Mentoría Data Science aplicado a BCI
Grupo 2
TP4: Aprendizaje Supervisado

A) Entendimiento del problema.

a) Está claro que hasta aquí hablamos de un problema de clasificación supervisada, pero ¿a qué subclase dentro de ellos pertenece? (multilabel? multiclase?)

Haciendo referencia al marco teórico:

Clasificación multiclase significa una tarea de clasificación con más de dos clases, parte del supuesto de que cada muestra está asignada a una sola etiqueta. clasificación multilabel asigna a cada muestra un conjunto de etiquetas de destino, no son mutuamente excluyentes.

La clasificación multilabel se da cuando cada muestra puede tener más de un label correspondiente a n clases.

https://scikit-learn.org/stable/modules/multiclass.html

En base a esta información podemos enmarcar al caso de uso como un problema de clasificación supervisada, subclase multiclase, ya que cada existen tres clases: "label 1, label 2 y label 99" y son mutuamente excluyentes entre sí. Un dato a clasificar no puede tomar valores de dos clases diferentes.

b) En palabras, describa cómo podría plantearse un problema de regresión con los datos estudiados.

Teniendo en cuenta que los datos obtenido en nuestro experimento son similares a lo de un EEG, podríamos plantear la siguiente incógnita:

¿Es posible el filtrado espacial del EEG como un problema de Regresión?

La señal EEG sin procesar consiste en la señal ERP (potenciales relacionados con eventos) y ruido. Un buen filtrado espacial nos permitirá lograr reducir el ruido y mantener la señal ERP.
Sabemos, por teoría de señales, que en la señal promediada de EEG tenemos la señal ERP casi sin ruido. Podemos plantear una transformación con el objetivo de lograr que la señal resultante sea similar al de ERP sin ruido utilizando regresión para encontrar un conjunto de pesos w (coeficientes de un polígono) que minimice la diferencia entre la señal ERP sin ruido y la señal EEG filtrada espacialmente.
Para lo cual se aplicaría un método de regresión donde X es la señal concatenada del EEG e Y es la señal concatenada del EEG promediada sin ruido. De esta forma obtendremos un polígono que nos permitirá filtrar la señal EEG logrando obtener la señal que nos interesa para nuestro estudio (ERP).

Link para más info:
https://www.researchgate.net/profile/Martin-Spueler/publication/319903900_Spatial_Filtering_of_EEG_as_a_Regression_Problem/links/59c10ce6458515af305c49bd/Spatial-Filtering-of-EEG-as-a-Regression-Problem.pdf

c) De la misma forma, ¿cómo podría pensarse un problema de clasificación no supervisada a partir de los datos disponibles?

Teniendo en cuenta el concepto teórico de aprendizaje no supervisado, donde la forma de trabajar de este tipo de modelos es aumentando el conocimiento estructural de los datos disponibles y posibles datos futuros que provengan de un experimento similar, logrando una agrupación de los datos según su similitud (concepto de clustering) podríamos plantear la búsqueda de un modelo no supervisado donde el objetivo sería lograr identificar outlier en nuestro conjunto de features, para luego eliminarlos y lograr de esta forma una mayor precisión y menor variabilidad en los mismo.

B) En el problema de clasificación supervisada:

a) ¿Qué métricas considera que son apropiadas para evaluar el desempeño de ¿Algún algoritmo de clasificación? Recuerde limitar la cantidad que elija a un valor pequeño ya que serán usadas para evaluar los algoritmos de la próxima sección.

Al ser un problema de clasificación supervisada, las métricas más apropiadas son:
- **Accuracy**, indica el número de elementos clasificados correctamente en comparación con el número total de artículos.
- **Recall**, indica la cantidad de verdaderos positivos que el modelo ha clasificado en función del número total de valores positivos.
- **Precisión**, indica el número de verdaderos positivos que son realmente positivos en comparación con el número total de valores positivos predichos.

Consideramos que la métrica más apropiada es **F1 score**, que relaciona Precisión y Recall, de esta forma estaremos teniendo una buena idea si el modelo está clasificando correctamente con respecto a falsos negativos (recall) y los falsos positivos (Precisión).

F1 score= Precisión*Recall/Precision+Recall
valor entre 0 (peor) y 1 (mejor valor).

Para evaluar nuestros modelos observaremos **F1-Score** por clase (label 1, 2 y 99) y el **F1 Score macro** (promedio general), y reforzaremos la evaluación teniendo en cuenta la **Accuracy** de cada uno.

b) ¿Qué funciones de costo consideran apropiadas para entrenar un algoritmo para este problema?

La funciones que consideramos podrían ser apropiadas para entrenar un algoritmo de clasificación son las siguientes:

- Categorical Cross Entropy Loss

Dicha función de costo es la cross entropy binaria extendida a multiclase. Utiliza softmax, para dicho cálculo.

- Hinge Loss

Es utilizada para SVM donde calcula el margen máximo en el hiperplano qué divide las clases

- Negative log-likelihood

Utilizado para regresión logística.

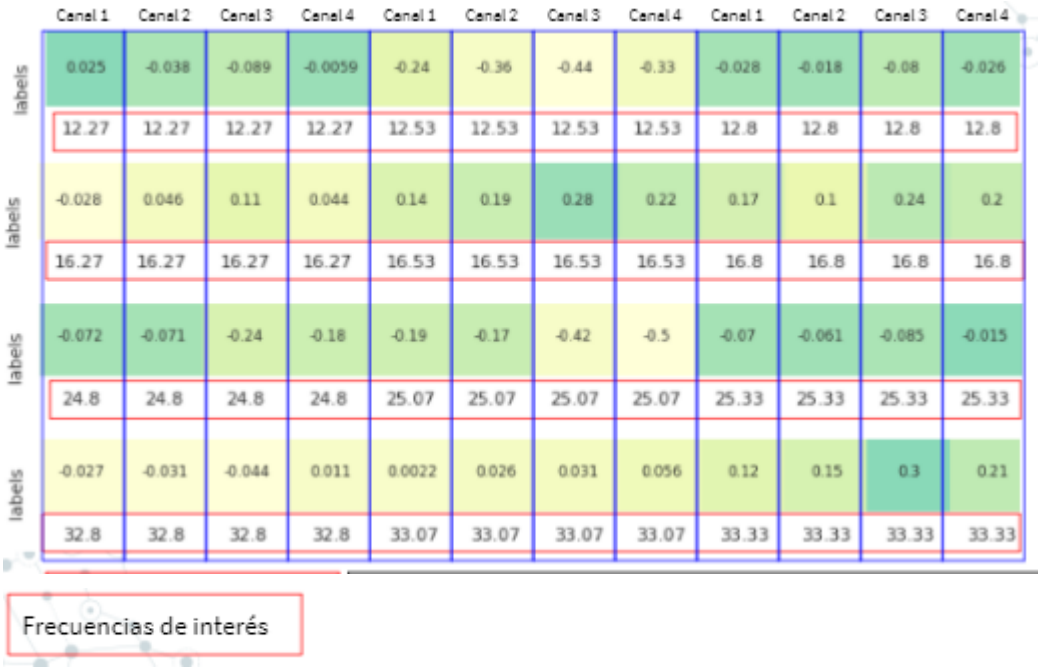
En conclusión la definición de una función de costo depende del modelo de clasificación que vamos a utilizar. Una vez definidos los modelos a utilizar tendremos más especificaciones de qué función de costo es apropiada.

