Introducción

El presente documento corresponde al informe capstone realizado para el análisis de un modelo de datos para la determinación de saturación en un conjunto de datos de consumo de ancho de banda en interfaces de red de un proveedor de internet y servicios de telecomunicaciones.

El mismo consiste en una muestra de 9982 filas con 31 columnas incluyendo el nombre de la interfaz y los datos de consumo de 30 días agrupados en un dato por día.

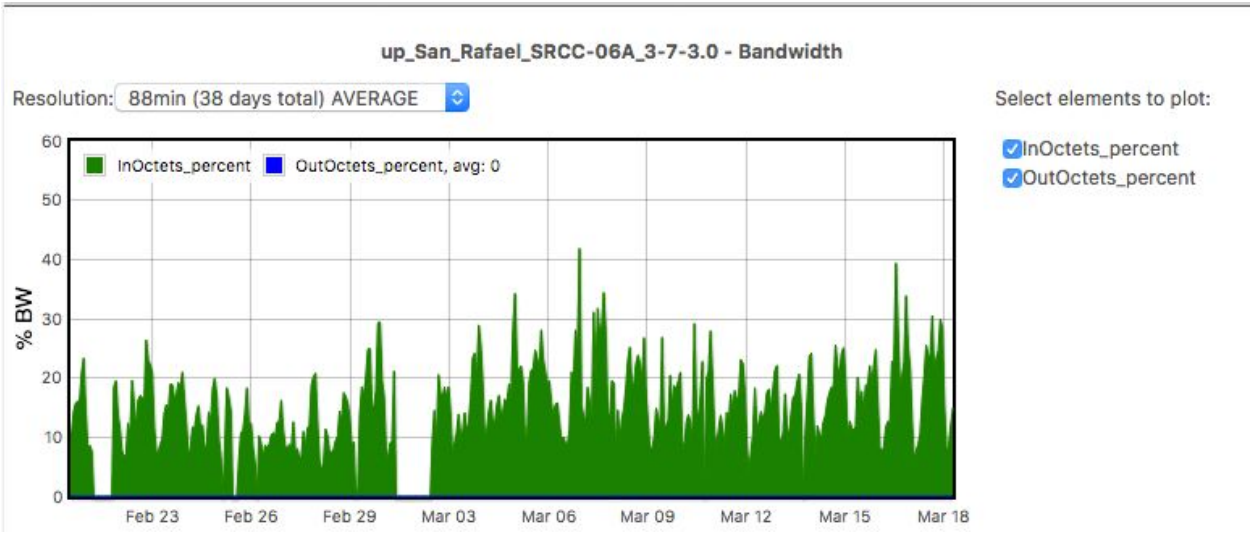
Descripción del DataSet

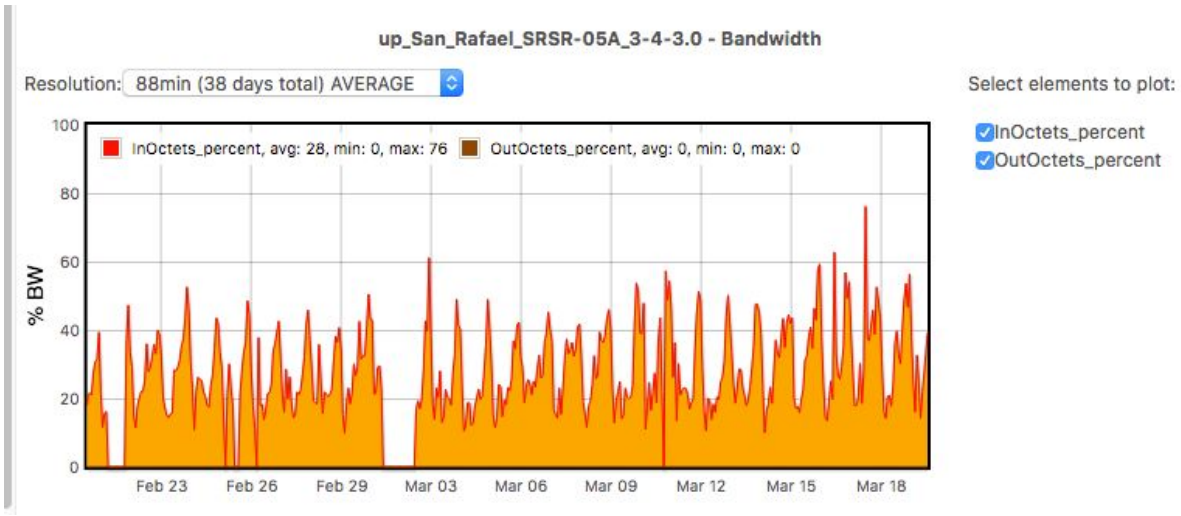
La fuente de los datos es un sistema de monitoreo que realiza una captura de consumo de ancho de banda de n interfaces de downstream y Upstream cada 2 minutos. El cálculo de ancho de banda se realiza con base en los contadores brindados por SNMP en el equipo donde se encuentra la interfaz y el tiempo transcurrido entre la lectura anterior y la actual.
Estos datos son guardados en una base de datos round robin (RRD) donde a las 24 horas son agregados con los datos del día anterior.

Por este motivo cada 24 horas (antes de la agregación) se ejecuta un proceso que obtienen la totalidad de las muestras para cada interfaz y con base en una herramienta que se llama rrdtool, calcula el percentil 95 de los datos y obtiene el valor que corresponde con dicho percentil y lo guarda. Esto dado que según las mejores prácticas y recomendaciones de la industria el percentil 95 representa de la mejor manera el valor normal debajo del cual el consumo de ancho de banda de una interfaz se puede encontrar (El 5% restante pueden ser picos puntuales y descartables).

El dataset contiene las lecturas de dichos percentiles en el último mes para un conjunto de alrededor de 9000 interfaces de subida y bajada en una red de proveedor de internet y se obtiene por medio de un script de perl que recorre el listado total de interfaces existentes en el monitoreo y obteniendo los valores correspondientes en el archivo para el día. En este caso se recolectó información desde el 20 de Febrero y hasta el 19 de marzo (1 mes).

