se decidió crear una transformación WoE a los datos.

Variables Discretas

Para este proceso, completamos los faltantes por “Without Category”; pero, al revisar las frecuencias, nos percatamos que habían datos con valores “Undefined”, a lo que procedimos a sustituirlos también.

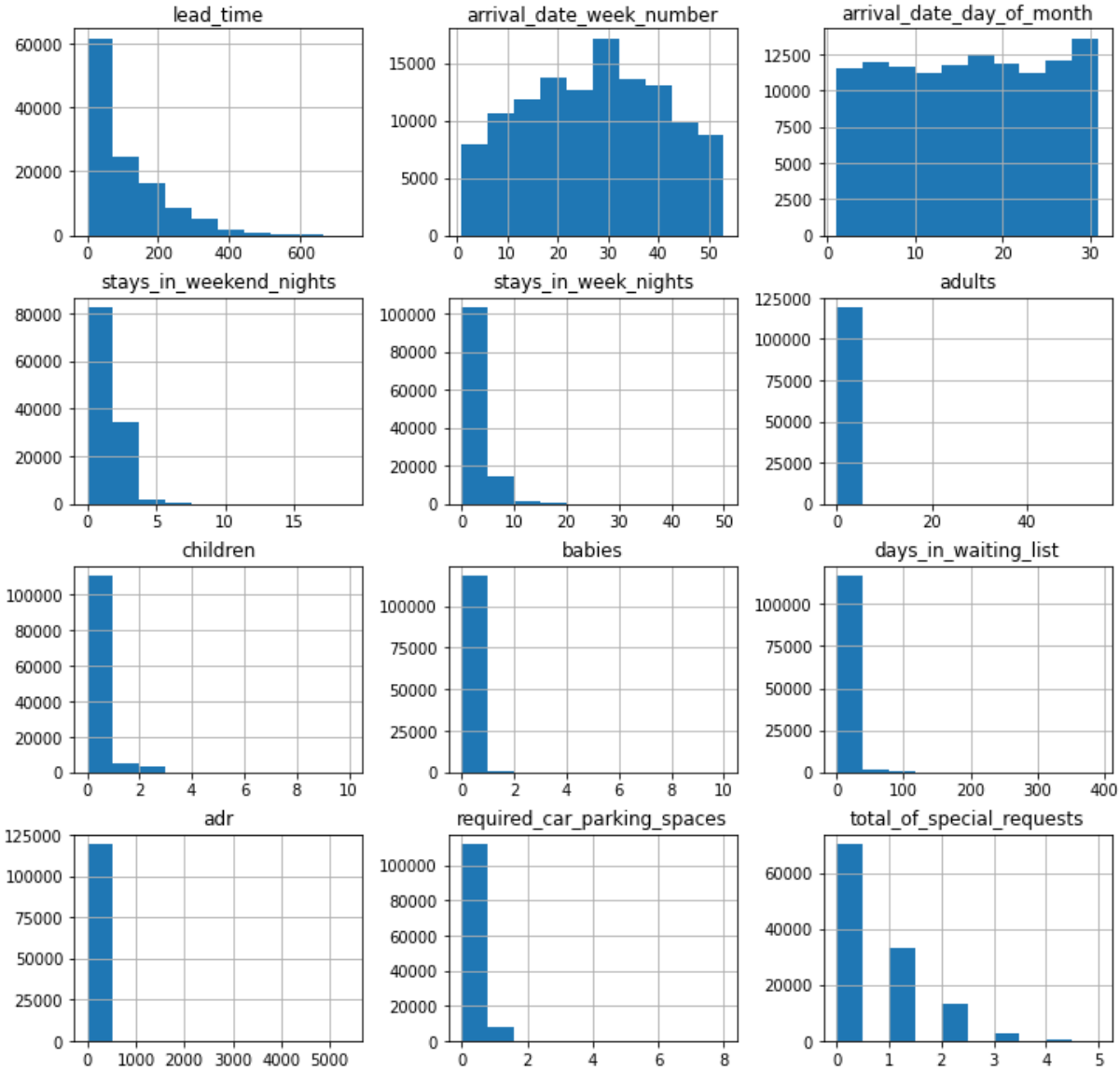
```
In [24]: #We will change all the "Undefined" values for Without Category

         df.replace(to_replace="Undefined", value="Without Category", inplace=True)
```

Luego, pasamos a normalizar estas variables con un threshold de 5%, sino cumplen con este porcentaje, las variables se convertirán en “SMALL_CATEGORIES”. Después, tiramos las variables unarias *is_repeated_guest* y *previous_bookings_not_canceled*.