Aprendizaje Supervisado

Extracción de nuevos features:

Como principal desafío de la extracción de los features es que con el dataset brindado se deben realizar ciertas transformaciones para poder determinar cuales son los features a extraer, a diferencia de otros problemas que las mismas columnas brindadas en el dataset se usan como features.
También se debe mencionar que los distintos individuos pueden generar respuestas fisiológicas distintas a la frecuencia objetivo, por lo que el problema es muy sensible a ruidos por el uso en distintas personas. En este caso se optó por utilizar los valores de todos los individuos en todas las sesiones para buscar la mayor generalización posible pero como indicaremos más adelante se plantea que los resultados pueden mejorar con el uso personificado de los algoritmos por persona para descartar variabilidades entre personas. Esto se podría pensar como un overfitting por persona pero lo cuál puede ser deseable en está aplicación ya que se busca que el modelo pueda aprender las señales de la persona que lo tiene en uso.

Para este trabajo se volvió a buscar features que mejoren los resultados de correlación del trabajo anterior. Para ello primero se tomó la resultante de la fft de cada frecuencia dentro del filtro (10 a 35 Hz) y para observar si las frecuencias de interés tenían correlación con los labels, se decidió tomar un rango de +- 0,5 Hz alrededor de la frecuencia de interés (12,5 y 16,5 Hz) y de sus primer armónico (25 y 33 Hz).
Para la visualización de la correlación se optó por retirar las etiquetas 99 así se disminuye la variabilidad de los datos y mejora la misma.
Nota: Se muestra sólo una extracción de cada matriz de correlación de las frecuencias de interés contra el label para mayor simplicidad del gráfico.



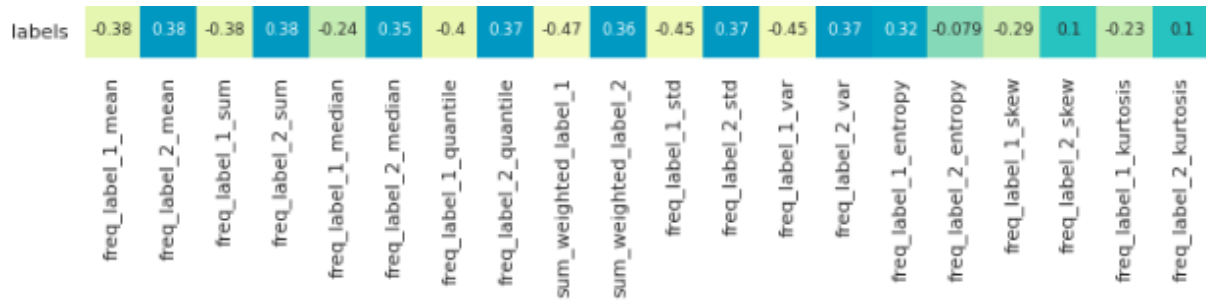
	Canal 1	Canal 2	Canal 3	Canal 4	Canal 1	Canal 2	Canal 3	Canal 4	Canal 1	Canal 2	Canal 3	Canal 4
labels	0.025	-0.038	-0.089	-0.0059	-0.24	-0.36	-0.44	-0.33	-0.028	-0.018	-0.08	-0.026
	12.27	12.27	12.27	12.27	12.53	12.53	12.53	12.53	12.8	12.8	12.8	12.8
labels	-0.028	0.046	0.11	0.044	0.14	0.19	0.28	0.22	0.17	0.1	0.24	0.2
	16.27	16.27	16.27	16.27	16.53	16.53	16.53	16.53	16.8	16.8	16.8	16.8
labels	-0.072	-0.071	-0.24	-0.18	-0.19	-0.17	-0.42	-0.5	-0.07	-0.061	-0.085	-0.015
	24.8	24.8	24.8	24.8	25.07	25.07	25.07	25.07	25.33	25.33	25.33	25.33
labels	-0.027	-0.031	-0.044	0.011	0.0022	0.026	0.031	0.056	0.12	0.15	0.3	0.21
	32.8	32.8	32.8	32.8	33.07	33.07	33.07	33.07	33.33	33.33	33.33	33.33

Frecuencias de interés

Vemos en está matriz como primero las mayores correlaciones están solamente en una de las tres frecuencias del rango tomada, y a su vez como varía entre canal y canal.

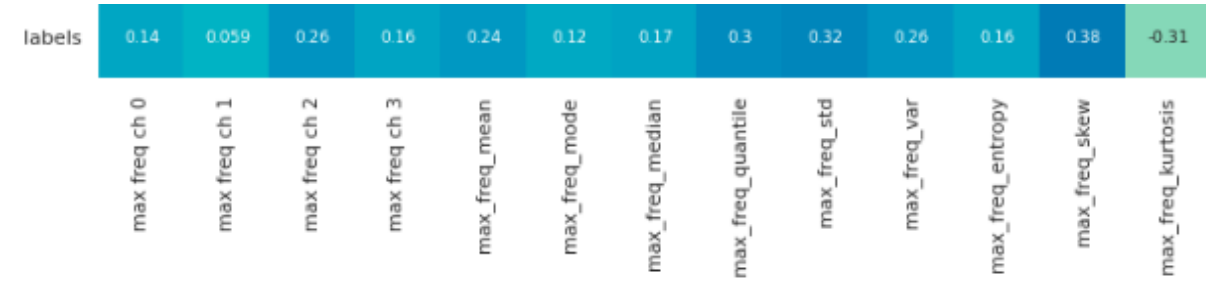
Luego de ver cómo actuaban los valores de frecuencia de interés se procedió a calcular distintos estadísticos para unificar los valores de los 4 canales de cada muestra en un solo valor usando también el rango de frecuencia cercana a la objetivo. Con ese objetivo se obtuvo la media, la mediana, el quantil, la suma y la suma ponderada dandole mayor peso a la frecuencia central y menos peso a las lejanas, el desvío estándar, la varianza, la entropía entre los datos, el sesgo y su kurtosis.