Individualmente cada interfaz se muestra y representa el histórico de consumo de la siguiente forma





También es posible por el mismo procedimiento obtener la información de otras variables como SNR(Señal a Ruido), Errores, Descartes, entre otros, sin embargo nos enfocaremos en ancho de banda.

El ancho de banda se encuentra segmentado en IN y OUT dado que unas interfaces hacen el IN(Upstream) y otras el OUT (Downstreams)

Descripción del Problema a resolver

Se dice que una interfaz se muestra saturada si alcanza más del 80% de utilización y en un ambiente de servicios de telecomunicaciones e internet este es un dato crítico para poder determinar en qué sector o sectores de la red se deben realizar trabajos ya sea de mejora o de correcciones, por lo tanto el objetivo con este ejercicio es determinar cuáles interfaces presentan saturación en el periodo de 1 mes.

Análisis Realizado

En primera instancia se revisó el dataset en excel y se le eliminó algunas columnas no representativas para el ejercicio, además se agregó inicialmente una columna con el cálculo de saturación basado en la obtención del promedio de uso para el mes y marcando con uno aquellas cuyo promedio se encuentra en 80 y superior. Posteriormente se aplicará la generación de modelos para validar la factibilidad de generar dicha información sin tener que manualmente generar los cálculos para cada interfaz.

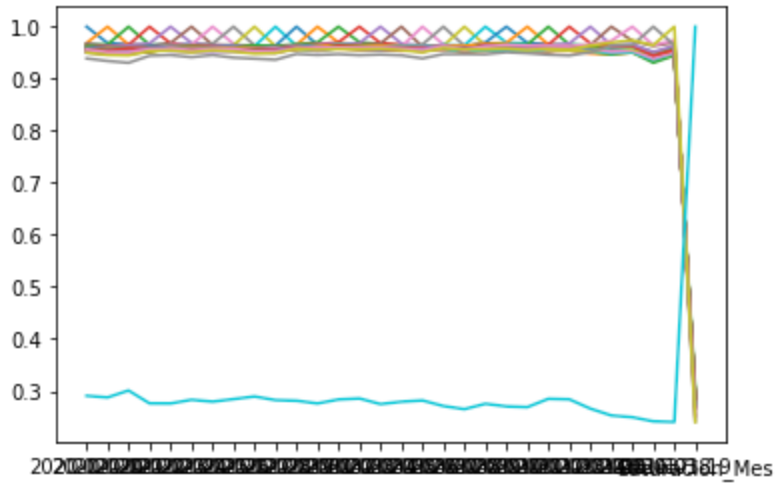
Adicionalmente se buscará la forma ágil por medio de python para extraer el listado de aquellas interfaces que se indiquen como saturadas y que por lo tanto son las que el cliente desea conocer.

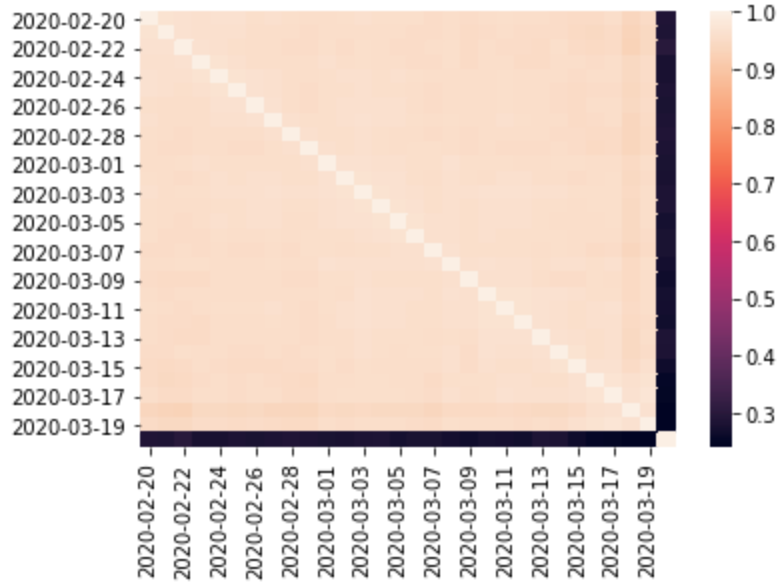
También se obtendrán algunos gráficos que nos permitiran validar de manera gráfica el comportamiento general de la red y de las interfaces saturas.

Matriz de correlación

Se obtiene la matriz de correlación para la totalidad de las columnas, dado que corresponden a valores diarios, se espera que la relación entre sus columnas sea alta, lo cual se ve reflejado en el gráfico de correlación

Así mismo se observa que la correlación es baja con nuestra columna a evaluar (Variable dependiente).

