UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

PDDetection Aplicación de técnicas de minería de datos para la detección de la enfermedad del Parkinson

Presentado por Adrián Arnaiz Rodríguez
en Universidad de Burgos — 26 de junio de 2019
Tutor: José Francisco Díez Pastor y Cesar Ignacio García Osorio

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Adrián Arnaiz Rodríguez, con DNI 71306880A, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 26 de junio de 2019

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
Dr. D. José Francisco Díez Pastor	Dr. D. Cesar Ignacio García Osorio

Resumen

La enfermedad del Parkinson causa discapacidades en el habla a más del 90 % de los pacientes. Esta enfermedad causa trastornos del habla como pueden ser monotonía, volumen monótono, imprecisión en la articulación y otros síntomas. Hay numerosos estudios que abordan el problema de desarrollar aplicaciones de ayuda de evaluación del Parkinson.

En esta investigación, a partir de una base de datos de audios adquirida, tratamos de realizar todo el proceso de minería de datos que nos permita llegar a construir un clasificador. Este clasificador distingue entre si la persona de un audio tiene Parkinson o no. Para ello, se abordarán diferentes tipos de características extraídas de los audios (características físicas y más novedosas) con las que se realizarán diferentes experimentos.

Por ello, se realizará una aplicación de ayuda al facultativo médico que, usando el clasificador desarrollado en la investigación, sea capaz de agilizar el proceso de diagnosis de la enfermedad del Parkinson, además de reducir el riesgo de fallo en la misma.

Descriptores

Minería de datos, Clasificador, Análisis del discurso, Parkinson, Aprendizaje automático, Aprendizaje profundo, Procesamiento de audio, Predicción enfermedades.

Abstract

Parkinson's Disease causes speech impairments in more than 90 % of patients. This disease causes different disorders like monotony, monotonous volume, inaccuracy articulation and other symptoms. There are numerous studies that have addressed the problem of developing applications for helping the diagnosis of Parkinson's.

In this research, with a database of acquired audios, we try to realice the whole data mining process, building a good classifier as the main achieve. This classifier distinguishes between if the person of an audio has Parkinson's or not. For this purpose, we address different experiments, which are carried out with different types of characteristics extracted from the audios (physical characteristics and newer ones).

Therefore, we will make an aid application addressed to medical professional who, using the classifier developed in the research, will be able to speed up the diagnosis process of Parkinson's disease, in addition to reducing the risk of failure in it.

Keywords

Data mining, Clasificator, Speech analysis, Parkinson, Machine learning, Deep learning, Audio processing, Disease prediction.

Índice general

Índice de figuras

Índice de tablas

Introducción

1.1. Introducción al proyecto

En enfermedades como la depresión y las neurodegenerativas es requerido un seguimiento cercano de la enfermedad para ver su evolución. Actualmente es necesario que un facultativo médico se desplace a los domicilios de los pacientes y realice una serie de test y entrevistas estructuradas para evaluar la progresión de la enfermedad (depresión, alzheimer, **Parkinson**, etc). En líneas generales se pretende que el sistema propuesto sea capaz de determinar esa diagnosis solamente a partir de grabaciones de voz. En la actualidad, la monitorización de los síntomas y la diagnosis de enfermedades son costosos y, logísticamente, son inconveniente para el paciente y el personal clínico, lo que también dificulta el reclutamiento para futuros ensayos clínicos a gran escala. Por ello, todo lo que signifique reducir complejidad y costes en este ámbito ayudará tanto a pacientes como a profesionales.

El deterioro del habla del Parkinson se ha estudiado utilizando grabaciones de voz o conjuntos de datos de voz. Los conjuntos de datos de voz son características extraídas de las grabaciones de voz. Como en el análisis de voz patológica, la lectura “The Rainbow Passage"también se usa en estudios de PD (Parkinson Detection). Sin embargo, las grabaciones de vocales sostenidas (pronunciación continua de una vocal durante un número determinado de segundos) son cada vez más populares como la entrada analítica. Holmes [17], estudió las características de la voz para etapas tempranas y posteriores de PD en comparación con personas sanas usando 4 segundos de vocal sostenida ’a’ y grabaciones de monólogos de 1 minuto. Holmes utilizó las grabaciones para extraer ocho características para el análisis acústico a partir de la mitad de la vocal sostenida y el canto a escala. Estas características incluyen la frecuencia fundamental y su

desviación estándar, la intensidad media y su desviación estándar, el rango de frecuencia de frecuencia máxima, la fluctuación de fase, el brillo y la relación de ruido a armónicos. Por ejemplo, en este estudio Holmes descubrió, entre otros muchos hallazgos, que en comparación con los controles y los datos normativos publicados previamente, las voces de los pacientes con PD [1] en etapa temprana y tardía se caracterizaron perceptualmente por la variabilidad de tono y volumen limitados, la transpirabilidad, la dureza y el volumen reducido.

Desde entonces, un número cada vez mayor de investigadores han estado utilizando la vocal sostenida 'a' para estudiar la disfonía de la enfermedad del Parkinson, incluidos los investigadores Little y Tsanas 2010. En esta investigación realizada por Little y Tsanas en el 2010 [33], demostraron la replicación rápida y remota de la evaluación UPDRS (escala de calificación de la enfermedad de Parkinson, véase sección 3.4) con una precisión clínicamente útil (aproximadamente 7.5 puntos UPDRS diferencia de las estimaciones de los clínicos), utilizando solo pruebas de voz simples, autoadministradas y no invasivas.

Por lo tanto, nuestro proyecto consistirá en lo siguiente. Debemos conseguir una base de datos de audios. Estos audios se deberán realizar las entrevistas estructuradas, en las cuales sus respuestas serán grabadas en el mismo momento de la realización. Las grabaciones se identificarán con el identificador del paciente, así como con datos adicionales, como edad y sexo, del paciente. Las grabaciones incluirán grabaciones estandar, donde siempre se recite la misma frase, la misma palabra o la misma vocal. Concretamente, la base de datos con la que trabajaremos se describe en [24]. Los audios de las grabaciones se procesarán para obtener diferentes atributos y características de la voz, como por ejemplo los usados en el estudio de Orozc et al. [25], o los anteriormente comentados que se usaron en el estudio de Holmes. En nuestro caso se obtendrán los indicadores tono, cadencia, volumen y ritmo para su posterior análisis. A partir de ahí se construirá y entrenará una serie de modelos para predecir si la persona de un audio tiene Parkinson o no. Se realizarán una serie de diferentes experimentos con el objetivo de crear el mejor modelo posible. Este modelo correctamente construido se utilizará para evaluar la enfermedad de un paciente. Con ello conseguiremos avanzar en las investigaciones relativas a las enfermedades neurodegenerativas y su diagnóstico, y también realizar un producto, aplicación, que ayude e intente mejorar la calidad de vida tanto a pacientes como a trabajadores sanitarios.

[1] Parkinson Disease

Objetivos del proyecto

2.1. Objetivos generales

- Investigar y condensar el estado del arte de la investigación sobre la detección del Parkison a través de la voz resumiendo los artículos científicos más importantes, identificando las tecnologías, herramientas y procesos actuales utilizadas en ellos, realizando taxonomías, etc.
- Recopilación de bases de datos adecuadas para la investigación, tanto para uso en este proyecto como para su uso en proyectos posteriores, cerciorando que son *datasets* de audios correctos para las tareas necesitadas (i.e. audios etiquetados).
- Realización de un estudio comparativo en cuanto a la utilización de diferentes modelos de clasificación y conjuntos de características extraídas de los audios. Se compararán resultados, tanto entre los diferentes experimentos que nosotros realicemos, como entre nuestros mejores experimentos y resultados científicos de anteriores experimentos (publicados en artículos científicos).
- Aportación de una nueva perspectiva a este campo de investigación: extracción de las características de los audios mediante *Deep Learning*.
- Finalmente, utilizar todo lo descrito anteriormente para realizar una aplicación la cual permita la monitorización de la diagnosis de la enfermedad del Parkinson a través de la voz. La aplicación será capaz de discernir, mediante un clasificador, si la persona de un audio subido a la aplicación web tiene Parkinson o no. Esta aplicación será un ejemplo de como poder plasmar los resultados de esta investigación en una herramienta funcional.

2.2. Objetivos técnicos

- Desarrollar un algoritmo, cuya implementación en Python permita la extracción de características de los audios de manera adecuada (envolver la extracción de características que realizan Disvoice y VGGish en clases).
- Desarrollar una aplicación de escritorio para la etapa de explotación, que muestre como se pueden plasmar los resultados de la investigación para en una herramienta funcional.
- Utilizar las herramientas más adecuadas para la realización del estudio comparativo: valoraremos Python y sus bibliotecas, Weka...
- Utilizar un sistema de control de versiones Git utilizando para ello la plataforma Github. Utilizar un cliente Git para el trabajo local para lo cual utilizaremos TortoiseGit.
- Utilizar la metodología Scrum en la elaboración del proyecto y concretamente la herramienta ZenHub (como extensión de Github) para la ayuda en la gestión de proyectos.

Conceptos teóricos

Este proyecto abordará diferentes temas dentro del campo conocido como *Data Mining*. A su vez se tratan temas sobre el análisis del discurso desde una perspectiva física. Se explicarán conceptos dentro de ambos campos para la correcta comprensión del proyecto.

3.1. Minería de datos y aprendizaje automático

La **minería de datos** es un campo del conocimiento dentro de las ciencias de computación cuyo objetivo general es analizar grandes volúmenes de datos para extraer conocimiento de ellos. Se basa en el concepto de que la sociedad, sobre todo actualmente, produce una gran cantidad de datos y de diferente origen. Sin embargo, extraer conocimiento de los datos no es un proceso tan trivial. No es posible sacar la información de los datos en crudo. Por ello se necesitan diferentes métodos y técnicas que nos permitan hacer esa tarea, como son las técnicas de **aprendizaje automático** [39]. Este proceso de divide en varias fases.

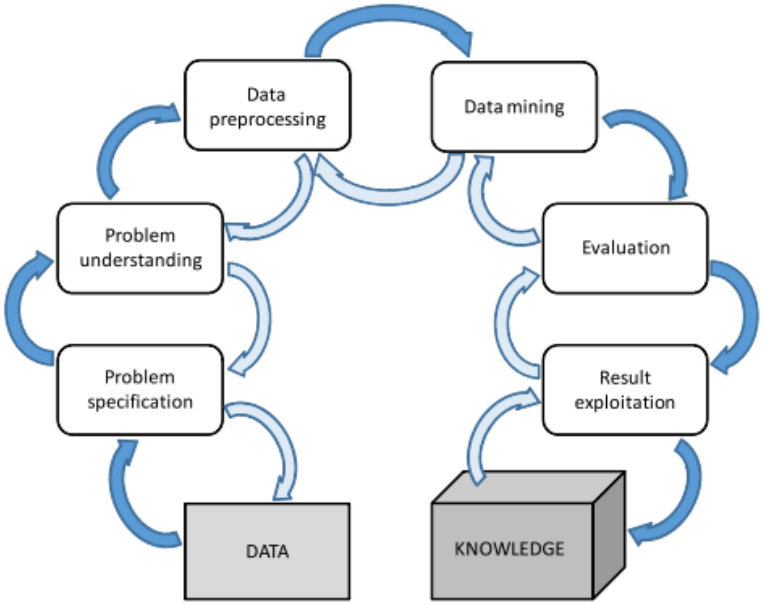


Figura 3.1: Fases en la extracción de conocimiento de un conjunto de datos. [10]

Podemos hablar de que la minería de datos es el análisis de un conjunto grande de datos para la obtención de conocimiento de ellos, ya sea relaciones entre esos datos o patrones ocultos, que nos permita obtener una ventaja competitiva o un conocimiento novedoso. Esto está relacionado con el proyecto en cuanto a que queremos obtener conocimiento de la relación voz-enfermedad del Parkinson a través un gran conjunto de audios. A continuación, definiremos varios conceptos dentro de este campo que son manejados en el proyecto.