Luego realizamos el mismo análisis de correlación entre los estadísticos y los labels sin la etiqueta 99 para ver su influencia.
Se utiliza el nombre freq_label_1 cuando el estadístico resume información de las frecuencias cercanas a 12,5 Hz y freq_label_2 cuando ese estadístico hace lo mismo pero para las frecuencias cercanas a 16,5 Hz.



Destacamos que en los dos casos de features mostrados ocurre la aparición de correlaciones positivas y negativas para las distintas frecuencias de interés debiéndose a la manera del etiquetado que se optó para cada frecuencia. Esto se da porque cuando la amplitud crece en la frecuencia de 12,5 hz el valor de la etiqueta decrece a 1 generando una correlación negativa, y viceversa para la etiqueta 2 y la frecuencia de 16,5 hz teniendo correlaciones positivas para la misma.

Por último con la idea de que la amplitud máxima de la señal para cada etiqueta va a situarse en la frecuencia de interés se decidió usar está frecuencia como otro feature a extraer en cada canal y luego se unificaron en los distintos estadísticos ya mencionados.



Con estos resultados aunque no sean correlaciones muy grandes, comparado con el resultado de la extracción anterior se cree que se mejora bastante, y se utiliza toda esta combinación de features para los modelos de aprendizaje supervisado.

Escalado

Se realizó de la división del conjunto de datos en train (72%), val (10%) y test (18%). Luego de ésta división se estandarizaron los datos utilizando la función StandardScaler() de Sklearn, para tener una escala común al construir el modelo de aprendizaje automático.

Manejo de desbalanceo

Debido a que dataset presenta un desbalance de casi 3 a 1 para la clase 99 con respecto a las otras clases, se decide utilizar la estrategia de “downsampling” para balancear las clases.

```
99.0    1609
2.0      524
1.0      481
```

Se realizaron pruebas de Downsampling para reducir el desbalance en proporciones 2:1 y Y se obtuvieron mejores métricas con el balanceo 2:1.

```
99.0    1000
2.0      524
1.0      481
```

Por lo cual los puntos que siguen se explican de acuerdo a éstos resultados con el balanceo 2:1.

A) Benchmarking y desarrollo del algoritmo evaluador:

a) Utilice una clasificación aleatoria de los ejemplos para utilizar como benchmark de los resultados posteriores. Este benchmark representa el peor de los desempeños de clasificación posibles.

b) Evalúe el desempeño/rendimiento de este benchmark bajo las métricas seleccionadas en el apartado anterior. Considere repetir este procesamiento algunas veces para obtener un promedio, máximo, mínimo u otro representante de estos resultados, ya que se trata de un proceso completamente aleatorizado.

Para tener un punto de partida con el cual poder comparar los modelos, se eligió como modelo baseline un “Dummy Classifier” de la librería Scikit-learn que genera predicciones random en base a reglas simples, en éste caso se utilizó un random uniforme. Se espera que el resto de los modelos de clasificación tengan un mejor rendimiento que éste baseline.

Como se espera de una estrategia uniforme, la distribución del target es aleatoria y no refleja la verdadera distribución de las etiquetas.

Para tener en cuenta el inconveniente mencionado anteriormente, podemos utilizar la estrategia "estratificada". Las predicciones se generan aleatoriamente, pero se mantiene la distribución de las clases del conjunto de entrenamiento. Al igual que en la estrategia estratificada train_test_split.

Veamos el reporte de métricas obtenido:

```
================================================================================
Model:  DummyClassifier(strategy='stratified')  -  Train  Clasification Report
================================================================================
              precision    recall  f1-score   support

         1.0       0.24      0.24      0.24       346
         2.0       0.29      0.30      0.29       377
        99.0       0.50      0.49      0.49       720

    accuracy                           0.38      1443
   macro avg       0.34      0.34      0.34      1443
weighted avg       0.38      0.38      0.38      1443

================================================================================
Model:  DummyClassifier(strategy='stratified')  -  Val  Clasification Report
================================================================================
              precision    recall  f1-score   support

         1.0       0.15      0.17      0.16        48
         2.0       0.26      0.26      0.26        53
        99.0       0.53      0.50      0.51       100

    accuracy                           0.36       201
   macro avg       0.31      0.31      0.31       201
weighted avg       0.37      0.36      0.36       201

================================================================================
Model:  DummyClassifier(strategy='stratified')  -  Test  Clasification Report
================================================================================
              precision    recall  f1-score   support

         1.0       0.30      0.25      0.27        87
         2.0       0.31      0.29      0.30        94
        99.0       0.50      0.56      0.53       180

    accuracy                           0.42       361
   macro avg       0.37      0.37      0.37       361
weighted avg       0.40      0.42      0.41       361
```

Se observa que tanto las métricas F1-Score de cada clase como el F1 Macro y Acc son bajos (en la mayoría de los casos no superan el 50%).