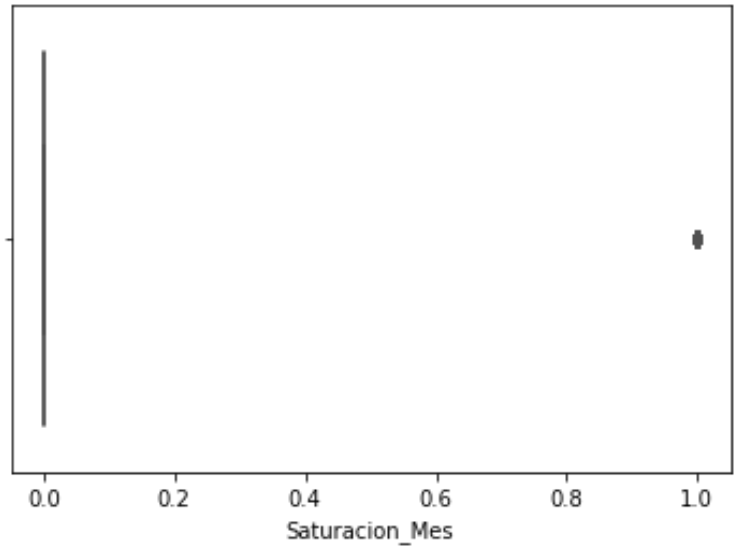


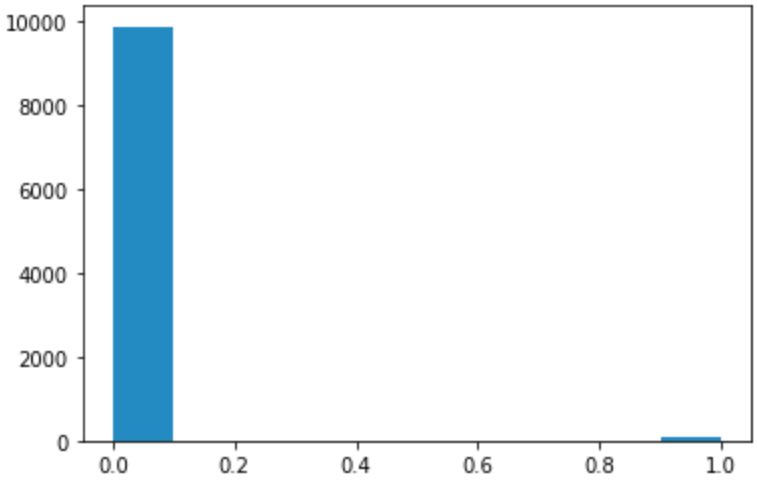


De esta forma y considerando que lo que nos interesa es conocer la saturación para el mes, no se elimina ninguna columna adicional del dataset.

Observaciones generales de la muestra.

Inicialmente se generó un gráfico que nos permite ver la distribución de la saturación entre la totalidad de las muestras y la relación entre saturación y no saturación.

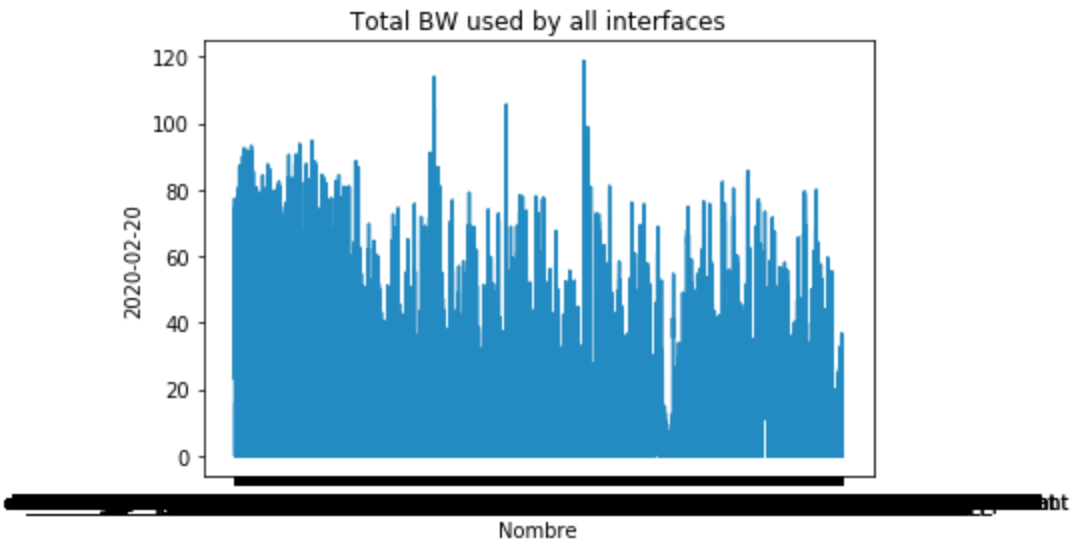




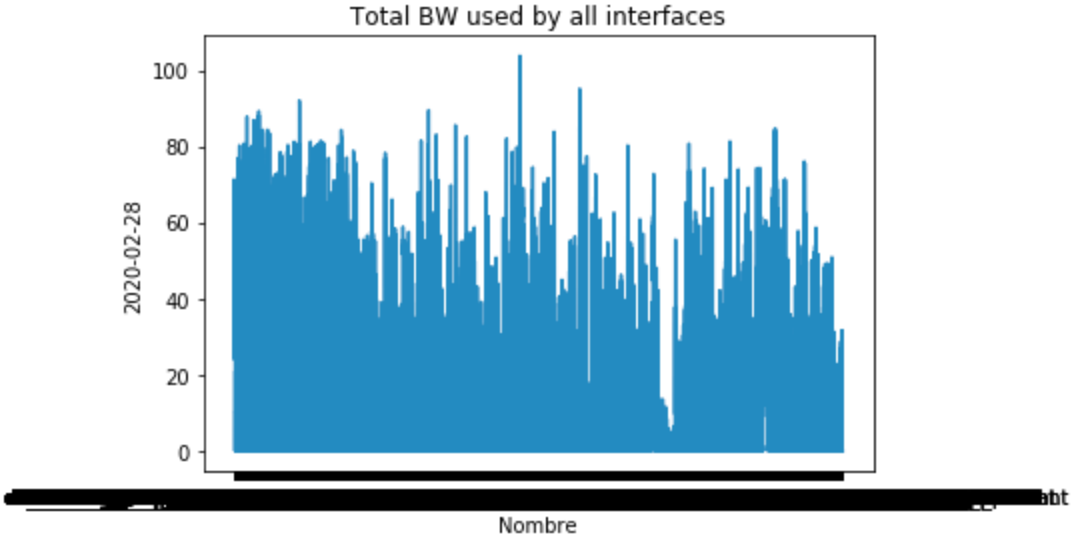
Inicialmente se observa que el nivel de saturación de la red es bajo con respecto a la totalidad de interfaces, sin embargo el número de interfaces que se encuentre saturada es importante para el mantenimiento de la red.

Comparación de la generalidad de consumo en la red.