Pre-procesamiento de datos

Como hemos comentado anteriormente, los datos no pueden ser tratados de manera directa, necesitan ser tratados en una etapa anterior al descubrimiento de información: pre-procesamiento. Esto es indispensable en la fase de pre-procesamiento, transformar los datos en bruto a otro conjunto de datos que pueda servir para ser procesado por los algoritmos necesarios. Específicamente en nuestro caso, no podemos obtener conocimiento de el conjunto de audios en bruto, estos audios deberán ser tratados para extraer un conjunto de características numéricas de cada uno de ellos y poder seguir avanzando a partir de ese punto. El conjunto de características extraídas

de los audios será en su mayoría propiedades físicas (3.6) o características extraídas con bibliotecas de *Deep Learning*. Como nuestro proyecto se basará en aprendizaje supervisado, también se llevará a cabo en esta etapa el etiquetado de los datos.

Otra parte importante, aunque no indispensable, dentro de esta etapa es la fase de selección de características. Cuando tenemos un gran número de atributos para cada instancia puede ser recomendable reducir su dimensionalidad. Una manera de reducir su dimensionalidad es a través de la selección de atributos manual, mediante un análisis estadístico [10]. Un ejemplo de su uso puede verse en [34], donde varios de ellos son utilizados para definir un subconjunto de 10 características de cada audio dentro de las 132 medidas que se obtenían inicialmente en cada uno de ellos. En nuestro caso, no se realiza ese proceso manualmente. Por un lado, hemos realizado algunos experimentos donde hemos proporcionado todos los atributos de cada audio al algoritmo de clasificación y trasladando a éste la tarea de discernir cuales serán más importantes. Por otro, hemos realizado experimentos donde, en el proceso, se utilizan diferentes métodos selectores de atributos como *Select K Best* o *Variance Treshold* (ver sección 3.1).

Aprendizaje automático

Es el campo de las ciencias de la computación enfocado a que los dispositivos *aprendan* por ellos mismos sin haber sido explícitamente programados para ello. Esta es una rama de la inteligencia artificial. Tras obtener un conjunto válido de características se aplican algoritmos de aprendizaje automático. Estos algoritmos son los encargados de detectar patrones en los datos. Estos algoritmos tienen un fuerte componente matemático y estadístico ya que, a través de métodos de estos campos del conocimientos, se extrae la información de los datos. Se divide en:

- **Aprendizaje supervisado**: para cada instancia del conjunto de datos tendremos tanto la entrada como la salida deseada. El objetivo será predecir la salida correcta para una nueva entrada. En nuestro caso la entrada serían las características de los audios y la salida si la persona de ese audio tiene parkinson (clasificación) o en qué nivel lo tiene (regresión) [2].

[2]Estos ejemplos ilustran la diferencia entre clasificación, donde hay que predecir una etiqueta de entre un conjunto finito de etiquetas posibles, y regresión, donde el valor que se predice es una magnitud continua de un conjunto con un número de valores potencialmente infinito.

- **Aprendizaje no supervisado**: únicamente se tienen los datos de entrada y no se sabe nada acerca de su salida o clase. Habitualmente estos algoritmos son utilizados para la detección de clases que permitan separar los datos en diferentes grupos. Nosotros no utilizaremos este tipo en nuestro proyecto.

Generalización

El objetivo de los algoritmos de aprendizaje supervisado no es lograr dar salidas correctas para los ejemplos de datos que tenemos, sino lograr clasificar o dar la salida correcta para un nuevo ejemplo que nos llegue. Se puede decir que el objetivo del aprendizaje automático es en realidad es obtener conocimiento a partir de un gran conjunto de datos el cual sea extrapolable para todos los datos dentro de esa misma situación (de ese mismo problema). El problema de ajustarse demasiado a los datos de entrenamiento perdiendo así capacidad de generalización se llama sobreajuste. Esto ocurre cuando se crean modelos demasiado complejos que se ajustan a los datos de entrenamiento al 100 %. Al ocurrir sobreajuste, corremos el riesgo de clasificar mal nuevos datos. Si además este sobreajuste a los datos de entrenamiento se combina con la aparición de datos ruidosos en nuestro dataset, se perderá aún más capacidad de generalización. Por ello es importante tener métodos de evaluación de modelos que nos permitan medir cuánto de bueno es nuestro modelo a la hora de generalizar. Aquí aparece el concepto **Cross-Validation**.

K-fold Cross Validation [29]

La validación cruzada es un método para la evaluación y comparación de algoritmos. Se basa en descomponer el conjunto de datos en 2 subconjuntos, entrenamiento y test, con el objetivo de evaluar que tal generaliza nuestro modelo. Sin embargo, de esta manera estamos perdiendo información de nuestros datos, ya que el conjunto que utilizamos para test nunca es usado para entrenar. La solución a este problema es la validación cruzada k-fold.

En **k-fold cross-validation** el conjunto entero de datos se divide en k conjuntos del mismo tamaño cada uno. A partir de ahí se itera sobre los subconjuntos k veces utilizando cada vez $k-1$ conjuntos para entrenar y 1 conjunto de test. Para cada una de las iteraciones se guarda la performance y finalmente se hace la media para obtener una performance total. El número idóneo de k es 10 según [29] y así es utilizado el 10-fold en [25].

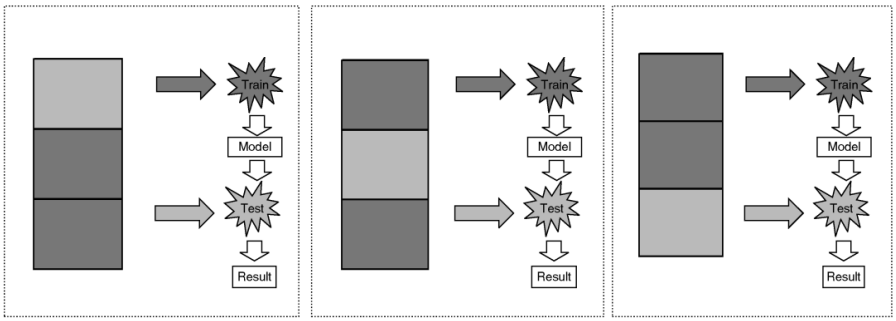


Figura 3.2: Ejemplo 3-fold cross-validation extraído de [29].

Cabe destacar que [29] explica que para *k-fold cross-validation* se debe realizar estratificación. **Estratificación** es el proceso por el cual se asegura que una de las k particiones de los datos total es una buena representación de los datos totales. Por ejemplo, en nuestro caso tenemos 50 % de personas con Parkinson y 50 % de personas sanas. Esto significa que para estratificar de manera correcta, cada partición deberá tener la mitad de instancias de cada clase. Cada partición k deberá tener aproximadamente la misma proporción de clases que el conjunto total.

Nested Cross Validation

A menudo, queremos ajustar los parámetros de un modelo (por ejemplo, C en SVM). Es decir, queremos encontrar el valor de un parámetro que maximice el rendimiento de nuestro clasificador. Sin embargo, Cawley y Talbot en [4], aseguran que si utilizamos la partición de test de los datos, tanto para ajustar los parámetros, como para evaluar el modelo, corremos el riesgo de sesgar de forma optimista las evaluaciones del modelo. Por esta razón, si un conjunto de pruebas se utiliza para seleccionar parámetros del modelo, entonces necesitamos un conjunto de pruebas diferente para obtener una evaluación imparcial de ese modelo seleccionado.

Una forma de superar este problema es tener validaciones cruzadas anidadas (*Nested Cross Validation*). Primero, se usa una validación cruzada interna para ajustar los parámetros y seleccionar el mejor modelo (o parámetros del modelo). En segundo lugar, se utiliza una validación de cruz externa para evaluar el modelo seleccionado por la validación de cruz interna. Ver Figura 3.3.

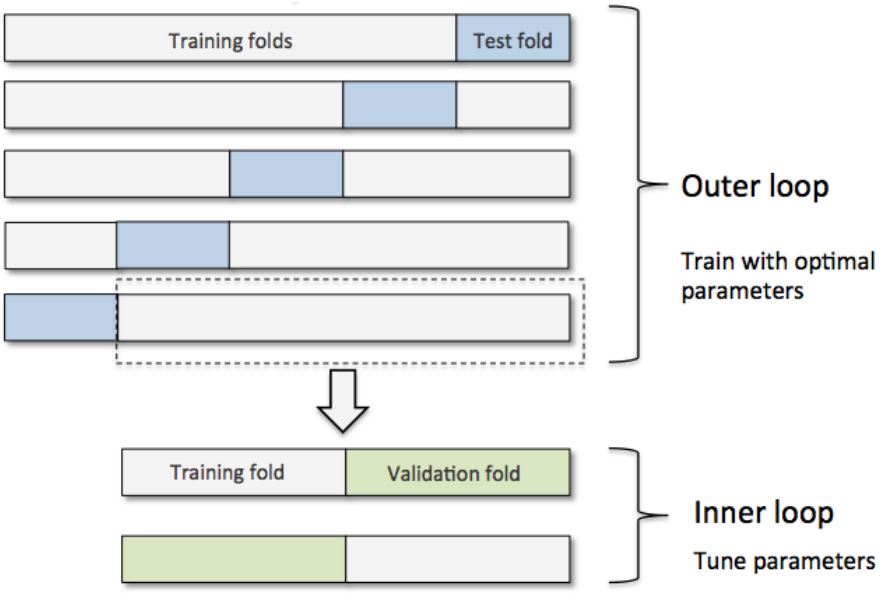


Figura 3.3: Validación cruzada anidada

Algoritmos de aprendizaje automático

A continuación explicaremos varios algoritmos de los utilizados en este proyecto.

SVM [7]

Este algoritmo de aprendizaje supervisado es llamado como máquinas de vector de soporte (SVM). Es un clasificador lineal basado en el concepto de margen máximo. Se utiliza tanto para regresión como para clasificación. Para clasificación su principio fundamental es separar dos clases en el conjunto de datos utilizando un hiperplano. El hiperplano utilizado como separador de las clases será el que maximice el margen entre las clases. Como sabemos, los problemas del mundo real tienen complicaciones como que no sean linealmente separables, que las clases estén solapadas, que haya más de 2 clases de datos, etc.

Para solucionar el problema de la existencia de más de 2 clases, se utiliza un SVM por cada una de las clases. Sin embargo, para el problemas de separar dos clases que no son linealmente separables, utilizamos el **kernel trick**. Esta característica del algoritmo es importante para poder separar este tipo de datos. Estas funciones kernel son utilizadas para aumentar la dimensionalidad de los datos de entrada, es decir, transformar el espacio de

entrada. De esta manera dividiendo linealmente el espacio transformado con el hiperplano pueden resolverse el problema de la separación lineal. Existen diferentes ejemplos de funciones kernel, entre las más habituales están el kernel, polinomial, funciones de base radial, sigmoide y kernel Gaussiano que es utilizado en [34] y [25].

Random Forest [3]

Es un método de **ensemble** (métodos combinados) basado en los árboles de predicción, la combinación de la selección aleatoria de atributos y el *bagging*. El proceso es el siguiente: se construirán de manera independiente un conjunto de árboles, los cuales serán todos diferentes, debido a que están construidos con *bagging* y aleatoriedad de atributos, a la hora de predecir una nueva instancia, cada uno hace su predicción y, teniendo en cuenta la decisión de todos los árboles, llegamos a una decisión final conjunta. **Bagging** consiste en obtener varios subconjuntos de datos a partir de un único conjunto mediante remuestreo con reemplazamiento. Esto se hace escogiendo N elementos del conjunto inicial de manera aleatoria pudiendo coger el mismo ejemplo varias veces. Esto se acopla a la construcción de cada árbol de la siguiente manera:

1. A la hora escoger el atributo discriminante de un nodo, el mejor lo elegiremos de entre un subconjunto de todos los atributos. El mejor tamaño de este subconjunto es de tamaño $\sqrt{M}$ o $\log M$, siendo M el número de atributos.