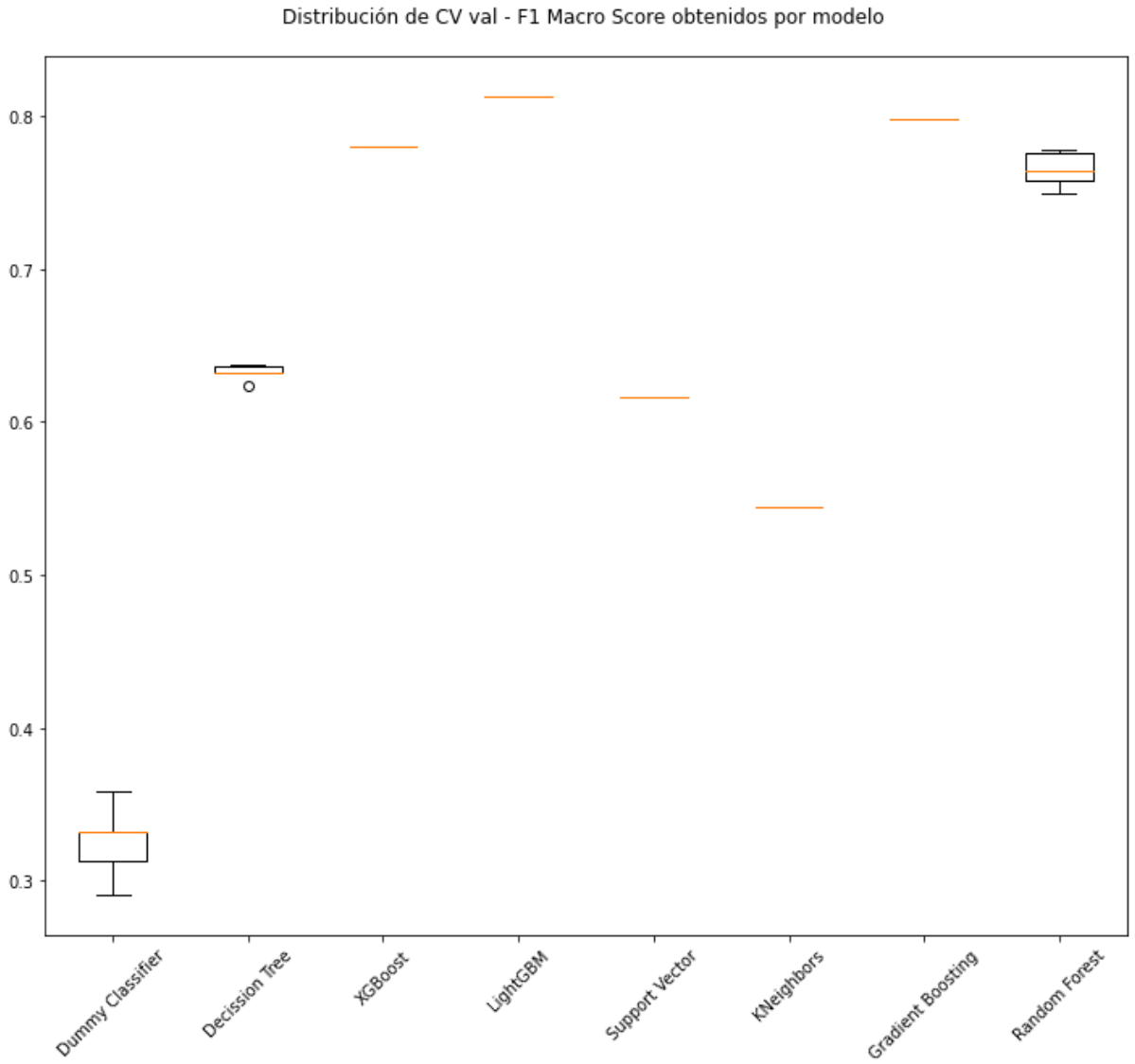
Comparemos las métricas obtenidas por un modelo baseline "Dummy Classifier", y otros modelos con sus parámetros por defecto.

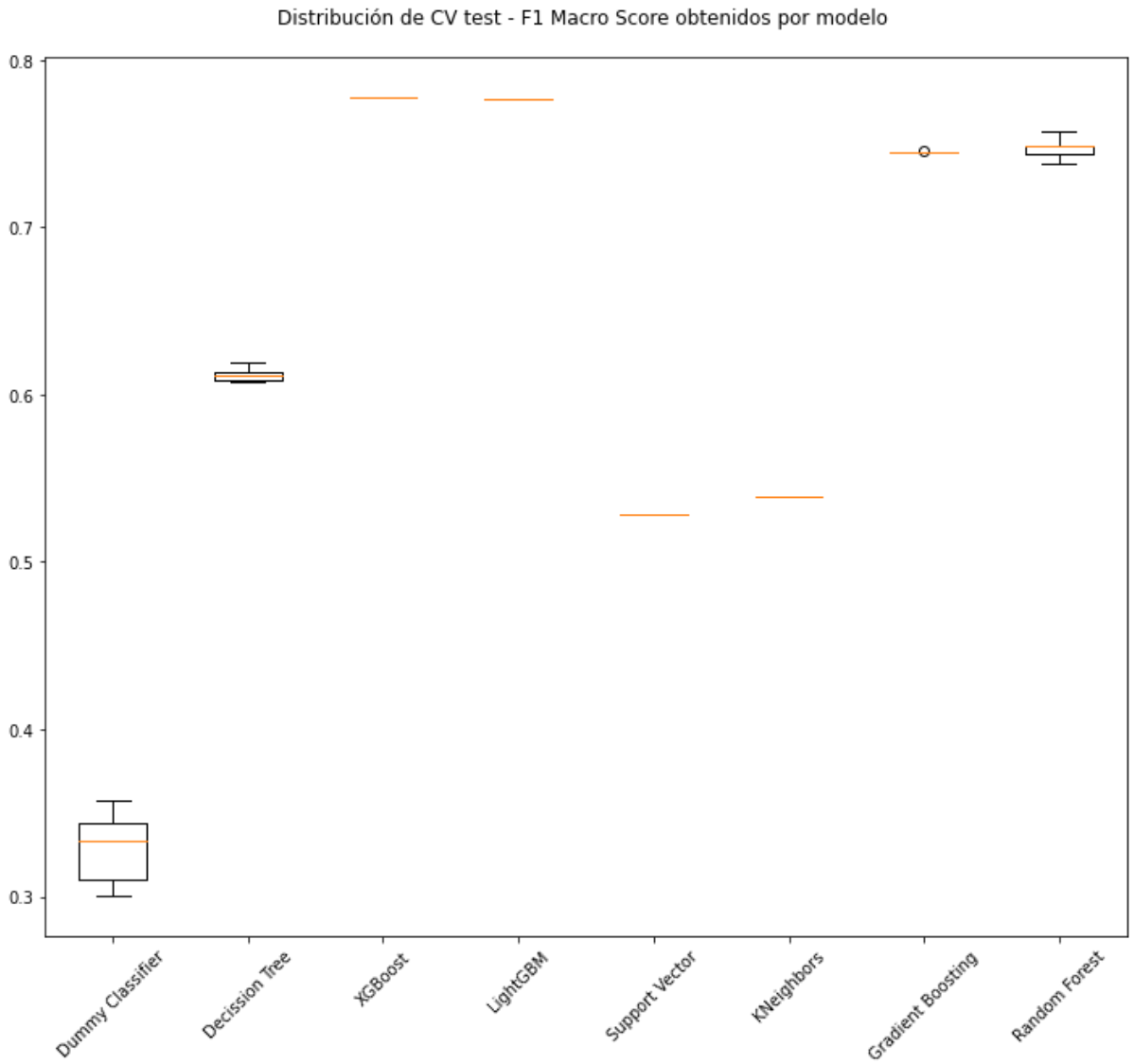
También se desarrollaron diferentes experimentos de modelos de clasificación con parámetros por defecto, que se compararon con el modelo baseline.