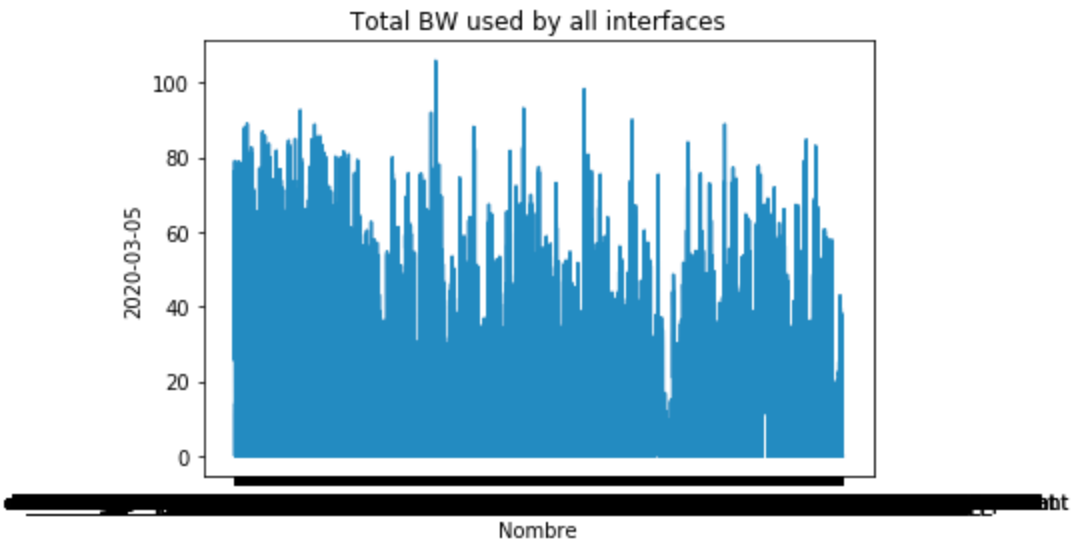
Se obtuvieron gráficos del día inicial y cada 5 o 7 días en adelante para evaluar el comportamiento de la red.



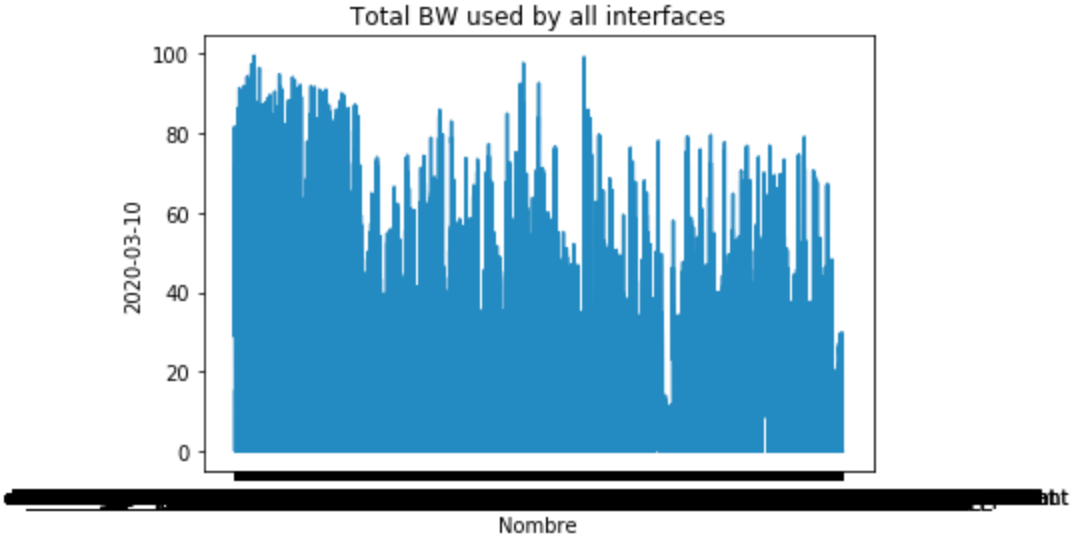
Comportamiento general del consumo de ancho de banda el 20 de febrero.



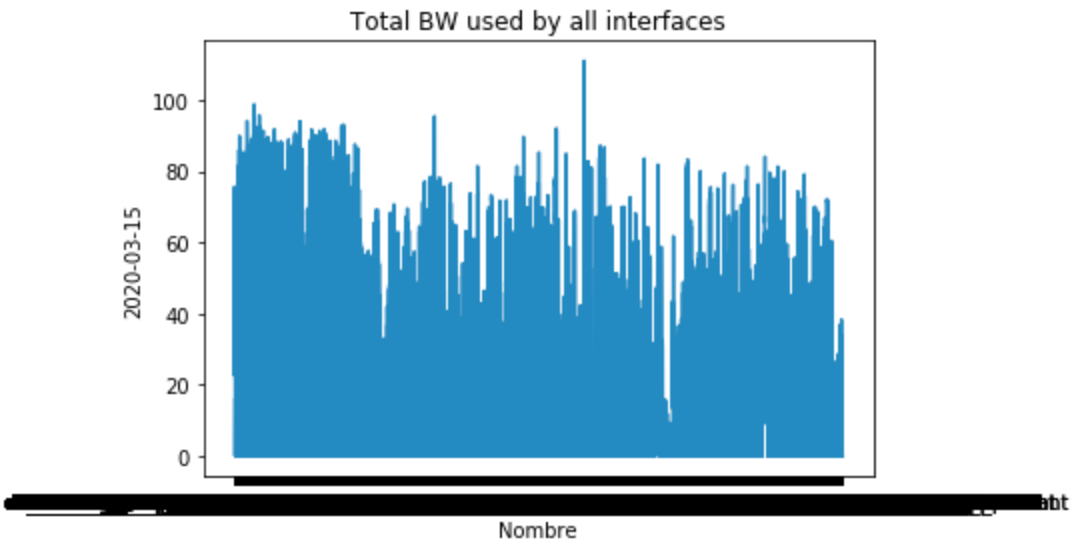
Comportamiento general del consumo de ancho de banda el 28 de febrero.



Comportamiento general del consumo de ancho de banda el 05 de Marzo.



Comportamiento general del consumo de ancho de banda el 10 de Marzo. Aquí se observa una mayor concentración de interfaces arriba de 80% de uso



Comportamiento general del consumo de ancho de banda el 15 de Marzo. Aquí se observa que se mantiene la tendencia presentada a partir del 10.