2. Se realiza ese proceso para cada nodo del árbol.

A la hora de realizar la predicción del error se sigue el método **out of bag**. Consiste en recorrer las instancias de entrenamiento y predecir la clase de cada una únicamente con el conjunto de árboles que no han tenido esa instancia en su conjunto de entrenamiento. Se hace ese proceso para todas las instancias que tenemos y predecimos el error.

Algoritmos de selección de atributos

Cuando nosotros tenemos un conjunto de datos, a veces es necesario saber qué atributos de los que tenemos con los más importantes. Puede ocurrir que haya atributos irrelevantes, lo que puede producir sobreajuste. También puede haber atributos redundantes, lo que para ciertos algoritmos de aprendizaje es nocivo. O, simplemente, que tenemos una dimensión de

datos de entrada muy grande, es decir, muchos atributos para cada instancia que, juntado al hecho de tener pocas instancias, puede ser un problema. Estos algoritmos seleccionan atributos en función de diferentes análisis: analizando cada atributo individual, utilizando un modelo de aprendizaje o evaluando grupos de atributos. A continuación, explicaremos los dos algoritmos de selección de atributos más usados en los experimentos del proyecto.

Variance Treshold

Este algoritmo selecciona atributos analizándolos de manera individual. Este algoritmo de selección de atributos se basa en eliminar atributos de baja varianza. Es decir, los atributos con poca varianza, aquellos que son casi idénticos en todos los ejemplos, son eliminados. Esto se realiza de esta manera, debido a que, los atributos con poca varianza, no aportan mucha información al ser muy parecidos en todos los ejemplos, lo que se traduce en que no nos aporta información para discernir si es de una clase u otra.

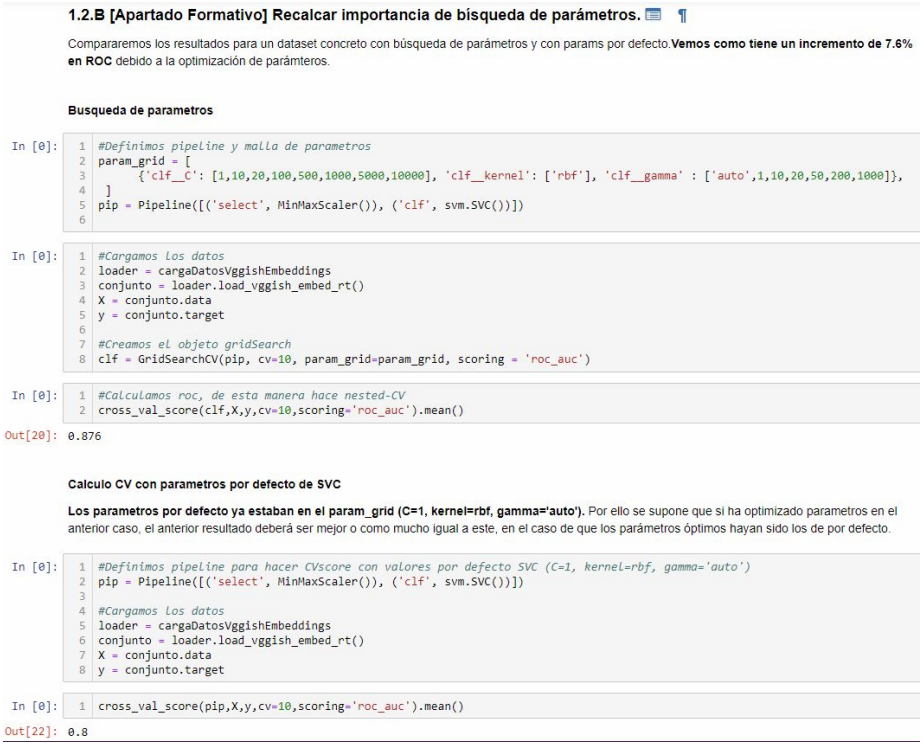
Seleck K best

Este algoritmo también selecciona atributos analizándolos de manera individual. Se basa en que a veces existen atributos poco discriminativos con la clase (el valor del atributo varía sin ninguna relación con la clase). En este método de selección, se aplica un test estadístico entre cada uno de los atributos y la clase, quedándonos con los k mejores, siendo k un parámetro ajustable.

Optimización de parámetros

Una de las tareas más complejas en la minería de datos, es acertar con la parametrización de los modelos. Para un mismo tipo de modelo, puede variar mucho el rendimiento dependiendo el valor de sus parámetros. Pero la tarea de saber que parámetros son los mejores no es una tarea trivial. Requiere de mucho conocimiento del propio modelo, circunstancia que la mayoría de las veces no ocurre.

Para ello, una de las soluciones que podemos tener es probar muchos tipos de parámetros para elegir el mejor. En *Python* esto se llama *Grid Search*. Básicamente, se trata de definir los valores diferentes que queremos probar de cada parámetro o parámetros y probar todas las posibles combinaciones en el modelo. En la Figura 3.4, vemos la importancia de la optimización de parámetros, donde se obtiene un 0.076 AUC más utilizando *Grid Search*.



1.2.B [Apartado Formativo] Recalcar importancia de bisqueda de parámetros.

Compararemos los resultados para un dataset concreto con búsqueda de parámetros y con params por defecto.**Vemos como tiene un incremento de 7.6% en ROC** debido a la optimización de parámteros.

Busqueda de parametros

In [0]:
```
#Definimos pipeline y malla de parametros
param_grid = [
     {'clf__C': [1,10,20,100,500,1000,5000,10000], 'clf__kernel': ['rbf'], 'clf__gamma' : ['auto',1,10,20,50,200,1000]},
 ]
pip = Pipeline([('select', MinMaxScaler()), ('clf', svm.SVC())])

```

In [0]:
```
#Cargamos los datos
loader = cargaDatosVggishEmbeddings
conjunto = loader.load_vggish_embed_rt()
X = conjunto.data
y = conjunto.target

#Creamos el objeto gridSearch
clf = GridSearchCV(pip, cv=10, param_grid=param_grid, scoring = 'roc_auc')
```

In [0]:
```
#Calculamos roc, de esta manera hace nested-CV
cross_val_score(clf,X,y,cv=10,scoring='roc_auc').mean()
```

Out[20]: 0.876

Calculo CV con parametros por defecto de SVC

Los parametros por defecto ya estaban en el param_grid (C=1, kernel=rbf, gamma='auto'). Por ello se supone que si ha optimizado parametros en el anterior caso, el anterior resultado deberá ser mejor o como mucho igual a este, en el caso de que los parámetros óptimos hayan sido los de por defecto.

In [0]:
```
#Definimos pipeline para hacer CVscore con valores por defecto SVC (C=1, kernel=rbf, gamma='auto')
pip = Pipeline([('select', MinMaxScaler()), ('clf', svm.SVC())])

#Cargamos los datos
loader = cargaDatosVggishEmbeddings
conjunto = loader.load_vggish_embed_rt()
X = conjunto.data
y = conjunto.target
```

In [0]:
```
cross_val_score(pip,X,y,cv=10,scoring='roc_auc').mean()
```

Out[22]: 0.8

Figura 3.4: Importancia optimización de parámetros

3.2. Deep Learning

El *Deep Learning* o aprendizaje profundo, consiste en llevar a cabo el proceso de aprendizaje automático mediante redes neuronales. Para ello, se modelan arquitecturas complejas que consiguen transformaciones no lineales múltiples, a través de datos expresados de forma matricial. Dependiendo del número de capas, número de neuronas, funciones de activación y demás elementos de la red neuronal, se obtienen unas arquitecturas u otras. Cabe destacar que estos modelos funcionan como una caja negra y son poco interpretables.

3.3. Conceptos estadísticos

Se describirán a continuación una serie de conceptos matemático-estadísticos utilizados tanto en la evaluación de clasificadores, como en la obtención de medidas de los audios.

ROC

Un gráfico de características operativas del receptor (ROC) [9], es una técnica para visualizar, organizar y seleccionar clasificadores según su rendimiento. Los gráficos ROC se han utilizado durante mucho tiempo en la teoría de detección de señales para representar la relación entre las tasas de aciertos y las falsas alarmas de los clasificadores. El análisis de ROC se ha extendido para su uso en la visualización y análisis del comportamiento de los sistemas de diagnóstico, como es el de nuestro caso. El gráfico ROC traza una curva de probabilidad. Visto de otra manera, puede representar el ratio de verdaderos positivos frente a la razón o ratio de falsos positivos. Es decir, hasta qué punto un clasificador es capaz de clasificar los casos positivos correctamente.

La medida de rendimiento de clasificadores que se saca de este gráfico es la llamada *Area Under Curve* (AUC). En líneas generales, esta medida indica **cómo de bien el modelo es capaz de distinguir entre clases**. Un modelo que es capaz de separar perfectamente todas las clases tendrá un AUC cercano a 1, mientras que si es incapaz de separarlas, tendrá un AUC de 0,5.

Medidas de distribución

Son diferentes medidas extraídas de cada medida física de los audios. Por ejemplo, de la amplitud de onda de un audio se sacan sus 4 funcionales correspondientes con sus momentos de distribución (media, desviación típica, coeficiente de asimetría, curtosis). Se obtienen debido a que caracterizan una muestra de tal manera que si dos distribuciones tienen los momentos iguales son iguales.

- **Media**: Es el resultado de la suma de todos los valores dividida entre el número de ellos. Se corresponde con el momento de la distribución de orden 1 respecto a la origen. m.
- **Desviación**: Mide la dispersión que tienen unos datos respecto a la media. Cuanto mayor sea este valor significará que los datos se encuentran en un rango amplio respecto a la media, mientras que si su valor en bajo significará que los valores se agrupan en un rango cercano a la media. Se corresponde con el momento de la distribución de orden 2 respecto a la media. d.
- **Oblicuidad o coeficiente de asimetría**: Mide la mayor o menor simetría de la distribución. Contra mayor sea el coeficiente de oblicuidad,

mayor será simetría de los datos respecto a la media. Se corresponde con el momento de la distribución de orden 3 respecto a la media. sk.

- **Curtosis**: Mide la mayor o menor concentración de datos alrededor de la media. A un mayor valor de este coeficiente, se entiende que los valores están más agrupados en torno a la media y en valores alejados de ella, dejando los tramos intermedios con menor frecuencia. Se corresponde con el momento de la distribución de orden 4 respecto a la media. k.

Momento de la distribución de orden r respecto a la media:

$$m_r = \sum_{i=1}^{n} (x_i - \overline{x})^r P(n_i) \tag{3.1}$$

3.4. UPDRS

Unified Parkinson's disease rating scale [8], **UPDRS** es la escala universal con la que se mide el grado de severidad del Parkinson. Fue creada para dar un estándar a la evaluación de áreas específicas de la discapacidad. Esta escala evalúa 6 partes fundamentales: 1-Comportamiento, 2-Actividades de la vida diaria, **3-Examen motor** [32]; 4-Complicaciones de la terapia, 5-Escala de Hoehn & Yahr (Severidad) [15] y 6-Escala de Schwab y England (vida cotidiana).

UPDRS-III

Debido al objetivo de nuestro estudio, nos centraremos en la parte 3: examen motor [32]. Utiliza una escala de 0-4, donde 0 es la ausencia de discapacidad motora y 4 discapacidad motora severa. Esta parte trata diversos temas todos relacionados con la capacidad motora corporal. Mide entre otras cosas elementos como el habla, la expresión facial, el temblor o la agilidad. Todo esto afecta a la voz en cuanto se debe realizar un control correcto de los músculos glotales, labios, faringe y más para conseguir una pronunciación satisfactoria. La discapacidad motora produce a la hora de hablar volúmenes bajos, discurso monótono, articulación imprecisa [25]. El englobe de estas capacidades se denomina disartria hipocinética [14].