Se realizaron 5 iteraciones para cada modelo en donde se buscó disminuir la aleatoriedad del procedimiento sin usar una semilla que sesgue el resultado y se obtuvieron algunas métricas, veamos por ejemplo para el modelo “Gradient Boosting” (para ver el resto de los modelos ir al notebook en gitHub):

```
Train F1 Score Macro:
[0.968, 0.968, 0.968, 0.968, 0.968]

Val F1 Score Macro:
[0.798, 0.798, 0.798, 0.798, 0.798]

Test F1 Score Macro:
[0.745, 0.745, 0.745, 0.745, 0.745]

Train - F1 Score Macro mean: 0.968 -  F1 Score Macro min: 0.968 -  F1 Score Macro max: 0.968
Val - F1 Score Macro mean: 0.798 -  F1 Score Macro min: 0.798 -  F1 Score Macro max: 0.798
Test - F1 Score Macro mean: 0.745 -  F1 Score Macro min: 0.745 -  F1 Score Macro max: 0.745
```

Graficamos diferentes boxplots para la comparación:





A modo de resumen las métricas de cada modelo son:

Modelo	Train		Test		Validation	
	Accuracy	F1 Score macro	Accuracy	F1 Score macro	Accuracy	F1 Score macro
DummyClassifier	0.38	0.34	0.39	0.33	0.39	0.35
DecisionTreeClassifier	1	1	0.62	0.6	0.66	0.65
XGBClassifier	1	1	0.78	0.78	0.79	0.78
LGBMClassifier	1	1	0.78	0.78	0.82	0.81
SVC	0.64	0.59	0.61	0.53	0.67	0.62
KNeighborsClassifier	0.68	0.67	0.56	0.54	0.56	0.54
GradientBoostingClassifier	0.97	0.97	0.76	0.75	0.8	0.8
RandomForestClassifier	1	1	0.76	0.75	0.8	0.78

Vemos que los modelos que presentan mejores métricas F1 Score para cada clase y F1 Score Macro avg son:

- XGBoost
- LGBM
- RandomForest
- Gradient Boosting

B) Búsqueda a grandes rasgos:

a) Utilicen la mayor cantidad de algoritmos de aprendizaje automático supervisado que puedan (mínimo 3). Creen para cada uno un modelo bajo el paradigma de dicho método y entrénelo con el dataset elegido. Opcional: generar curvas de progreso de métricas y funciones de pérdida a lo largo del entrenamiento.
b) No hace falta que la búsqueda de hiper parámetros sea extremadamente detallada, ya que esto se llevará a cabo en el inciso C), sino más bien una exploración general de las tendencias de cada modelo.
c) Si encuentra resultados de las métricas analizadas o gráficos de entrenamiento ploteados que resulten destacables -no sólo porque sean valores más óptimos, sino también posibles casos extraños, situaciones de over y underfitting, etc- muéstrelos y analícelos en este inciso.

Entrenamos los modelos baselines que mejores resultados han dado cambiando los parámetros por defecto:

- XGBoost
- LGBM
- Gradient Boosting
- RandomForest

Modelo XGBoost

Se entrenó el modelo con los siguientes parámetros:

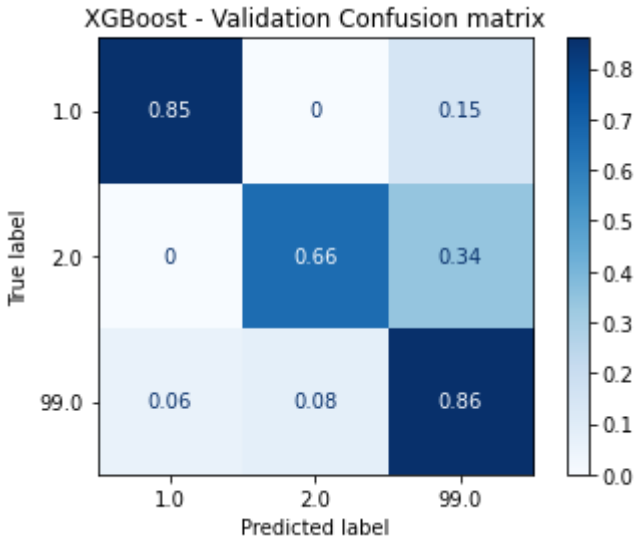
```
params = {
    'alpha': 0.1,
    'colsample_bytree': 0.5,
    'gamma': 0.2,
    'learning_rate': 0.2,
    'max_depth': 3,
    'min_child_weight': 1,
    'subsample': 1.0
}
```

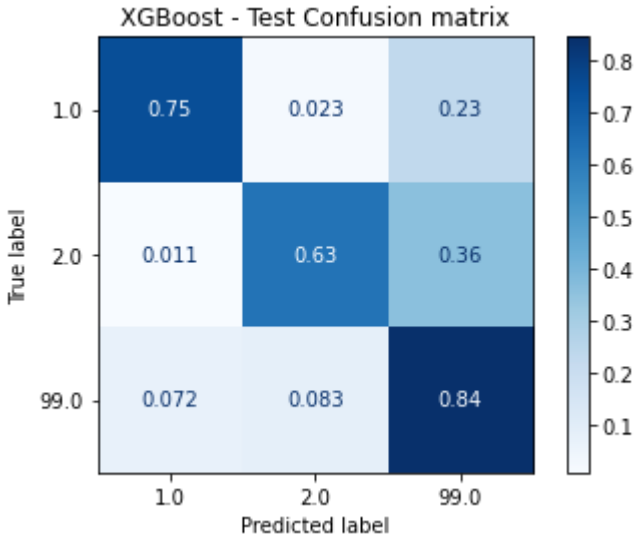
Se obtuvieron las siguientes métricas para los conjuntos de Validación y Test:

```
============================================================
Model:  XGBClassifier  -  Val  Clasification Report
============================================================
              precision    recall  f1-score   support

         1.0       0.87      0.85      0.86        48
         2.0       0.81      0.66      0.73        53
        99.0       0.77      0.86      0.82       100

    accuracy                           0.81       201
   macro avg       0.82      0.79      0.80       201
weighted avg       0.81      0.81      0.80       201

============================================================
Model:  XGBClassifier  -  Test  Clasification Report
============================================================
              precision    recall  f1-score   support

         1.0       0.82      0.75      0.78        87
         2.0       0.78      0.63      0.69        94
        99.0       0.74      0.84      0.79       180

    accuracy                           0.76       361
   macro avg       0.78      0.74      0.75       361
weighted avg       0.77      0.76      0.76       361
```

Nota: Se omiten en los gráficos las métricas del conjunto de Train sólo por simplicidad, pueden verse los gráficos completos en gitHub.