Adicionalmente se procedió a agrupar las interfaces según el dato de nuestro interés, por lo tanto se generó una agrupación de interfaces saturadas para poder evaluar este grupo específicamente.

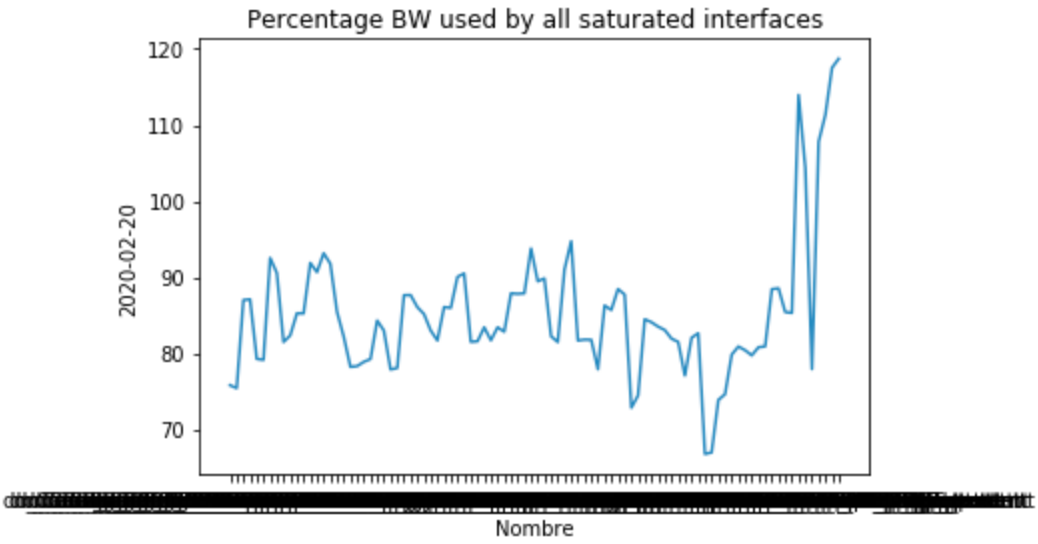
```
In [161]: #Crear agrupación de interfaces de acuerdo a si están saturadas o no
          saturacion = BWFinal.groupby('Saturacion_Mes')
          saturacion.first()
```

Out[161]:

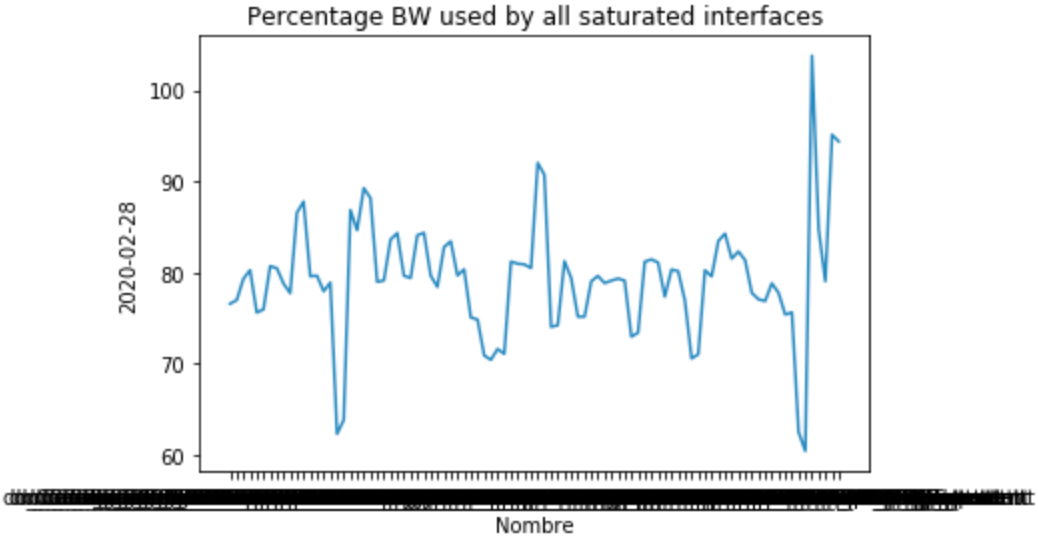
Saturacion_Mes	Nombre	2020-02-20	2020-02-21	2020-02-22	2020-02-23	2020-02-24	2020-02-25	2020-02-26	2020-02-27	202
0	dn_Belen_d0_0_0_Bandwidth+OUT_percent	73.95	69.80	70.98	70.73	72.67	73.87	67.83	67.56	
1	dn_Cartago_d0_3_0_Bandwidth+OUT_percent	75.80	83.44	80.80	83.69	69.28	70.68	72.44	81.08	

2 rows × 30 columns

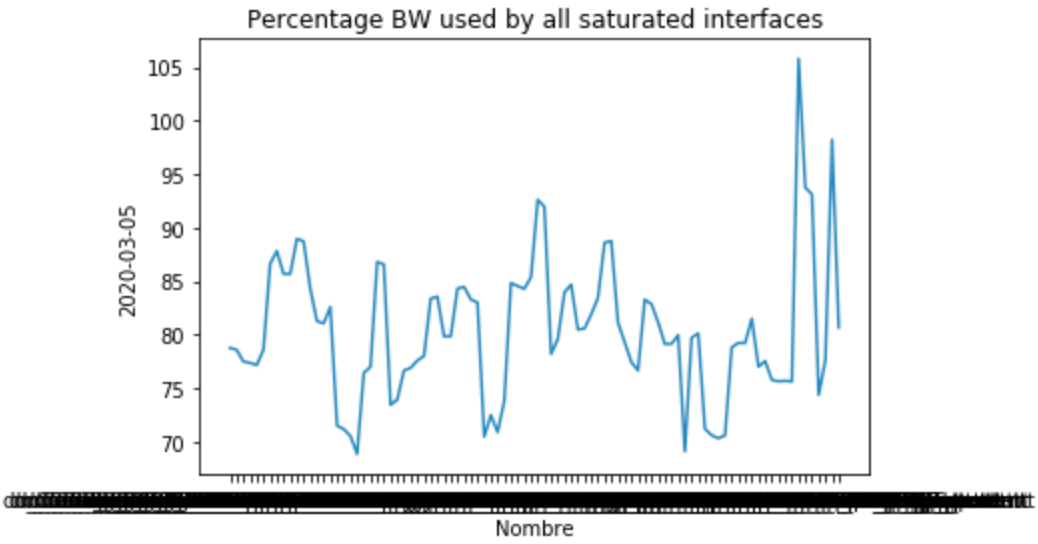
Por este medio se determina que el total de interfaces saturadas es de 92 y se procede a evaluar el comportamiento de este grupo de manera similar que la totalidad de la red.