Hoehn & Yahr

La escala de Hoehn & Yahr [15] evalúa la enfermedad del Parkinson en una escala del 1 al 5. La enfermedad es más severa en función del aumento de la escala. El grado mínimo 1 se corresponde con que la enfermedad es exclusivamente unilateral, aumentando hasta el grado 5 en el que el paciente está en silla de ruedas o en cama, si no tiene ayuda.

3.5. Análisis del discurso

El análisis del discurso se realiza desde 3 prismas diferentes, que son la fonación, la articulación y la prosodia [30]. En nuestro proyecto, fijaremos estos grupos de características a extraer de cada tipo de audios.

Análisis de la fonación

La fonación aborda la vibración de las cuerdas vocales a la hora producir un sonido [30]. Desde el punto de vista clínico, este análisis está relacionado con la curvatura y el cierre incompleto de las cuerdas vocales a la hora de emitir un sonido [28]. Las medidas más típicas para el análisis de fonación son Jitter, Shimmer, APQ y PPQ [26]. Explicadas en la sección 3.6.

Análisis de la articulación

La articulación comprende la modificación de la posición, el estrés y la forma de los órganos y tejidos involucrados en la producción del habla [30]. Esto se manifiesta en déficits como, por ejemplo, una reducción de la amplitud y la velocidad de los movimientos articulatorios de los labios, la mandíbula y la lengua [31]. La capacidad de articulación es evaluada a través de la energía que se libera en las transiciones entre segmentos *voiced* $\rightarrow$ *unvoiced* (transición entre segmentos con voz y sin voz) y viceversa con medidas como MFCC o BBE, explicadas en 3.6. Se basa en que los pacientes de PD tienen dificultad para comenzar y para detener la vibración de las cuerdas vocales [26].

Análisis de la prosodia

La prosodia aborda temas como la variación del volumen, el tono y la sincronización para hablar de manera natural [30]. Se manifiesta con monotonía en el volumen y en el tono, cambios de rapidez en el habla y dificultades a la hora de expresar emociones a través del discurso [22]. Las

medidas más típicas para el análisis de prosodia son las relacionadas con la frecuencia fundamental, el contorno de la energía y la duración. Explicadas en la sección 3.6.

3.6. Características Físicas de la voz

Desde el punto de vista ingenieril y relacionado con el aprendizaje automático, todos los análisis anteriores se deben expresar de una manera numérica para poder crear vectores de características de un audio. Por ello, se extraen diferentes características numéricas de los audios que intentan expresar los análisis de fonación, articulación y prosodia de la manera más óptima posible. Otra dificultad importante desde el punto de vista ingenieril es la continuidad de los audios. Los audios cambian de manera continua en el tiempo, lo que complica el proceso de la extracción de características. Por ello, el audio se divide pequeños segmentos de tiempo en los que se calculará las características de manera estática para ese tramo. Las medidas físicas extraídas de los archivos de voz son de gran variedad y por ello en este apartado explicamos las más comunes y más usadas en los diferentes estudios del análisis del discurso.

Jitter

El concepto **Jitter** es usado para evaluar la variabilidad temporal durante el envío de señales digitales [36]. Concretamente, en el análisis del discurso, esta medida significa la variación temporal de la frecuencia fundamental del discurso [26]. N es el número de fragmentos, M_f la frecuencia máxima y $F_0(k)$ la amplitud de ese frame en concreto:

$$Jitter(\,\%) = \frac{100}{NM_f}\sum_{k=1}^{N}|F_0(k) - M_f| \tag{3.2}$$

Shimmer

El concepto **Shimmer** es usado, en el análisis del discurso, para medir la variación temporal de la amplitud del discurso [26]. N es el número de *frames*, M_a la amplitud máxima y $A(k)$ la amplitud de ese *frame* en concreto:

$$Shimmer(\,\%) = \frac{100}{NM_a}\sum_{k=1}^{N}|A(k) - M_a| \tag{3.3}$$

APQ y PPQ

APQ mide la variabilidad a largo plazo de la amplitud de la voz. Para calcularla, se suaviza mediante una media móvil de tamaño 11 y se calcula como la diferencia media absoluta entre la amplitud de un *frame* y las amplitudes promediadas sobre sus vecinos, dividida por la amplitud media.

La **PPQ** mide la variabilidad a largo plazo de la frecuencia fundamental y para calcularla se suaviza mediante una media móvil de tamaño 5. Se calcula como la diferencia promedio absoluta entre la frecuencia de cada cuadro y el promedio de sus vecinos, dividida por la frecuencia media. Ambas medidas se calculan de igual manera, APQ relativa a la amplitud y PPQ relativa a la frecuencia [26].

Mel Frequency Cepstral Coefficients - MFCC

Los **Coeficientes Cepstrales en las Frecuencias de Mel** son 12 coeficientes para la representación del habla basados en la percepción auditiva humana. Con ellos podemos obtener la información más relevante de una porción de audio, obviando partes menos importantes como el ruido. El cálculo de estos coeficientes está basado en la transformada de Fourier y la transformada del coseno discreta [38]. Como hemos comentado anteriormente, nos sirve para medir la energía que se libera en las transiciones entre segmentos *voiced* $\rightarrow$ *unvoiced* y viceversa. Tiene un crecimiento logarítmico y se calcula según la siguiente función:

$$m = 1127,01048 \ln \frac{1+f}{700} \tag{3.4}$$

Bark Band Scale

Bark Band Scale es otro método de caracterización de la energía liberada en las transiciones de la voz. Da 24 coeficientes acorde a la escala de Bark [40]. Es una escala psicoacústica en la que se definen límites de frecuencias para definir la escala (i.e. grado 1 [100Hz-200Hz], grado 2 [200Hz-300Hz],..., grado 23 [12000-15500], grado 24 [15500- ∞]) Tiene un crecimiento logarítmico y se calcula según la siguiente función:

$$Bark(f) = 13 \arctan(0{,}00076 f) + 3{,}5 \arctan\left(\frac{f}{7500}\right)^2 \tag{3.5}$$

Medidas relacionadas con la frecuencia fundamental

Su objetivo es caracterizar los patrones de entonación de la voz. Por ello, se calcula el contorno de la frecuencia máxima y se calculan diferentes estadísticos sobre ella, como medias en Hz, desviación en Hz, máximos, etc [26].

Medidas relacionadas con la energía

Al igual que para el anterior caso, pero esta vez con la energía. Se calculan medidas estadísticas sobre el contorno de la energía: media en dB, desviación en dB, máximo o el coeficiente de regresión entre el contorno de energía y una regresión lineal [26].

Técnicas y herramientas

Explicaremos las herramientas utilizadas en nuestro proyecto.

4.1. Python

Se ha utilizado el lenguaje de programación *Python*. Es un lenguaje de programación dinámicamente tipado, que soporta orientación a objetos. Posee una licencia de código abierto y es uno de los lenguajes más usado en el ámbito del *Data Science*, junto con R.

4.2. Jupyter Notebook IDE

IDE de programación de *Python* basado en *iPython* de código abierto. El cuaderno Jupyter es una aplicación web que le permite crear y compartir documentos que contienen código en vivo, ecuaciones, visualizaciones y texto narrativo. Sus usos más frecuente son: limpieza y transformación de datos, simulación numérica, modelado estadístico, visualización de datos, aprendizaje automático, etc.

4.3. Bibliotecas de Python

NumPy

NumPy [23] es una biblioteca de *Python*, especializada en computación de datos. Posee gran potencial para el manejo de datos numéricos y sus operaciones, sobre todo de manera matricial, ya que contiene funciones sofisticadas y de uso simple.

Scikit-Learn

Scikit-Learn [27] es una biblioteca con funciones de aprendizaje automático. Contiene paquetes y funciones útiles para la creación de modelos. Incluye desde los modelos más simples, hasta todas las diferentes funciones para seleccionar atributos, optimizar parámetros, realizar validaciones cruzadas, mostrar resultados, etc.

Scipy

SciPy [19] es una biblioteca que contiene herramientas y algoritmos matemáticos. Su base es el objeto multidimensional de *NumPy*.

Pandas

Pandas [21] es una biblioteca, escrita como extension de *NumPy*, para manipulación y análisis de datos. Es utilizada para mostrar todos los datos y características extraídas.

Matplotlib

Matplolib [16] es una biblioteca que se usa para la generación y muestra de diferentes gráficos. Procesa los datos de *Python* y genera diferentes salidas. Su funcionamiento es parecido a *Matlab*.

Sounddevice

Este módulo de *Python* proporciona enlaces para la biblioteca *PortAudio* y algunas funciones convenientes para reproducir y grabar arreglos *NumPy* que contienen señales de audio. Tiene licencia libre MIT.

Pydub

Se utiliza para manipular el audio de una manera simple y con una interfaz de alto nivel. En su documentación se detalla:“ *Pydub lets you do stuff to audio in a way that isn't stupid* ".

Os

La biblioteca *os* proporciona una forma sencilla de utilizar la funcionalidad del sistema operativo. Proporciona una interfaz para utilizar los comandos del sistema operativo, independientemente de cual sea este.

Disvoice

La biblioteca Disvoice[3] [26] es un conjunto de *scripts* de Python para la extracción de medidas del habla. Disvoice calcula medidas de articulación, de la fonación y de prosodia a partir de vocales sostenidas y expresiones verbales continuas, con el objetivo de evaluar las capacidades de comunicación de los pacientes con diferentes trastornos de la voz o trastornos neurodegenerativos como la enfermedad de Parkinson. Ha sido desarrolada por Juan Camilo Vásquez-Correa, el cual es co-autor de varios artículos con Juan Rafael Orozco-Arroyave como [26], y tiene licencia de software MIT. Cabe destacar que se han intercambiado dos correos electrónicos con JC Vásquez-Correa, en los que se nos explica tanto la utilización de los *scripts*, como la salida detallada de cada uno de ellos.

Contiene 3 *scripts* principales para la extracción de características de audios (fonación, articulación y prosodia), extrayendo por ejemplo de un audio hasta 488 medidas relacionadas con la articulación. Tiene una gran parte de su contenido dedicado a la visualización de los audios mediante diferentes métodos, característica que no es usada en nuestro proyecto.

Esta biblioteca ha sido usada, como puede entenderse, para la etapa de extracción de características de los audios. Está escrita para ser usada como *scripts* de Python y utiliza internamente varias bibliotecas como *scipy*, *numpy*, *scikitlearn*, *pysptk*, *sounddevice*, *os* y programas como *Praat* [2]. A la hora de ser utilizada por nosotros ha tenido que ser configurada para su correcto funcionamiento en nuestro entorno, por lo que el código de la herramienta exacto que utilizamos está alojado en Disovice[4] de mi repositorio. Los detalles de la configuración y los cambios realizados se dan en el manual del programador.

Keras

Keras [37] es una biblioteca de Redes Neuronales de Código Abierto escrita en *Python*. Es capaz de ejecutarse sobre *TensorFlow*. Está especialmente diseñada para posibilitar la experimentación en más o menos poco tiempo con redes de Aprendizaje Profundo. Sus fuertes se centran en ser amigable para el usuario, modular y extensible.

[3] https://github.com/jcvasquezc/DisVoice
[4] https://github.com/AdrianArnaiz/DisVoice

VGGish

VGGish[5] [13] es una red neuronal convolucional, la cual está preentrenada con Audioset[6] [11], un conjunto de datos de más de 2 millones de pistas de sonido de vídeo de 10 segundos etiquetadas por humanos, con etiquetas tomadas de más de 600 clase. Está codificada usando *TensorFlow*. *AudioSet* fue lanzado en marzo de 2017 por el equipo de *Sound Understanding* de Google para proporcionar una tarea de evaluación común a gran escala para la detección de eventos de audio. Esta red VGGish ha sido utilizada para extraer características de audios, es decir, de la red hemos utilizado únicamente la etapa de extracción.