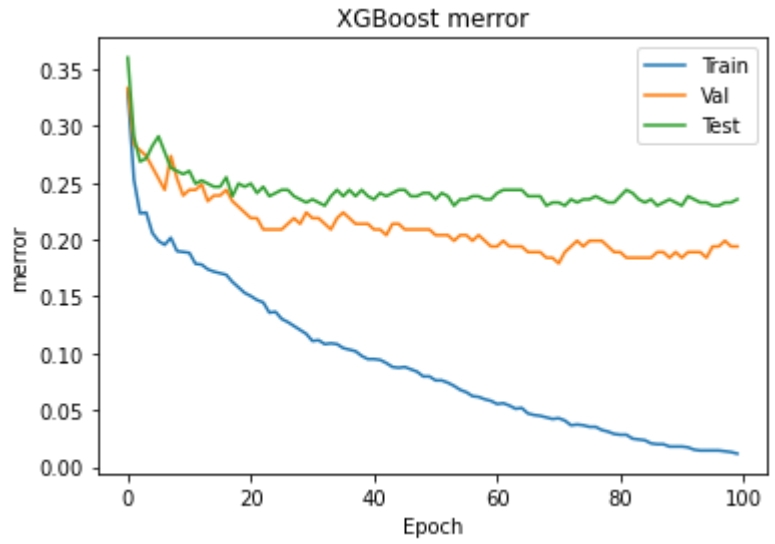
Matrices de confusión

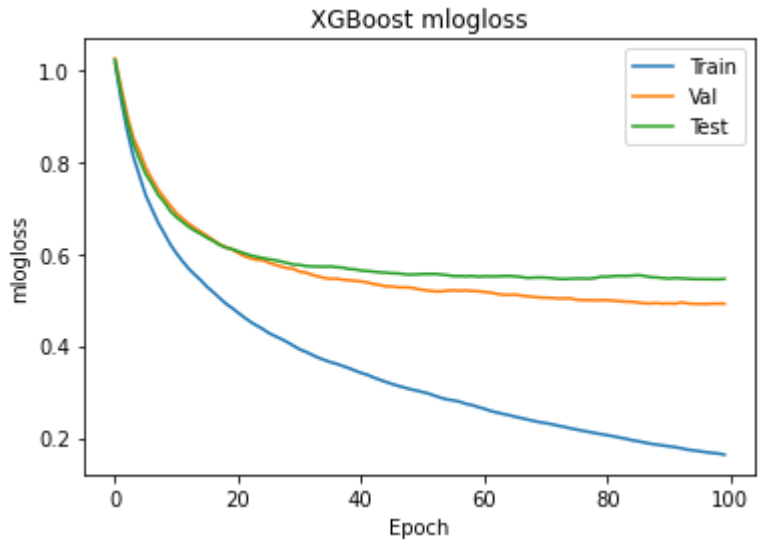




Se puede observar en las métricas que el modelo XGBoost presenta un F1 Score menor para la clase 2, con respecto a las clases 1 y 99. Ésto puede deberse a que para las frecuencias 16,5 y 33 (correspondiente al label 2), no se presentan picos muy marcados cuando se visualizan las frecuencias gráficamente.
Esto mismo puede observarse en las diagonales de las matrices de confusión (correspondiente a la métrica Recall). También se visualiza en la matriz de confusión que ...ca de un 35% de las etiquetas que originalmente eran '2', se predijeron como '99', el resto de los porcentajes "no predichos correctamente" está por debajo del 20% y en la gran mayoría muy cercano a cero. Por lo cual es un buen modelo candidato para elegirlo como el mejor.

Funciones de pérdida





Si las predicciones se desvían demasiado de los resultados reales, la función de pérdida del modelo arrojaría un número muy grande.
Al revisar los gráficos de pérdida, parece que hay una oportunidad para detener prematuramente el entrenamiento (‘learning early’), tal vez en algún lugar alrededor de la época 40 a la época 60, para limitar el overfitting (un “equilibrio” entre dónde se minimiza el error tanto en el conjunto de val como en el de test y menor cantidad de ‘epochs’).

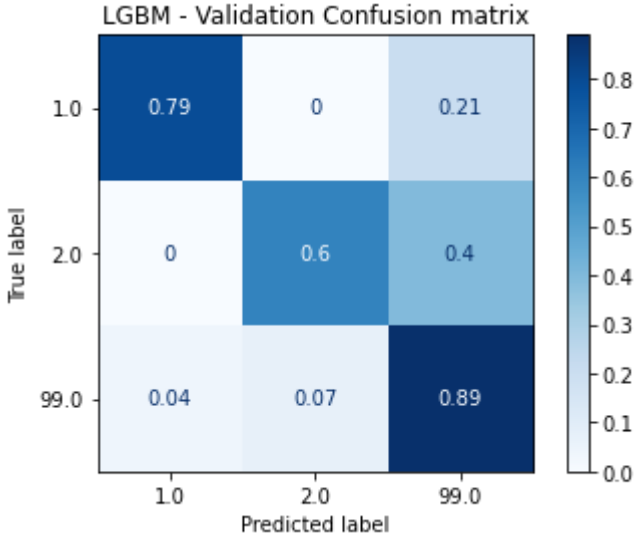
Modelo LGBM

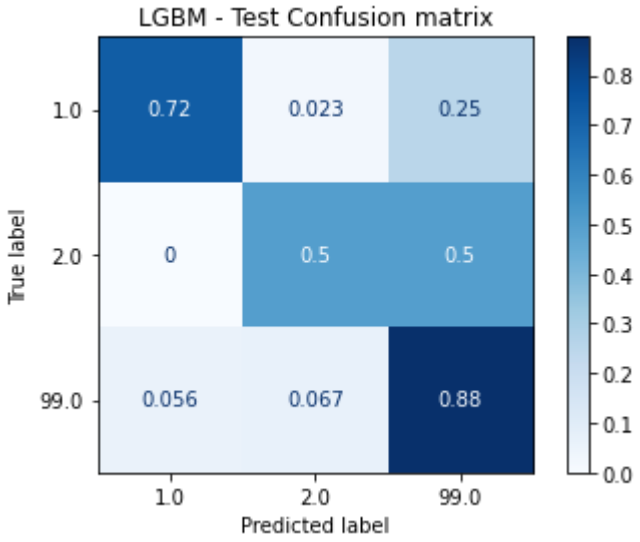
Se entrenó el modelo con los siguientes parámetros:

```
params = {
    'is_unbalance': True,
    'learning_rate': 0.01,
    'objective': 'multiclass'
}
```

```
============================================================
Model:  LGBMClassifier  -  Val  Clasification Report
============================================================
              precision    recall  f1-score   support

         1.0       0.90      0.79      0.84        48
         2.0       0.82      0.60      0.70        53
        99.0       0.74      0.89      0.81       100

    accuracy                           0.79       201
   macro avg       0.82      0.76      0.78       201
weighted avg       0.80      0.79      0.79       201

============================================================
Model:  LGBMClassifier  -  Test  Clasification Report
============================================================
              precision    recall  f1-score   support

         1.0       0.86      0.72      0.79        87
         2.0       0.77      0.50      0.61        94
        99.0       0.70      0.88      0.78       180

    accuracy                           0.74       361
   macro avg       0.78      0.70      0.72       361
weighted avg       0.76      0.74      0.73       361
```