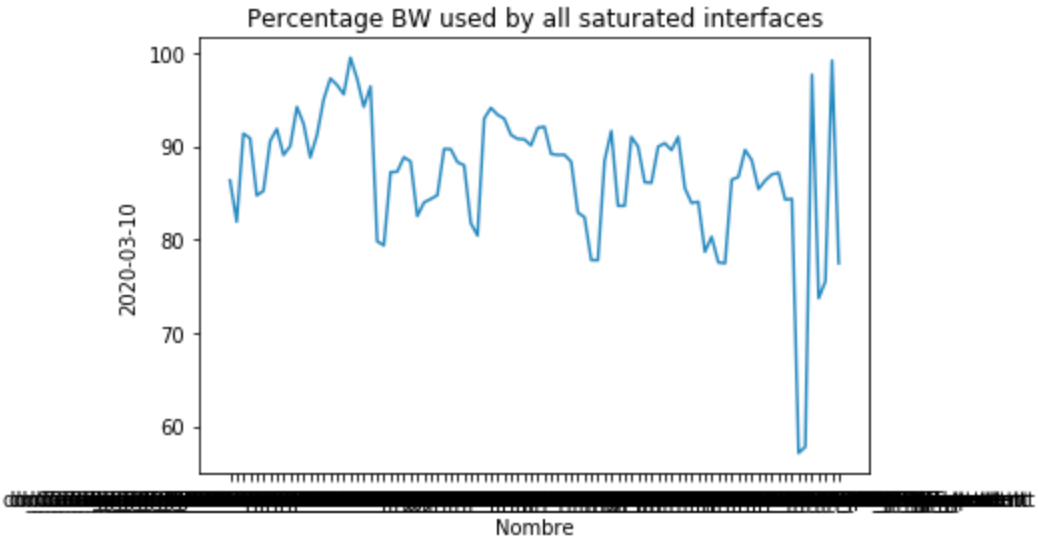
Comportamiento de las interfaces saturadas el 20 de febrero



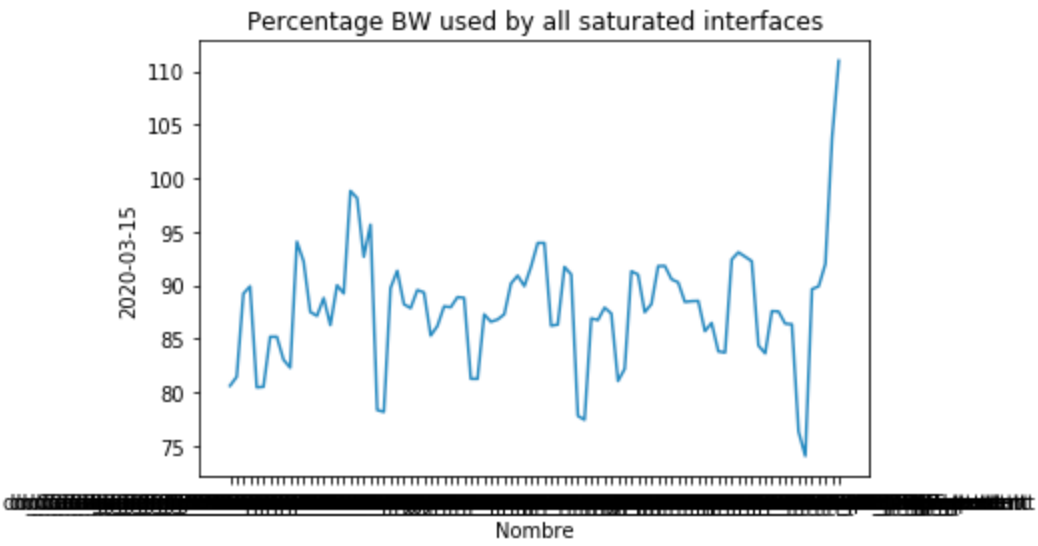
Comportamiento de las interfaces saturadas el 28 de febrero



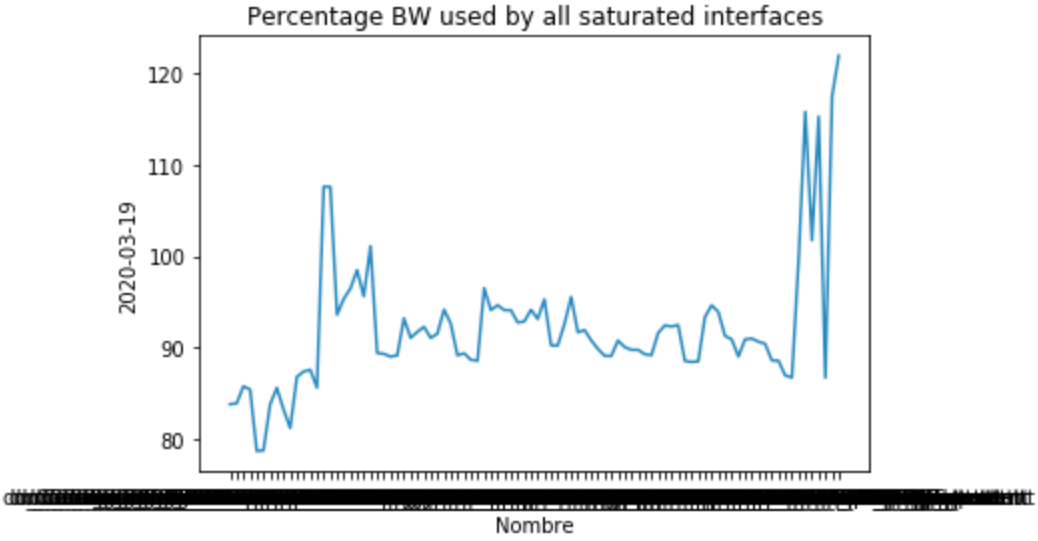
Comportamiento de las interfaces saturadas el 05 de marzo