VGGish2Keras

VGGish2Keras[7]. Es una biblioteca alojada en un repositorio público de *Github*, la cual convierte el modelo *VGGish* a un modelo tipo *Keras*. Para su uso, primero deberemos descargar *VGGish* y sus dependencias y, posteriormente, descargar los archivos del repositorio *VGGish2Keras* en el mismo directorio y ejecutarlos.

4.4. Praat

Praat[8] [2] es un programa el cual nos permite realizar análisis fonéticos de audios vocales. Esta herramienta está enfocada a la investigación del habla. Permite hacer una multitud de análisis diferentes entre los que se encuentran análisis del discurso (análisis espectrales, análisis de intensidad, de formantes...) o análisis estadístico. Una característica importante para nosotros es que permite ser ejecutado mediante línea de comandos con diferentes parámetros. Otro aspecto interesante es que Praat tiene un *wrapper* para *Python* llamado *Parselmouth* [18], aunque nosotros no lo utilizamos en el proyecto. Ha sido desarrolada por Paul Boersma y David Weenink de la Universidad de Ámsterdam.

Será utilizada internamente por la biblioteca Disvoice para analizar una serie de características de los audios, que posteriormente volverá a procesar Disvoice con diferentes métodos de *Python* para devolvernos a nosotros las características finales deseadas.

[5] https://github.com/tensorflow/models/tree/master/research/audioset
[6] https://research.google.com/audioset/
[7] https://github.com/antoinemrcr/vggish2Keras
[8] http://www.fon.hum.uva.nl/praat/

4.5. Git

Sistema de control de versiones distribuido de código abierto. Estos sistemas son útiles ya que nos permiten funciones como volver a un punto anterior del proyecto, a parte de tener información detallada de los cambios.

4.6. Github

Github es un servicio *online* de código abierto que nos permite alojar nuestro repositorio del proyecto usando el control de versiones Git. Permite la integración de varias herramientas de ayuda al desarrollo, como puede ser *ZenHub*.

4.7. ZenHub

Zenhub es una herramienta, que se integra sobre *Github*, y que nos permite llevar un mejor control del proyecto, añadiendo elementos *Scrum* a nuestro repositorio *Github*.

4.8. TortoiseGit

TortoiseGit es una cliente de control de versiones de *Git*, que nos proporciona una herramienta de escritorio para manejar nuestro repositorio en el escritorio. Nos permite utilizar todas las herramientas de *Git* en una interfaz gráfica de manera sencilla.

4.9. LaTeX

LaTeXes un sistema de composición de textos, orientado a la creación de documentos escritos que presenten una alta calidad tipográfica. [35]

TexMaker

TexMaker es un editor de LaTeXde código abierto. Integra variedad de herramientas para desarrollar documentos con LaTeX. Para que previamente funcione es necesario haber instalado MiKTeK.

MiKTeK

MiKTeK es una distribución de LaTeXpara Windows. Es libre, incluye muchas tipografías, contiene compiladores para generar archivos de diferentes tipos (por ejemplo pdf) y herramientas para generar bibliografías e índices.

Aspectos relevantes del desarrollo del proyecto

5.1. Inicio del proyecto

Este proyecto se presentó como un proyecto de investigación sobre la extracción de biomarcadores de la voz para la detección de enfermedades neurodegenerativas o depresión. Al principio, se carecía de un objetivo concreto, debido a la incertidumbre inicial de qué camino se debía seguir y cómo iba a estar de avanzado este área de investigación.

Tras recopilar información, tanto de artículos científicos, como de otras fuentes, se llegó a la conclusión de que el objetivo del proyecto era mejor que estuviera relacionado con la enfermedad del Parkinson. Valoramos diferentes enfermedades como alzheimer, depresión y esclerosis lateral amiotrófica (ELA). Elegimos la enfermedad del Parkinson debido a que la investigación de la detección de esta enfermedad a través de la voz estaba más avanzada y hay muchos artículos recientes y noticias de la realización actual de proyectos en este campo.

5.2. Investigación del proceso a seguir

Una vez establecida la enfermedad decidimos el proceso a seguir para conseguir un modelo correcto de clasificación de la enfermedad. El proceso a grandes rasgos estaba claro: extraer características de audios y utilizarlas para crear un clasificador. Pero antes, había que decidir varios puntos importantes en el proceso a seguir: ¿De qué audios se deben extraer las características? ¿De dónde íbamos a obtener esos audios? ¿Qué tipo de

pre-procesamiento requieren los audios? ¿Qué características se sacan de cada uno de ellos?...

Antes de todo, cabe destacar que como uno de los objetivos era hacer un estudio de investigación sobre diferentes algoritmos para la clasificación de los audios, necesitamos obtener un conjunto de audios de un proyecto concreto para poder comparar nuestros resultados de manera objetiva con ese proyecto concreto. Por ello, el proceso que íbamos a seguir podía estar influenciado de manera directa por el conjunto de datos que se nos prestara.

A la hora de intentar responder a las anteriores preguntas, nos documentamos a través de los artículos más importantes en este campo, con el objetivo de obtener las ideas más relevantes de cada uno de ellos, para decidir el enfoque de nuestro proceso. Los grupos de investigación más importantes se correspondían con dos grupos diferentes de investigadores, los cuales tienen artículos relevantes sobre este tema. La explicación en detalle se dará en el apartado *Trabajos Relacionados* 5.9, sin embargo aquí haremos eco de las ideas más importantes. Un grupo es el compuesto por Max A. Little y Tsanas Thanasis con artículos como [20]. Se puede obtener una serie de ideas principales de este grupo:

- Utilizan únicamente audios de **vocales sostenidas** para la obtención de características.
- Sostienen que un conjunto pequeño de características (<20) es suficiente para una correcta clasificación de los audios. Incluso [34] está relacionado íntimamente con esta idea, ya que a partir de un conjunto grande de características utiliza diferentes técnicas de selección de características y demuestra que con un conjunto menor que 20 se obtienen buenos resultados.

Otro grupo, también destacado, es el de J. R. Orozco-Arroyave J. D. Arias-Londoño y J. F. Vargas-Bonilla, con artículos como [25]. La idea más importante que podemos obtener es la siguiente:

- La pronunciación de **consonantes en discurso corrido** (frases, textos, palabras...) **aporta mucha información** de la pronunciación debido a la intervención de diferentes músculos necesarios para ésta. Por ello, se deben analizar otros tipos de audios para obtener más información que si analizamos únicamente pronunciación de vocales sostenidas.

- Cada tipo de audio debe ser utilizado para hacer un clasificador diferente. No se pueden utilizar diferentes tipos de audio dentro del mismo clasificador, ya que obtenemos resultados confusos, derivados de las diferentes pronunciaciones de diferentes palabras, frases, etc.

Condensando ambas, llegamos a la conclusión de que debíamos analizar diversa variedad de audios (ya que aportan más información) sin obsesionarnos por obtener un número inmenso de características de cada uno. Por ello obtendremos variedad de clasificadores que se corresponderán con la variedad de tipos de audio que tengamos y haremos un estudio sobre ellos. Los temas de qué audios utilizar, como pre-procesarles o que características obtener de cada uno se abordará en los siguientes apartados.

También se realizó una investigación de todas las posibles características a extraer de los audios y todas las posibles herramientas con las que extraer esas características. Como resultado de esta investigación se crearon una serie de taxonomías, incluidas en el **anexo de investigación**, en donde se resumen:

- Artículos relacionados con el tema con información del mismo.
- Bases de datos encontradas en el estado del arte con información de las mismas (privacidad, características...).
- Características que se sacan de cada tipo de audio, con información del artículo donde están descritas.
- Características que se sacan en cada artículo con información sobre ellas.
- Relación Bibliotecas-Características y viceversa.

5.3. Conjunto de audios

El conjunto de datos de audios usado es el descrito en [24]. Como se explica en el artículo, es un conjunto de audios en castellano de 100 personas: 50 de ellas pacientes con Parkinson (PD [9]) y 50 de ellas personas sanas,

[9]Parkinson Disease

pacientes de control (HC [10]). Es un conjunto de datos realizado de la manera más balanceada posible, a parte de contener 50PD-50HC, también está balanceado en cuanto a sexo y edad tanto dentro de los 50 PD como dentro de los 50 HC. Todos los pacientes han sido diagnosticados por expertos. Se recoge en un arcivo excel las características de los audios: edad, sexo, tiempo desde la diagnosis y 3 diferentes medidas: PD/HC (sano o Parkinson), UPDRS-III [8] y Hoehn & Yahr scale [15]. Como se ha visto en 3.4 , tanto UPDRS como Hoehn & Yahr son dos escalas utilizadas para medir la severidad del Parkinson.

El conjunto de audios contiene un total de 4200 audios tipo wav, divididos en los siguientes tipos:

Monólogos 50 monólogos de PD y 50 de HC. El contenido es discurso libre sobre la respuesta a la pregunta *¿Qué haces cuando te levantas por la mañana?*. La duración de este tipo de audios comprende desde 00:30 a 02:30.

Texto leído 50 audios de PD y 50 de HC. El contenido es el siguiente texto balanceado: *Ayer fui al médico. ¿Qué le pasa? Me preguntó. Yo le dije: Ay doctor! Donde pongo el dedo me duele. ¿Tiene la u~ña rota? Sí. Pues ya sabemos qué es. Deje su cheque a la salida.*

Vocales 750 audios de PD y 750 de HC. Pronunciación sostenida de cada una de las 5 vocales 3 veces por persona. 150 audios de HC y 150 de PD por cada vocal.

Palabras 1250 audios de PD y 1250 de HC. Pronunciación de palabras para el análisis silábico. Cada persona tanto PD como HC pronunciará 25 palabras diferentes: *brasa, coco, petaca, etc.*

5.4. Metodología del estudio - Resumen general

En esta sección se hará una mera introducción inicial al estudio realizado, explicado más adelante en las siguientes secciones y mucho más detallado en los *notebooks* del proyecto. En estos notebooks se detallan tanto detalles de implementación, como detalles de uso, como detalles de análisis de resultados y del proceso de la investigación. Los notebooks se encuentran tanto en

[10]Healthy Control

el directorio `/src` como en el directorio `/src/vggish`. A lo largo de esta sección, se irán detallando en qué notebooks aparece cada experimento o cada extracción de características.

Un aspecto importante del proyecto es el siguiente. En un primer momento se planeó la extracción de características, creación de los clasificadores y finalmente utilización de ellos. Para ello comenzamos, en la **primera fase** de experimentos, extrayendo las características (que comentaremos posteriormente) con la biblioteca **Disvoice**. Ésta es una herramienta pública desarrollada por el mismo grupo de investigación del artículo [25]. Creamos los diferentes clasificadores para esas características y obtuvimos resultados inferiores a los recogidos en el estado del arte, i.e. [25]. Obtuvimos resultados inferiores a pesar de usar los mismos clasificadores con los mismos parámetros y de mantener conversaciones con los autores para resolver posibles ambigüedades sobre las herramientas o el proceso utilizado. Destacamos que en las diferentes fases del estudio, lo que hemos ido variando han sido los conjuntos de datos extraídos de cada audio, realizando los mismos experimentos con clasificadores (o modificando los experimentos muy poco).

Como no obtuvimos los resultados esperados, decidimos hacer una mejora a los experimentos, una **segunda fase** del estudio. Esta mejora consistía en dos partes. La primera de ellas fue añadir para cada instancia los atributos **edad y sexo** del paciente a las características extraídas por Disvoice. Estos datos les tenemos disponibles en el archivo excel descrito en [24], que se nos envió junto con los audios cortesía de J.R Orozco-Arroyave de la Universidad de Antioquía, Colombia. La segunda mejora fue separar las instancias del conjunto de datos entre hombres y mujeres. Se planteó de esta manera, debido a que [25] hace validación cruzada estratificada según dos elementos: edad y clase (PD: *Parkinson Disease* o HC: *Healthy control*). Con la biblioteca utilizada para los experimentos, *scikitlearn*, solamente se puede estratificar según 1 atributo. Separando los conjuntos de datos por sexos y estratificando cada sexo por clase, estamos simulando esa validación cruzada estratificada por 2 atributos que utiliza ese artículo. Se mejoraron los resultados obtenidos con los conjuntos de datos de la primera fase, pero aun así estaban lejos de los resultados en el estado del arte.