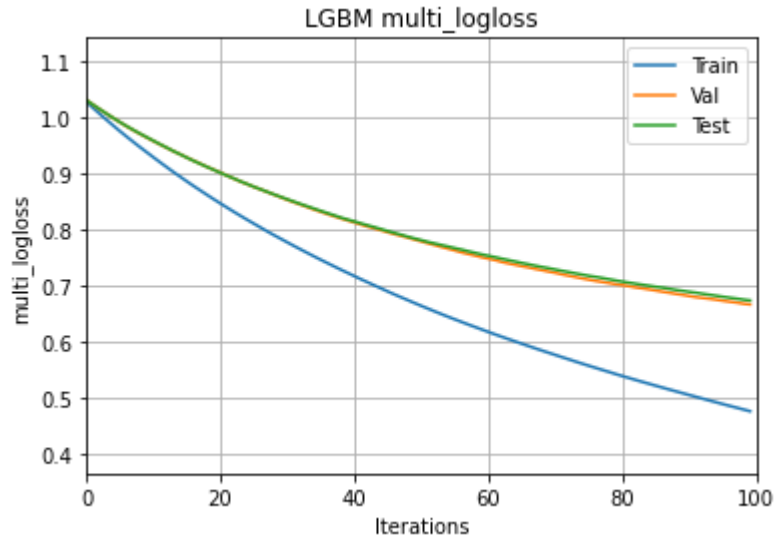
Matrices de confusión





Para el modelo LGBM también se presenta un F1 Score menor para la clase 2 con respecto a las otras clases, tanto para val como para test.
Y un menor F1 macro avg con respecto al modelo XGBoost.

Funciones de pérdida



En éste caso, la función de pérdida del modelo XGBoost no se estabiliza, y se minimiza mientras más iteraciones se ejecutan.

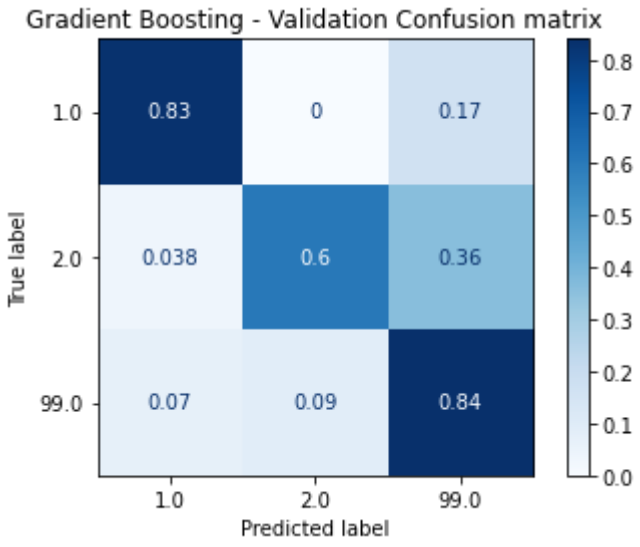
Modelo Gradient Boosting

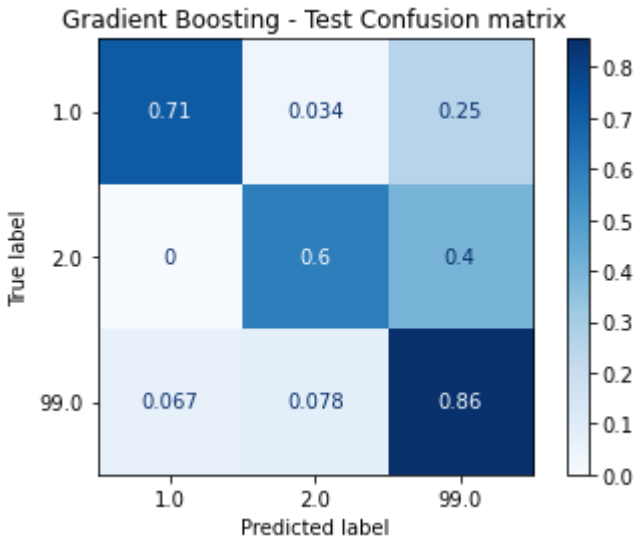
Se entrenó el modelo con los siguientes parámetros:

```
params = {
    'n_estimators': 100,
    'learning_rate': 0.5,
    'max_features' : 'sqrt'
}
```

```
==========================================================================
Model:  GradientBoostingClassifier  -  Val  Clasification Report
==========================================================================
              precision    recall  f1-score   support

         1.0       0.82      0.83      0.82        48
         2.0       0.78      0.60      0.68        53
        99.0       0.76      0.84      0.80       100

    accuracy                           0.78       201
   macro avg       0.78      0.76      0.77       201
weighted avg       0.78      0.78      0.77       201

==========================================================================
Model:  GradientBoostingClassifier  -  Test  Clasification Report
==========================================================================
              precision    recall  f1-score   support

         1.0       0.84      0.71      0.77        87
         2.0       0.77      0.60      0.67        94
        99.0       0.72      0.86      0.78       180

    accuracy                           0.75       361
   macro avg       0.77      0.72      0.74       361
weighted avg       0.76      0.75      0.75       361
```

Matrices de confusión





Para el modelo Gradient Boosting también se presenta un F1 Score menor para la clase 2 con respecto a las otras clases, tanto para val como para test.
Y un menor F1 macro avg con respecto al modelo XGBoost, pero similar al LGBM.