Comportamiento de las interfaces saturadas el 10 de marzo



Comportamiento de las interfaces saturadas el 15 de marzo



Comportamiento de las interfaces saturadas el 20 de marzo

Como se observó en el comportamiento general, también se observa que en las interfaces saturadas se presenta el aumento en el consumo de ancho de banda.

Evaluación de los modelos.

Al ser este ejercicio una clasificación entre si una interfaz presenta saturación o no, se evaluaron los siguientes modelos de clasificación:

1. Random Forest
2. Support Vector Machine (SVC)
3. Linear Regression

4. Logistic Regression
5. DecisionTreeClassifier
6. KNeighborsClassifier (knn)
7. Linear Discriminant Analysis.

Preparación de la muestra

Se separó un total de 1000 filas del total de la muestra y a este se le aplicó una separación estándar de 70-30 para el entrenamiento y el testing.

```
# Definición del tamaño de la muestra de entranamiento
X_train = (features[: 1000])
X_train.head()
```

	2020-02-20	2020-02-21	2020-02-22	2020-02-23	2020-02-24	2020-02-25	2020-02-26	2020-02-27	2020-02-28	2020-02-29	...	2020-03-11	2020-03-12	2020-03
0	73.95	69.80	70.98	70.73	72.67	73.87	67.83	67.56	70.10	65.16	...	70.59	64.01	74
1	73.23	65.24	71.18	70.55	73.03	73.73	67.75	66.87	71.35	65.30	...	71.14	64.67	73
2	23.81	30.81	30.92	26.54	22.39	23.12	22.86	23.28	24.09	23.12	...	21.88	27.98	21
3	23.34	30.83	30.49	26.59	22.24	23.02	22.57	23.20	24.53	22.82	...	29.34	27.11	44
4	31.30	40.79	31.03	33.33	34.24	32.25	30.47	30.90	31.99	29.26	...	36.15	39.77	33

5 rows × 30 columns

```
n [472]: #Dependent Variable Training Set (y Training)
         y_train = depVar[: 1000]
         y_train_count = len(y_train.index)
         print('The number of observations in the Y training set are:',str(y_train_count))
         y_train.head()

         The number of observations in the Y training set are: 1000

ut[472]: 0    0
         1    0
         2    0
         3    0
         4    0
         Name: Saturacion_Mes, dtype: int64
```

Además se aplicó la separación para generar la evaluación de los diferentes modelos.

```
In [392]: #Cross Validation
          X_train, X_test, y_train, y_test = train_test_split(X_train, y_train)

In [393]: X_train.shape, X_test.shape

Out[393]: ((750, 30), (250, 30))

In [394]: X_train, X_test, y_train, y_test = train_test_split(X_train, y_train)
          X_train.shape, X_test.shape

Out[394]: ((562, 30), (188, 30))
```

Evaluación de los modelos indicados anteriormente.

Al evaluar los modelos contra la muestra se encontró que el rendimiento de muchos de los modelos presentaban un perfect fit, por lo que estos no podrían ser utilizados para las predicciones. Dichos modelos fueron los siguientes

1. Random Forest
2. Linear Regression
3. Logistic Regression
4. DecisionTreeClassifier
5. Linear Discriminant Analysis.
6. GaussinaNB

Los modelos que no dieron perfect fit y por lo tanto podemos considerar para nuestro modelado fueron los siguientes.

1. KNeighbors Classifier (Knn) 0.996

```
#KNeighborsClassifier
Modelknn = KNeighborsClassifier()
Modelknn.fit(X_train, y_train)
print('Accuracy of K-NN classifier on training set: {:.2f}'
     .format(Modelknn.score(X_train, y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'
     .format(Modelknn.score(X_test, y_test)))
```

```
Accuracy of K-NN classifier on training set: 1.00
Accuracy of K-NN classifier on test set: 0.98
```

```
print(cross_val_score(Modelknn, X_train, y_train))
Modelknn.score(X_train,y_train)
```

```
[1.         0.99115044 0.98214286 1.         0.99107143]

0.99644128113879
```

2. SVC (0.989)

```
#Model Fitting SVC
modelSVC.fit(X_train,y_train)
print(cross_val_score(modelSVC, X_train, y_train))
modelSVC.score(X_train,y_train)
```

```
[1.         0.97345133 0.98214286 0.99107143 0.99107143]

0.9893238434163701
```

3. Linear Discriminant Analysis (0.964)

```
#Linear Discriminant Analysis
ModelLda = LinearDiscriminantAnalysis()
ModelLda.fit(X_train, y_train)
print('Accuracy of LDA classifier on training set: {:.2f}'
     .format(ModelLda.score(X_train, y_train)))
print('Accuracy of LDA classifier on test set: {:.2f}'
     .format(ModelLda.score(X_test, y_test)))
```

```
Accuracy of LDA classifier on training set: 0.96
Accuracy of LDA classifier on test set: 0.93
```

```
print(cross_val_score(ModelLda, X_train, y_train))
ModelLda.score(X_train,y_train)
```

```
[0.92035398 0.98230088 0.94642857 0.98214286 0.95535714]

0.9644128113879004
```