Con intención de mejorar los resultados y dar una perspectiva diferente a los experimentos, realizamos una **tercera fase** de los mismos. En esta tercera fase utilizamos la biblioteca de *Deep Learning* llamada **VGGish** [13] (ver sección). Consiste en la extracción de características mediante la una red neuronal pre-entrenada con audios de *Youtube*, que utiliza capas convolucionales. Al estar pre-entrenada, ya tenemos los pesos para la extrac-

ción de características y, por tanto, lo único que debemos realizar es utilizar las funciones de extracción de esa biblioteca. En esta fase, sacamos para cada audio otros dos conjuntos de características, embeddings de VGGish (media y desviación de embeddings, se explicará en 5.7) y espectros de frecuencia, con los cuales realizamos los experimentos con los clasificadores. Los resultados obtenidos seguían estando en la magnitud de los obtenidos por nosotros, pero sin llegar a los mejores del estado del arte.

Cabe destacar que en todos ellos se ha realizado una validación cruzada con 10 folds, ya que es la que se utiliza en [25] o [33]. También destacamos que la medida que hemos utilizado para los clasificadores es el área bajo la curva ROC, `roc_auc_score` en *Scikit-Learn*. Esto es debido al siguiente motivo. Lo más óptimo hubiera sido extraer tanto el área bajo la curva, como el *accuracy*, como otras medidas. Sin embargo, en las clases creadas por nosotros (experimenters y funciones de realización de experimentos), no se pueden extraer multimétricas de los experimentos, si queremos extraer dos métricas diferentes, se debe realizar otra vez el experimento indicando que se devuelve la otra métrica. Debido a la cantidad de experimentos realizada, hemos decidido que la mejor medida era el área bajo la curva ROC. Es debido a que si obtenemos altas magnitudes de esta medida, aunque después medidas como la *accuracy* no sean tan altas, lo único que debemos hacer el elegir correctamente cual es el umbral de decisión del clasificador. Es decir, elegir a partir de qué probabilidad de clase predicha por el clasificador, nuestra instancia es de una clase o de la otra. El último aspecto a destacar es que todos los experimentos han sido realizados en *notebooks* de Jupyter. Se ha considerado que es el entorno interactivo más idóneo para la realización y presentación de experimentos. En todos estos *notebooks* se realiza una explicación detallada de los experimentos y, también, una visualización de resultados con multitud de gráficas y tablas para su mejor análisis y comprensión. Algunos de los resultados más importantes se encuentran en la sección 5.8.

Tras finalizar todos estos experimentos con todos los diferentes conjuntos de datos extraídos de los audios, realizamos una aplicación de escritorio con tkinter, en la que se muestre cómo podría aplicarse el resultado de investigación a un producto final. En ella se podrá analizar un audio y, tras procesarlo, que nos diga la probabilidad de que la persona de ese audio tenga parkinson o no, y se muestren las gráficas de amplitud de onda y de espectrograma de frecuencias.

5.5. Primera Fase: atributos Disvoice

Nuestro estudio comprenderá la comparación de diferentes clasificadores construidos cada uno con diferentes conjuntos características para cada tipo de audio. Utilizaremos 3 diferentes tipos de audio: **vocales sostenidas, 5 palabras diferentes y texto leído**. Qué características sacamos para cada tipo de audio se presentará en la subsección *Modelado del discurso* 5.5. El proceso para la realización de los clasificadores es el siguiente (ver Figura 5.5):

1. Pre-procesamiento de los audios: preparación de audios para la extracción de diferentes medidas.
2. Modelado del discurso: extracción de diferentes tipos de características para cada tipo de audios.
3. Clasificación: construir diferentes clasificadores para cada conjunto de características para hacer un estudio comparativo.

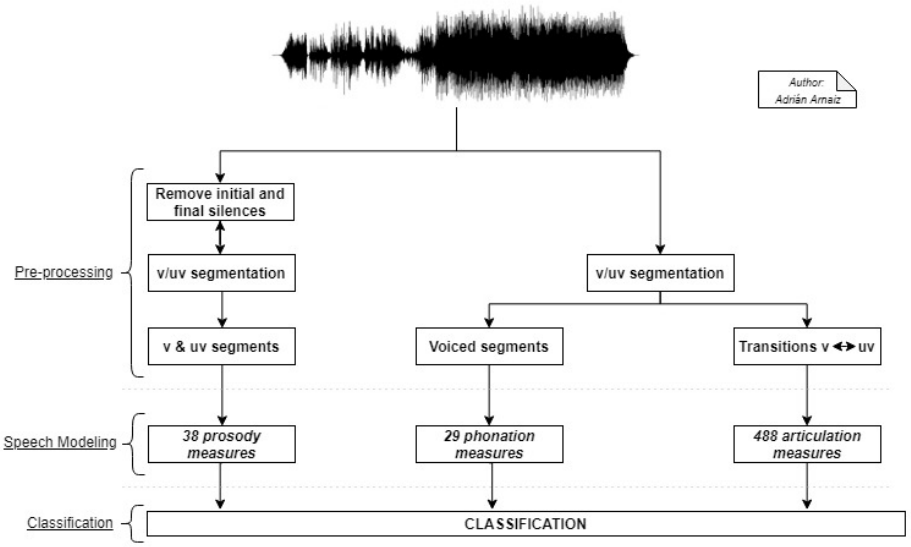


Figura 5.5: Esquema del proceso para abordar los experimentos, basada en [25]

Pre-procesado de audios

En esta etapa se realizan 2 tareas principales: la eliminación de silencios iniciales y finales de los audios y la segmentación en fragmentos con voz

y sin voz de los mismos (los llamaremos segmentos *voiced* y *unvoiced*). La eliminación de sonidos inicial y final de los audios se realiza ya que a la hora de extraer las medidas de prosodia se puede tener algunos inconvenientes si los silencios iniciales son muy largos. Si fueran excesivamente largos tendríamos resultados erróneos en las características, como, por ejemplo, las relacionadas con la duración promedio de silencios, la variabilidad de la duración de los silencios y otras medidas que se calculan sobre las pausas. Para la extracción de medidas de fonación y articulación este proceso lo realizan internamente los *scripts* de la biblioteca Disvoice [26] utilizando Praat. Sin embargo, a la hora de obtener las medidas prosódicas es necesario realizar la eliminación de manera previa.

La **segmentación en fragmentos *voiced* y *unvoiced*** se realiza para analizar el discurso, es decir, se sacarán medidas que necesitan de esta fragmentación (i.e. *Jitter* de los fragmentos *voiced* o MFCC de las transiciones entre *voiced* y *unvoiced*). Esta tarea la hacen internamente los *scripts* de la biblioteca Disvoice utilizando Praat.

Modelado del discurso

Se extraerán, con la herramienta Disvoice, 3 diferentes conjuntos de características para cada tipo de audio: de fonación, de articulación y prosódicas. Como hemos visto en nuestro conjunto de datos, tenemos texto leído, palabras, vocales y monólogos. En este proyecto utilizaremos el texto leído, las 5 palabras con mejor resultado en [25] (*atleta, campana, braso, gato, petaca*) y las 5 vocales. El monólogo no será utilizado debido a que al ser discurso libre y no predefinido, las características extraídas dependen también de cómo sea el discurso (i.e. las palabras que se digan, las pausas...). También se realizará la limpieza de las características, en nuestro caso, tratar los datos de tipo *NaN* resultantes en la extracción. Como se ve en los *notebooks* de extracción de características y se comentará posteriormente, se decidió por eliminar las instancias que contienen *NaN*, ya que no eran muchas.

La extracción de estas características se encuentra realizada y explicada en el notebook de la ruta `TFG-Neurodegenerative-Disease-Detection/src/Extracción de características.ipynb`.

Medidas de fonación

De las medidas de fonación obtendremos 11 conjuntos diferentes: 1 para el texto leído, 5 para las palabras (1 por palabra elegida) y 5 por vocal (1

Caract.	Número	Breve descripción
1ª derivada F0	1×4=4	1ª deriv. frec fundamental
2ª derivada F0	1×4=4	2ª deriv. frec fundamental
Jitter	1×4=4	Perturbación de F0
Shimmer	1×4=4	Perturbación de Amplitud
APQ	1×4=4	Cociente de perturb. de amplitud
PPQ	1×4=4	Cociente de perturb. de periodo
Energía Log	1×4=4	Explicado en [1]
Grado unvoiced	1	Grado *unvoiced*

Tabla 5.1: Características de fonación. En detalle en [26].

por vocal). En total, de cada audio se sacan un conjunto de **29 medidas** basadas en la perturbación de la fonación. Las medidas de fonación, son extraídas de los segmentos *voiced*, utilizando para ello la biblioteca Disvoice (`phonation.py`). Estas características son descritas en la tabla 5.1.

Obtenemos un vector de 29 características, las descritas en la tabla 5.1: las 7 medidas por sus 4 funcionales (media m, desviación std, curtosis k y oblicuidad sk) + grado de *unvoiced* [a].

[a] El grado de unvoiced es el ratio entre la duración de los segmentos sin voz entre la duración total del audio [26].

Medidas de articulación

De las medidas de articulación obtendremos 6 conjuntos diferentes: 1 para el texto leído y 5 para las palabras (1 por palabra elegida). En total de cada audio se sacan un conjunto de **488 medidas** de articulación. Las medidas de articulación son extraídas de las transiciones entre los segmentos *voiced* y *unvoiced* utilizando para ello la biblioteca Disvoice (`articulación.py`). Estas características son descritas en la tabla 5.2

Obtenemos un vector de 488 características, las descritas en la tabla 5.2: las 122 medidas por sus 4 funcionales(media m, desviación std, curtosis k y oblicuidad sk).

Caract.	Número	Breve descripción
BBE onset	22×4=88	22 coef. BBE de trans. *v* ->*uv*
MFCC onset	12×4=48	12 coef. MFCC de trans. *v* ->*uv*
1ªD MFCC onset	12×4=48	1ª deriv. 12 coef. MFCC de trans. *v* ->*uv*
2ªD MFCC onset	12×4=48	2ª deriv. 12 coef. MFCC de trans. *v* ->*uv*
BBE offset	22×4=88	22 coef. BBE de trans. *uv* ->*v*
MFCC offset	12×4=48	12 coef. MFCC de trans. *uv* ->*v*
1ªD MFCC offset	12×4=48	1ª deriv. coef. 12 MFCC de trans. *uv* ->*v*
2ªD MFCC offset	12×4=48	2ª deriv. coef. 12 MFCC de trans. *uv* ->*v*
1ª formante F0	1×4=4	1ª formante de frecuencia
1ªD 1ª formante F	1×4=4	1ª deriv. 1ª formante de frecuencia
2ªD 1ª formante F	1×4=4	2ª deriv. 1ª formante de frecuencia
2ª formante F	1×4=4	2ª formante de la frecuencia
1ªD 2ª formante F	1×4=4	1ª deriv. 2ª formante de frecuencia
2ªD 2ª formante F	1×4=4	2ª deriv. 2ª formante de frecuencia

Tabla 5.2: Características de articulación. En detalle en [26].

Caract.	Número	Breve descripción
Frec. fundamental	7	relativas a la frec. fundamental
Energía	9	9 medidas relativas a la energía
Ratios *v*-*uv*	22	22 medidas relativas a *v*-*uv*

Tabla 5.3: Características de prosodia. En detalle en [26].