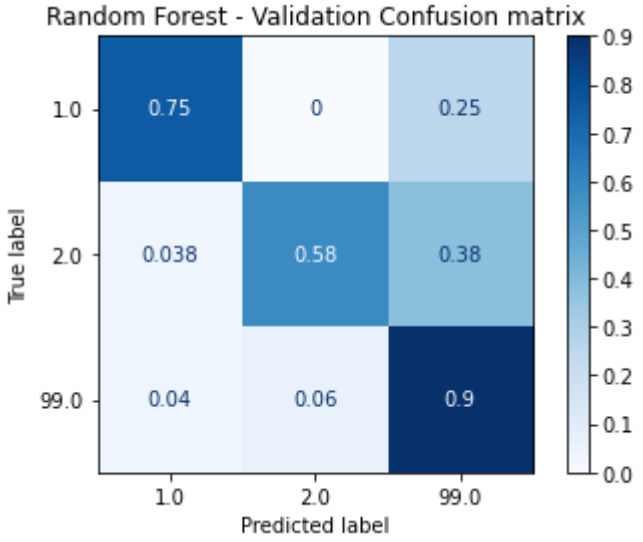
Modelo Random Forest

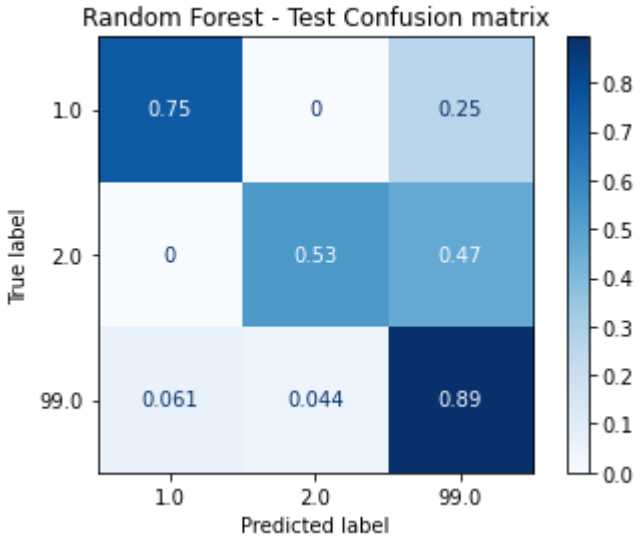
Se entrenó el modelo con los siguientes parámetros:

```
params = {
    'n_estimators': 300,
    'max_depth'   : 15,
    'criterion'   : 'gini',
    'max_features': 0.25,
    'min_samples_split': 4
}
```

```
============================================================
Model:  RandomForestClassifier  -  Val  Clasification Report
============================================================
              precision    recall  f1-score   support

         1.0       0.86      0.75      0.80        48
         2.0       0.84      0.58      0.69        53
        99.0       0.74      0.90      0.81       100

    accuracy                           0.78       201
   macro avg       0.81      0.74      0.77       201
weighted avg       0.79      0.78      0.78       201

============================================================
Model:  RandomForestClassifier  -  Test  Clasification Report
============================================================
              precision    recall  f1-score   support

         1.0       0.86      0.75      0.80        87
         2.0       0.86      0.53      0.66        94
        99.0       0.71      0.89      0.79       180

    accuracy                           0.76       361
   macro avg       0.81      0.72      0.75       361
weighted avg       0.78      0.76      0.76       361
```

Matrices de confusión





Para el modelo Random Forest también se presenta un F1 Score menor para la clase 2 con respecto a las otras clases, tanto para val como para test.

C) Búsqueda puntualizada:

a) Con el modelo que presente mejores resultados, lleve a cabo una búsqueda ahora sí más detallista, variando los hiper parámetros y funciones de costo. Si estos métodos permiten variar la cantidad de instancias/épocas de entrenamiento, analice lo que sucede cuando varía las duraciones de entrenamiento.
b) Reporte los resultados obtenidos y seleccione el set completo de configuraciones que mejor resuelven, bajo su criterio, nuestro problema de clasificación.

Debido a que en el punto anterior, para los 4 modelos elegidos no tenemos grandes diferencias en las métricas, realizaremos ajuste de hiper parámetros para evaluarlos nuevamente.

Parámetros utilizados para el ajuste:

XGBoost

```
param_grid = {
    "learning_rate": [0.1, 0.2, 0.3] ,
    "max_depth": [ 3, 6, 10],
    'subsample': [0.6, 0.8, 1],
    'eval_metric': ['merror', 'mlogloss']
}
```

LGBM

```
param_grid = {
    'boosting_type': ['gbdt','dart','goss'],
    'n_estimators': [50, 100, 150],
    'max_depth': [-1, 3], #, 5, 10],
    'learning_rate' : [-1, 0.01, 0.1],
}
```

Gradient Boosting

```
param_grid = {
    'n_estimators': [50,100,150],
    'loss': ['deviance', 'exponential'],
    'criterion': ['friedman_mse', 'squared_error', 'mse', 'mae'],
}
```

Random Forest

```
param_grid = {
    'n_estimators': [100,150,200],
    'max_features': [5, 9,'log2','auto'],
    'max_depth'   : [None, 10, 20],
    'criterion'   : ['gini', 'entropy']
}
```

Modelo	Train		Validation		Test	
	Accuracy	FI Score Macro Avg	Accuracy	FI Score Macro Avg	Accuracy	FI Score Macro Avg
XGBoost	1	1	0.81	0.80	0.77	0.76
LGBM	1	1	0.80	0.80	0.79	0.78
GradientBoosting	0.97	0.97	0.80	0.80	0.76	0.75
Random Forest	1	1	0.79	0.78	0.77	0.76

Como resultado de las pruebas de ajuste por hiper parámetros, obtuvimos mejores métricas en el modelo “Light Gradient Boosting Machine (LGBM)” (tanto en F1 Score Macro, ACC como para el F1 Score de cada clase).

LGBM es un algoritmo que utiliza la técnica Gradient Boosting para combinar árboles de decisión que individualmente tienen poco poder de predicción pero sumados dan resultados excelentes.

Éste modelo presenta algunas ventajas como:
- una mayor velocidad de entrenamiento y
- una alta eficiencia en el uso de memoria.

Por lo cual éste modelo es un candidato para la resolución del problema con un F1 Score Macro aproximado de 0.80.

Conclusión

Se logró resolver la problemática del experimento planteado, obteniendo un modelo que permite identificar y clasificar la respuesta de la señales cerebrales (SSVEP) a un estímulo luminoso de dos frecuencias diferentes con un alto porcentaje de precisión.

Sin olvidar el objetivo principal del problema el cual logra generar una aplicación para que distintas personas puedan comunicarse. A su vez se pueden crear distintas aplicaciones de toma de decisiones con la misma tecnología utilizando el procesamiento de señales EEG.

Propuesta de mejora

Observando la variabilidad entre los valores de voltaje de las muestras entre individuos y entre sesiones de un mismo individuo, se realizaron pruebas de modelos particulares para cada sujeto logrando mejorar los resultados.

Por lo que se propone como una futura mejora tomar como base el modelo logrado en esta primera etapa y realizar un fine tuning con mayor cantidad de ensayos de cada individuo para lograr incrementar el porcentaje de precisión del modelo.