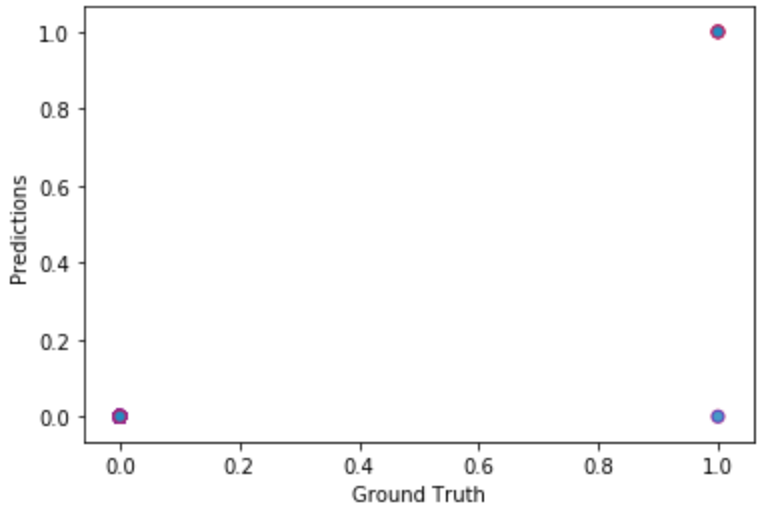
Predicciones

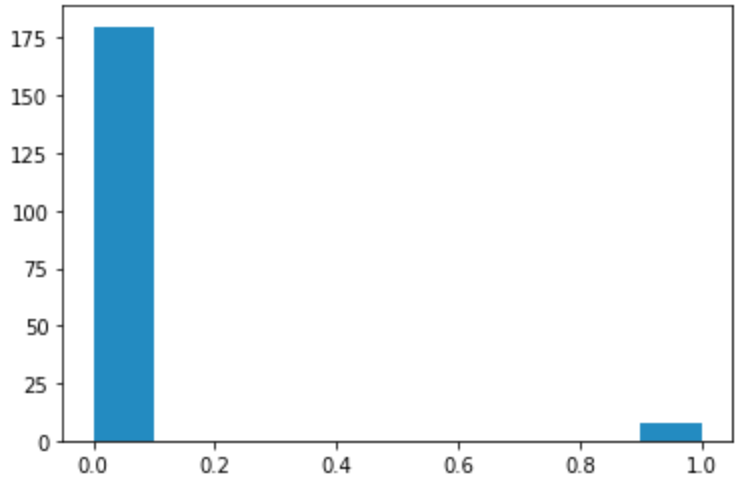
Habiendo evaluado los modelos anteriores se realizó la evaluación de las predicciones con base en el modelo KNN que fue el que presentó el mejor rendimiento. Obteniéndose la siguiente tabla de resultados.

```
predictions = Modelknn.predict(X_test)
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(predictions)
```

```
[[176   1]
 [  3   8]]
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       177
           1       0.89      0.73      0.80        11

    accuracy                           0.98       188
   macro avg       0.94      0.86      0.89       188
weighted avg       0.98      0.98      0.98       188

[0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0]
```

Siendo que dichas predicciones se pueden representar de la siguiente forma





Finalmente habiendo elegido el modelo se aplicó este sobre el total de los datos para comparar y evaluar su rendimiento, habiendo asignado las 9982 filas y aplicando la preparación del dataset se obtuvo el siguiente cuadro de rendimiento

```
#Obtener el total de las muestras para aplicar el modelo
BWTotal = (features[: 9982])
```

```
#Asignar la variable dependiente a toda la muestra
y_Total = depVar[: 9982]
```

```
#Reemplazar NANs en la totalidad de las muestras
BWTotal=BWTotal.fillna(0)
```

```
#Validar contra el total de las muestras
predictionsFinal = Modelknn.predict(BWTotal)
print(confusion_matrix(y_Total, predictionsFinal))
print(classification_report(y_Total, predictionsFinal))
print(predictionsFinal)
```

```
[[9883    7]
 [  13   79]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      9890
           1       0.92      0.86      0.89        92

    accuracy                           1.00      9982
   macro avg       0.96      0.93      0.94      9982
weighted avg       1.00      1.00      1.00      9982

[0 0 0 ... 0 0 0]
```

Y por último se agregaron la columna de nombres y de predicciones a nuestro set de datos de evaluación y se exportó a csv

```
BWTotal['Nombre']=Bandwidth['Nombre']
BWTotal['predictions']=predictionsFinal
```

```
BWTotal.head()
```

	2020-02-20	2020-02-21	2020-02-22	2020-02-23	2020-02-24	2020-02-25	2020-02-26	2020-02-27	2020-02-28	2020-02-29	...	2020-03-13	2020-03-14	2020-03
0	73.95	69.80	70.98	70.73	72.67	73.87	67.83	67.56	70.10	65.16	...	74.58	66.12	7
1	73.23	65.24	71.18	70.55	73.03	73.73	67.75	66.87	71.35	65.30	...	73.86	65.99	7
2	23.81	30.81	30.92	26.54	22.39	23.12	22.86	23.28	24.09	23.12	...	21.59	23.18	2
3	23.34	30.83	30.49	26.59	22.24	23.02	22.57	23.20	24.53	22.82	...	44.78	28.56	2
4	31.30	40.79	31.03	33.33	34.24	32.25	30.47	30.90	31.99	29.26	...	33.83	33.58	3

5 rows × 32 columns

```
BWTotal.to_csv(r"BWFilaPred.csv",index=False)
```

El resultado final debería poder generar un filtrado de datos evaluados a 1 (saturación) el cual debe ser entregado al cliente para su evaluación y atención correspondiente.

Conclusiones

1. Se determina que se pueden evaluar y preparar modelos de clasificación para determinar la recurrencia de saturación en un conjunto de interfaces de la red.
2. Se podría aplicar este mismo esquema de modelado y predicción para sets de datos con menor cantidad de columnas, por ejemplo semanal, quincenal, etc.
3. Los modelos evaluados presentan alto rendimiento, por lo que podría decirse que se pueden aplicar a datos de producción con una alta fiabilidad.
4. Se podría mejorar aplicar este mismo modelado de datos a otras variables que el cliente desea evaluar como Señal a Ruido, Errores Corregidos entre otros.