Medidas de prosodia

De las medidas de prosodia obtendremos 1 conjuntos para el texto leído. En total de cada audio se sacan un conjunto de **38 medidas** basadas en la duración, la frecuencia fundamental, la energía y ratios de la composición del audio en lo relativo a segmentos *voiced* y *unvoiced*. Las medidas de prosodia son extraídas del audio completo, tanto segmentos *voiced* como *unvoiced*, utilizando para ello la librería Disvoice (`prosodia.py`). Estas características son descritas en la tabla 5.3.

Obtenemos un vector de 38 características, las descritas en la tabla 5.3. Esta vez sin sacar los funcionales para cada medida.

Conjuntos de datos totales

En total tenemos **18** conjuntos diferentes de características: medidas de fonación articulación y prosodia para la frase, medidas de fonación y articulación para cada una de las palabras y medidas de fonación para las vocales sostenidas. Se construyen clasificadores independientes para cada subconjunto diferentes, como hemos comentado anteriormente, no se pueden mezclar características extraídas de diferentes frases, palabras o vocales, ya que hay características intrínsecas a la pronunciación de esos elementos. Cada conjunto de características se ha extraído en *scripts* en *notebooks* de Python, utilizando métodos de clases que envuelven el funcionamiento de la biblioteca Disvoice. Los conjuntos de datos extraídos de vocales y palabras tienen 100 instancias con N atributos, siendo N el número de atributos descrito en las 3 anteriores subsecciones. Los conjuntos extraídos de las vocales tienen 300 instancias, ya que para cada vocal tenemos 3 audios distintos. Para cada conjunto de características se ha creado una función del mismo estilo que los cargadores de datos de *Scikit-Learn*: funciones dentro de módulos que devuelven objetos de tipo *Bunch*. Además, cada conjunto de datos tiene un nombre del estilo (fon|art|prs)_(rt|w_palabra|v_vocal)_ccas y se guardan en formato numpy en el directorio `src/CaracteristicasExtraidas`.

Explicación de la nomenclatura:

fon Fonación.

art Articulación.

prs Prosodia.

rt Frase, *read-text*.

w_palabra Palabra, i.e. *fon_w_gato_ccas* o *art_w_braso_ccas*.

v_vocal Vocal, i.e. *fon_v_A_ccas* o *art_v_E_ccas*.

Experimentos con clasificadores

Los experimentos de esta fase se encuentran realizados y explicados en el *notebook* alojado en `TFG-Neurodegenerative-Disease-Detection/src/1os Experimentos clasificadores.ipynb`. Para todos los conjuntos de datos se han realizado varios tipos de experimentos, comprendiendo desde los clasificadores estándar de *Scikit-Learn*, hasta métodos más complejos de *bagging* o *boosting* con preprocesado de datos (normalización), selectores de

características (*select k best, variance treshold...*), búsquedas exhaustivas de parámetros...

Los experimentos realizados han sido los siguientes:

- Experimento con clase `experimenter`. Esta es una clase implementada por nosotros, que, explicada a grandes rasgos, dados unos conjuntos de datos y clasificadores pasados como un diccionario, realiza todos los pares de experimentos devolviendo todos los resultados. Los conjuntos de datos serán los 18 descritos anteriormente y los clasificadores serán 8 clasificadores diferentes de *Scikit-Learn* con sus valores por defecto y sin ningún elemento de slección de atributos o búsqueda de parámetros. Obtenemos una tabla de resultados de 144 resultados de los 144 experimentos diferentes. El AUC más alto obtenido es de **0.836**, obtenido con el método AdaBoost para el conjunto de datos de fonación de la frase (*read_text*).

- Experimento llamado *Orozco* ya que es el experimento realizado en [25]. Consiste en utilizar el algoritmo de clasificación SVM con kernel gaussiano, con una búsqueda exhaustiva de los parámetros C y lambda. Por ello hemos implementado un *grid search* con *Scikit-Learn*. El mejor resultado es de **0.744** AUC para las características de fonación de la vocal U. Muy lejos del 0.99 AUC y 0.99 *accuracy* que se obtiene en [25] con el mismo experimento.

- Reducción de dimensionalidad PCA, con clasificador SVM y búsqueda de parámetros. Se implementa un *pipeline* que contiene todos esos pasos del experimento. El mejor resultado es de **0.76** AUC para las características de fonación de la vocal U.

- AdaBoost con búsqueda de parámetros del número de estimadores. El mejor resultado es de **0.832** AUC para las características de fonación de la frase.

- Selección de atributos con *variance threshold* para diferentes clasificadores. El mejor resultado es de **0.884** AUC, con AdaBoost, para las características de fonación de la frase.

- Selección de atributos con el método K best, con el clasificador AdaBoost y con búsqueda de parámetros para el parámetro k del selector. El mejor resultado es de **0.74** AUC para las características de fonación de la frase.

Exp.	AUC
Experimenter	0.836
GS - SVM	0.744
GS - PCA - SVM	0.768
GS - VT - AdaBoost	0.832
VT - varios	0.884
KBest - varios	0.780
GS - KBest - AdaBoost	0.744

Tabla 5.4: Resumen resultados fase 1.

- Selección de atributos con el método K best para diferentes clasificadores. El mejor resultado es de **0.78** AUC para las características de fonación de la palabra gato.

Resultados

Vistos los resultados (véase Tabla 5.4), quedan muy lejos de los obtenidos en [25]. El mejor de los anteriores es el de selección de atributos con *variance threshold* para diferentes clasificadores, cuyo resultado es de **0.884** AUC, con AdaBoost, para las características de fonación de la frase. Se considera que este resultado por el mismo no es malo, pero sí lo es en comparación con el estado del arte. En el notebook comentado, se incluyen gráficos de barras y tablas que hacen más vistoso y comprensible el análisis de resultados del experimento.

Proyección gráfica de las características

Realizamos una proyección gráfica de las características de esta fase, para tener una idea de como se distribuyen los datos y tener una visión de si las clases son claramente separables o no. Se realiza en el *notebook* alojado en `src/Proyeccion_ccas_DisVoice.ipynb`. Como nuestros vectores de características tienen muchos atributos, realizamos la reducción de dimensionalidad con 4 métodos diferentes: análisis de componentes principales (PCA con y sin kernel), TSNE y análisis de discriminante linear (LDA). Llegamos a la conclusión que no hay una separación clara de las clases, únicamente se atisba una ligera separación con el la reducción realizada con el método LDA (ver Figura 5.6).

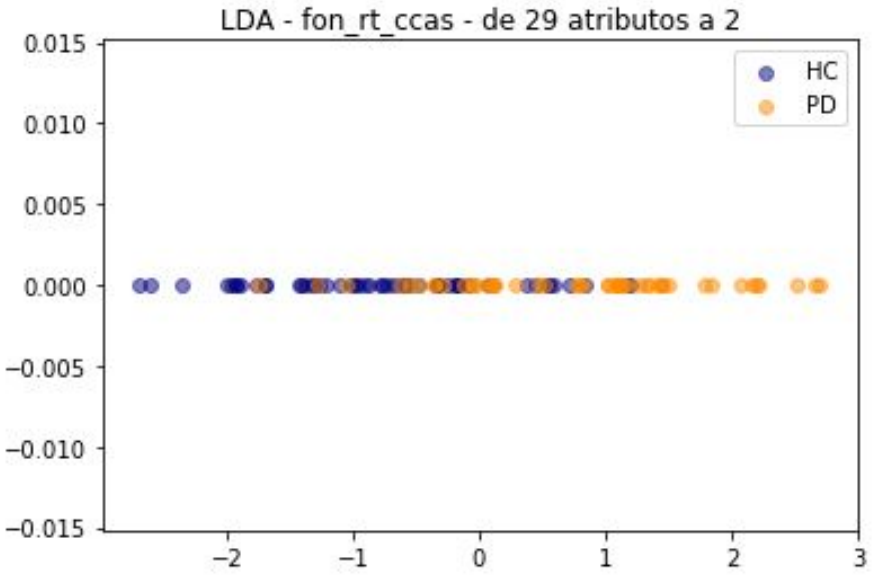


Figura 5.6: Proyeccion realizada con LDA.

5.6. Segunda Fase: Disvoice modificado

Como anteriormente hemos comentado, realizamos modificaciones sobre los datos anteriores con el objetivo de mejorar los resultados de los experimentos. Para ello, lo primero que realizamos fue añadir los atributos edad y sexo del paciente a nuestro vector de características. Por ello, tenemos los mismos 18 subconjuntos de características explicados en 5.5, pero cada instancia con dos atributos más: edad y sexo. Estos atributos fueron añadidos ya que les consideramos importantes a la hora de analizar físicamente la onda de la voz. Evidentemente habrá diferencias en medidas como frecuencias fundamentales o *Jitter* entre sexos, o entre personas mayores y personas jóvenes.

La segunda mejora fue separar las instancias del conjunto de datos entre hombres y mujeres, y realizar clasificadores diferentes para cada grupo. Se planteó de esta manera, debido a que [25] hace validación cruzada estratificada según dos elementos: Edad y clase (PD: *Parkinson Disease* o HC: *Healthy control*). Con la biblioteca utilizada para los experimentos, *Scikit-Learn*, solamente se puede estratificar según 1 atributo. Separando los conjuntos de datos por sexos y estratificando cada sexo por clase, estamos simulando esa validación cruzada estratificada por 2 atributos que utiliza ese artículo. De esta manera obtenemos los conjuntos de datos de medidas

de fonación articulación y prosodia, pero únicamente añadiendo el atributo edad, ya que el atributo sexo no hace falta: debido a la división por sexo todos son el mismo. Un problema que tenemos es que, como nuestros conjuntos de datos son pequeños (100 o 300 instancias), a la hora de dividir son todavía más pequeños: tendremos el doble de conjuntos de datos de la mitad de tamaño (50 0 150 instancias).

Por último, creamos unos conjuntos de datos con únicamente las características MFCC de los audios, estas son un subconjunto de las medidas de articulación (véase 5.5), que comprenden únicamente los atributos MFCC y sus dos primeras derivadas. Esto es debido a que son las características con las que mejor resultado se obtiene en el estado del arte, y son exactamente las mismas que las extraídas en [25]. Así que, para cada audio posible, hemos obtenido este conjunto de características, también dividido por sexo.

Por ello, hemos realizado 3 *notebooks* de experimentos en esta segunda fase: Uno para los experimentos con las características Disvoice más edad y sexo, otro para las características Disvoice de las mujeres más el atributo edad y otro para las características Disvoice de los hombres más la edad. Los conjuntos de datos de las características MFCC comentadas como tercera mejora, fueron divididos por sexo y participan en los experimentos de los *notebooks* de experimentos con mujeres y de experimentos con hombres.

Modelado del discurso

Se extraen las características de fonación, articulación y prosodia de cada audio comentado anteriormente, y posteriormente se añaden los atributos edad y sexo. La extracción de las características físicas se realiza con la herramienta Disvoice al igual que anteriormente. Los atributos edad y sexo se añaden con el extractor de características creado, que añade automáticamente al conjunto de datos los datos extra (en nuestro caso edad y sexo) que contiene un diccionario. Esta clase creada por nosotros, perimte tanto añadir esos atributos extra a un conjunto de datos ya extraído, como extraer las características de un audio y añadir esos atributos extra en el propio proceso de extracción directa del audio.

Para la extracción de las características MFCC, como son un subconjunto de las características de articulación, nos fijamos en los índices de las características en la documentación de Disvoice y extrajimos esas columnas indicadas.

La extracción de estas características se encuentra realizada y explicada en el notebook de la ruta `TFG-Neurodegenerative-Disease-Detection/src/`

`Extracción de características MODIFICADAS Disvoice.ipynb`.