Variables Continuas

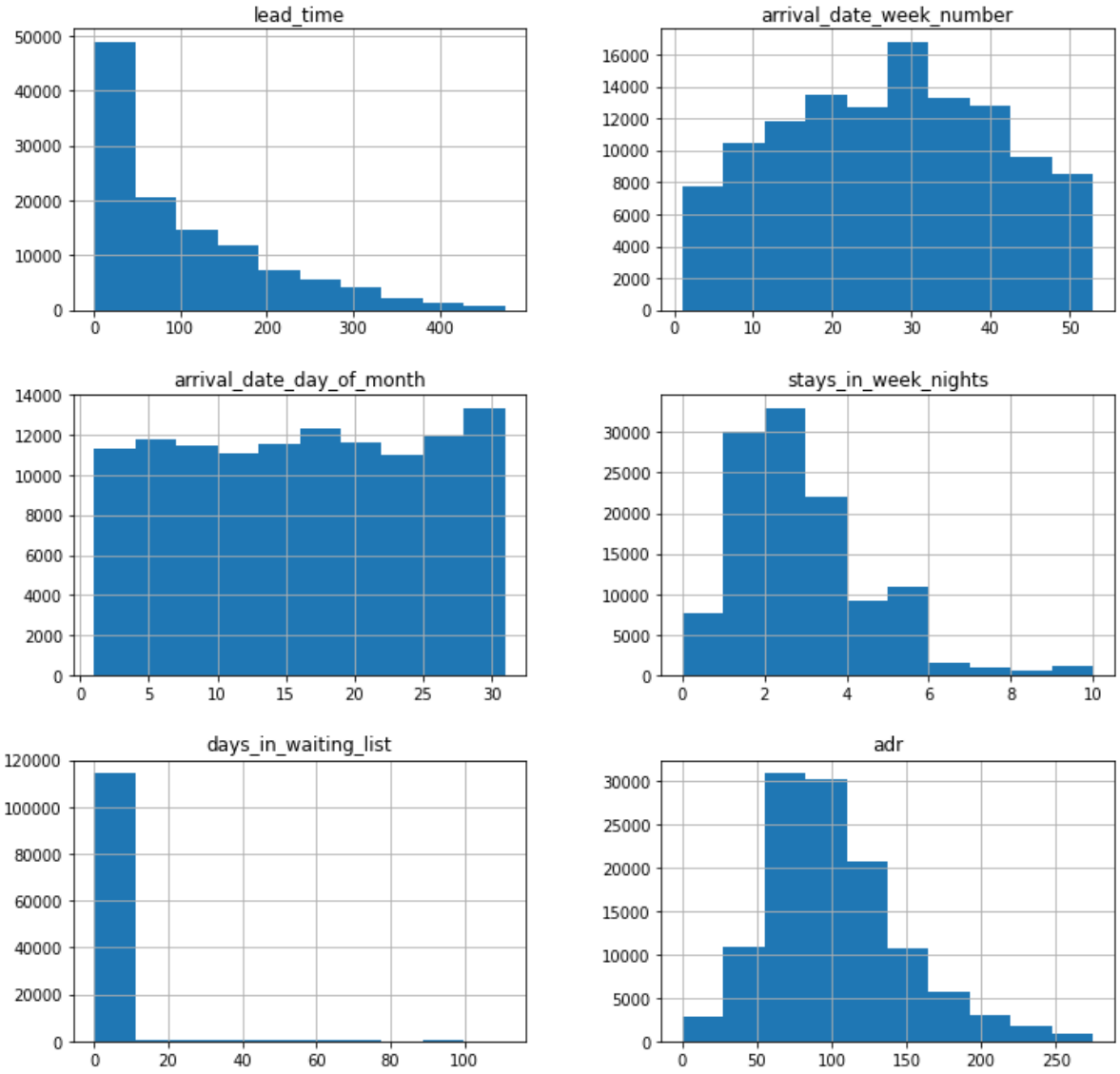
En el caso de las variables continuas, pudimos observar que había unos datos extremos muy significativos. Tansolo en la variable *adr*, el 0.99 percentil tiene un valor de 252; mientras que, en el valor máximo tiene 5,400. Al momento de hacer histogramas, podemos verlo graficamente.



Para hacer tratamiento, nos dimos cuenta que no había valores ausentes, luego hicimos un test de varianza, a lo cual, tiramos 4 variables:

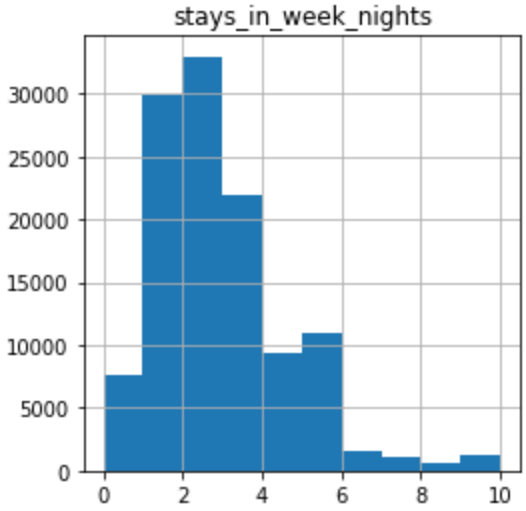
- stays_in_weekend_nights
- Adults
- Children
- Babies
- Required_car_parking_spaces
- Total_of_special_requests

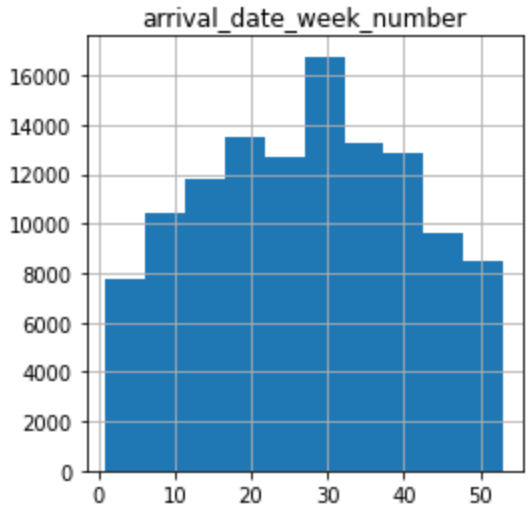
Para poder eliminar valores extremos, utilizamos la metodología de cerca percentil, la cual nos dijo que el 1.8% de los datos eran valores extremos, por lo que los eliminamos. De esta manera, la mayoría de nuestras variables ya tenían mejor distribución.



Al ver estos histogramas, saltó la duda con la variable *days_in_waiting_list*, al analizarla, nos dimos cuenta que casi el 90% de los datos tenían valores < 1 cuando el valor más alto era 111. La eliminamos.

Para el análisis de correlación, nos dimos cuenta que todas eran indepoendientes una de otra. Al pasar a multicolinealidad, las dos mejores variables que obtuvimos fueron: *stays_in_week_nights* y *arrival_date_week_number.*





Pasamos a discretizar con hasta 5 bins distintos. Pero, al elegir las mejores variables con nuestra “IV Function”, nos dimos cuenta que estas no tenían un poder de predicción significativo a comparación de otras. Una vez seleccionadas las mejores variables, pasamos a crear la TAD con la WoE.

Solución - Gradiente Descendiente Estocástico (SGD)

Una vez hecho una limpieza y preparación de los datos con la transformación WoE, hacemos el split de nuestras variables X y Y sobre la TAD, se procedió a hacer un grid de hiper-parámetros, y luego un “RandomizedSearchCV” para encontrar el mejor “setup” para el modelo.

Después de haber transcurrido casi 15 min, encontramos la mejor opción, la cual nos permitía tener un F1-score de 0.81 y 0.6 para valores de entrenamiento y valores muy similares para los datos de validación, de esta manera podemos darnos cuenta que nuestro modelo puede generalizar para los usuarios que vayan a cancelar dadas sus características.

In [99]:
```
print(classification_report(yt, model_best.predict(Xt)))
```

```
              precision    recall  f1-score   support

           0       0.76      0.87      0.81     36405
           1       0.70      0.52      0.60     21032

    accuracy                           0.74     57437
   macro avg       0.73      0.70      0.70     57437
weighted avg       0.74      0.74      0.73     57437
```

In [100]:
```
print(classification_report(yv, model_best.predict(Xv)))
```

```
              precision    recall  f1-score   support

           0       0.76      0.87      0.81     15527
           1       0.71      0.53      0.61      9090

    accuracy                           0.75     24617
   macro avg       0.73      0.70      0.71     24617
weighted avg       0.74      0.75      0.74     24617
```