Conjuntos de datos totales

Debido a lo que acaba de ser explicado, tenemos un total de **78** conjuntos de datos diferentes. 18 de ellos son igual que los de la anterior fase, pero añadiendo los atributos edad y sexo. 30 de ellos son los extraídos con Disvoice para las mujeres más la edad (18 anteriores + 12 de características MFCC). Los 30 restantes son los 30 extraídos de los hombres. El número de instancias de cada conjunto de datos varía. Para los divididos por sexo (60 conjuntos diferentes) serán la mitad que los de la primera fase: 50 para las palabras y la frase y 150 para las vocales. En cambio, para los que únicamente hemos añadido edad y sexo (18 diferentes) tendremos las mismas instancias que en la primera fase, 100 o 300. Se construyen clasificadores independientes para cada conjunto de los 78 diferentes, como hemos comentado anteriormente

Las características extraídas se guardan en formato numpy en los directorios: `src/CaracteristicasExtraidas/DivisionSexo/hombres`, `src/CaracteristicasExtraidas/DivisionSexo/mujeres` y `src/CaracteristicasExtraidas/EdadYSexo`.

Experimentos con clasificadores

Los experimentos de esta fase se encuentran agrupados en 3 *notebooks* diferentes. Uno para los experimentos con las características Disvoice más edad y sexo, otro para las características Disvoice de las mujeres más el atributo edad y otro para las características Disvoice de los hombres más la edad. Son los realizados en:

- `src/2os Experimentos A - Disvoice + Edad y Sexo.ipynb`
- `src/2os Experimentos B - Disvoice Mujeres.ipynb`
- `src/2os Experimentos B - Disvoice Hombres.ipynb`

Todos estos documentos contienen explicación detallada de los experimentos y visualización de resultados en multitud de gráficas y tablas para su mejor análisis y comprensión. Para todos los conjuntos de datos se han realizado varios tipos de experimentos al igual que anteriormente. Se han realizado por lo general los mismos experimentos que en la primera fase, con algunos nuevos experimentos en cada notebook que íbamos avanzando.

Para los experimentos de las características Disvoice más edad y sexo sin dividir por sexos (`TFG-Neurodegenerative-Disease-Detection/src/2os Experimentos A - Disvoice + Edad y Sexo.ipynb`) se han realizado la repetición de los mismos experimentos de la primera fase pero con estos nuevos datos(véase 5.5).

Para los 2 *notebooks* de experimentos de las características Disvoice divididas por sexos se han realizado la repetición de los mismos experimentos de la primera fase, más u nuevo experimento: Selección de características con K Best, clasificador *Gradient Boost Classifier* y búsqueda de parámetros.

Resultados

Por lo general, el AUC de los experimentos ha aumentado debido a los cambios realizados, pero sin ser un incremento muy notable, ni llegar a resultados parecidos a los comentados del estado del arte. El mejor resultado para los experimentos de las características Disvoice más edad y sexo sin dividir por sexos es el de selección de atributos con *variance threshold* para diferentes clasificadores, cuyo resultado es de **0.884** AUC, con AdaBoost, para las características de fonación de la frase. Coincide exactamente con los resultados de la primera fase. Es mera coincidencia, ya que en los demás experimentos, los resultados sí que han variado mejorando muy poco por lo general.

Para los experimentos con las mujeres conseguimos **el mejor resultado conseguido en todos los experimentos**. Lo hemos conseguido con selección de parámetros con K Best, AdaBoost y búsqueda de parámetros. Conseguimos un **0.908 AUC** con el conjunto de datos de características de articulación de la palabra *campana*. Cabe destacar que en todos los experimentos con la separación por sexos, se aumenta el rendimiento de los clasificadores: un 0,02 AUC el mejor caso y hasta un 0,05 AUC la media de todos los experimentos.

Para los experimentos con los hombres no conseguimos tan buen resultado como para las mujeres. El mejor resultado lo hemos conseguido con selección de parámetros con K Best, Gradien Boost Classifier y búsqueda de parámetros. Conseguimos un **0.867** AUC con el conjunto de datos de características MFCC de la palabra *atleta*, es decir, uno de los nuevos conjuntos de datos que hemos sacado en esta fase.

Los experimentos y resultados al detalle se explicarán de manera más extensa en el anexo de investigación y en los *notebooks*. Sin embargo, el resumen de los resultados se puede ver en la tabla 5.5. En las casillas donde

Exp.	+Ed-Sx	Muj	Hom
Experimenter	0.836	0.894	0.813
GS - SVM	0.721	0.725	0.769
GS - VT - AdaBoost	0.860	0.905	0.838
VT - varios	0.884	0.883	0.813
KBest - varios	0.759	0.816	0.838
GS - KBest - AdaBoost	- -	**0.908**	0.816
GS - KBest - Rand For	0.748	0.820	0.816
GS - KBest - GBC	- -	0.880	0.867

Tabla 5.5: Resumen resultados fase 2.

no hay resultados, es debido a que los experimentos les hemos realizado de manera adicional en los siguientes pasos (para hombres y mujeres) y no en anteriores *notebooks*.

5.7. Tercera Fase: VGGish

Como último recurso para mejorar los resultados y, además, dar una nueva visión a la extracción de características, añadimos la innovación al estado del arte de utilizar para la extracción una red neuronal con capas convolucionales preentrenada: **VGGish** (véase sección 5.4). Esta red ha sido utilizada para extraer características, 128 *embeddings* de cada audio. Estos *embeddings* no son interpretables.

A su vez, en esa biblioteca VGGish, también se provee un extractor de las características de estilo MFCC y espectros, ya que es parte del proceso para finalmente sacar los *embeddings*. Dado nuestro escepticismo surgido con le herramienta Disvoice, debido a los bajos resultados obtenidos comparando con el artículo de los mismos autores [25], también hemos generado conjuntos de datos que son MFCC y espectros. Por ello, hemos obtenido otra serie de conjuntos de datos de esta manera.

Cabe destacar que, con el modelo VGGish solo podemos sacar características para los audios de más de 0.975 segundos. Esto se debe al funcionamiento de VGGish. Para VGGish una instancia es un frame de 0.96s de un audio, no el audio entero. Por ello, saca características para cada ventana de 25ms (por eso el limite de 0.975s) con solapamiento de 10ms. Es decir, todas las palabras y muchas vocales no podrán ser procesadas por este modelo debido

a su corta duración (el 99 % de las palabras duran menos de un segundo y algunos audios de las vocales también). Por tanto, realizaremos la extracción de características para la frase y para las vocales que se pueda.

Modelado del discurso

VGGish tiene varios pasos. La entrada de la red VGGish debe ser una matriz de tamaño 96x64 y la salida un vector unidimensional de tamaño 128. Concretamente, 96 son las ventanas que analiza y 64 las características que saca el preprocesado de cada ventana. Por ello, el preprocesado devuelve un array de matrices de 96×64, que servirá de entrada a la red VGGish, cada una como ejemplo independiente. Por ello, si hacemos el proceso entero con un audio, al final con VGGish obtendremos una matriz de tamaño (duracion del audio/0.975)×128. Debido a que la entrada a nuestro modelo deberá ser una instancia, hemos optado por hacer la media y desviación de cada embedding obtenido con VGGish y así finalmente obtener un vector de tamaño 256. El esquema del proceso es el siguiente:

Audio wav → **|Preprocesado|** → MFCC y espectros para cada 0.96s → **|VGGish|**→ Embeddings → **|Media y desv|** → vector final (size 256)

Como hemos comentado también extraemos las características de estilo MFCC. Por ello, usamos únicamente la etapa de preprocesado para obtener los datos. Como hemos comentado, nos devuelve array de matrices de 96x64, es decir, devuelve un array de tamaño 3, de la forma numpy (duracion del audio/0.975),96,64. Esto quiere decir que en cada posición del array, tenemos una matriz de 96 filas y 64 columnas: una fila por cada ventana analizada y 1 columna por cada característica sacada. Dicho de otro modo, tendremos N matrices de 96x564 siendo N=(duracion del audio/0.975). Por ello, realizamos lo mimo que antes, primero aplanamos la primera dimension del array para que nos quede en análisis de las ventanas todos seguidos (de X,96,54 a X*96,64), sin división entre conjuntos de frames de 0.975 segundos. Y a continuación sacamos la media y desviación de las 64 características para obtener finalmente un array de tamaño 128. Concretamente el preprocesado realiza los siguientes pasos (explicado en el apartado *Input:Audio Features* de VGGish[11]):

- Todo el audio se remuestrea a 16 kHz mono.

[11] https://github.com/tensorflow/models/blob/master/research/audioset/vggish/README.md

- Un espectrograma se calcula utilizando magnitudes de la Transformada de Fourier a corto plazo con un tamaño de ventana de 25ms, un salto de ventana de 10ms y una ventana de Hann periódica.
- Un espectrograma de mel, ver sección 3.6, se calcula al mapear el espectrograma a 64 casillas de mel que cubren el rango de 125-7500 Hz.
- Se calcula un espectrograma log mel estabilizado aplicando log (espectro de mel + 0.01) donde se usa el desplazamiento para evitar tomar un logaritmo de cero.
- Estas características se enmarcan en ejemplos no superpuestos de 0.96s, donde cada ejemplo cubre 64 bandas mel y 96 frames de 25 ms con windowing 10ms cada una.

El esquema del proceso es el siguiente:

Audio wav → **|Preprocesado|** → MFCC y espectros para cada 0.96s → **|Media y desv|** → vector final(size 128)

La extracción de estas características se encuentra realizada y explicada en el *notebook* de la ruta `src/vggish/Extracción cararterísticas VGGish.ipynb`.

Conjuntos de datos totales

Tras una identificación de los audios cortos (menores de 0.975) realizada en el *notebook* de extracción de características VGGish, hemos llegado a la conclusión que, en las vocales, hay los siguientes audios cortos:

- A: 18/300 cortos. 282 restantes para clasificación.
- E: 16/300 cortos. 284 restantes para clasificación.
- I : 19/300 cortos. 281 restantes para clasificación.
- O: 38/300 cortos. 262 restantes para clasificación.
- U: 46/300 cortos. 254 restantes para clasificación.

Concluimos que podemos crear clasificadores con los audios de las vocales, nos queda un porcentaje de audios bastante alto respecto al original. De las palabras: todas las palabras tienen menos de 6 audios de más de la

duración necesitada. Por lo tanto, no se podrán crear clasificadores con las palabras. Por ello, obtenemos un total de **12** conjuntos de características: 6 extraídas con la red neuronal VGGish (frase y 5 vocales) y 6 para los MFCC y espectros del preprocesado (frase y 5 vocales).

Las características extraídas se guardan en formato numpy en los directorios `src/CaracteristicasExtraidas/vggish/embbedings` y `/src/Caracteristicas Extraidas/vggish/espectros`.

Experimentos con clasificadores

Los experimentos de esta fase se encuentran agrupados en 2 *notebooks* diferentes. Uno para los experimentos con los *embeddings* y otro para las características MFCC y espectros. Son los realizados en:

- `src/3os Experimentos A - VGGish Embeddings.ipynb`
- `src/3os Experimentos B - VGGish Espectros.ipynb`

Todos estos documentos contienen explicación detallada de los experimentos y visualización de resultados en multitud de gráficas y tablas para su mejor análisis y comprensión. Para todos los conjuntos de datos se han realizado varios tipos de experimentos al igual que anteriormente. Se han realizado por lo general los mismos experimentos que en la segunda fase, con algunos nuevos experimentos en cada notebook que íbamos avanzando.

A los experimentos de las anteriores fases hemos añadido experimentos con el clasificador MLP, *Multi-Layer Perceptron*. Hemos añadido ese clasificador con los parámetros por defecto al ya comentado *experimenter* y, además, hemos añadido el experimento que consiste en un selector de parámetros K best, con el clasificador MLP y búsqueda exhaustiva de parámetros.

Resultados

Los resultados de este experimento (véase 5.6), están en la magnitud de rendimiento de los anteriores: la mayoría de 0.8 a 0.9 AUC y sin llegar a los resultados del estado del arte. Para los experimentos con los *embeddings* conseguimos un **0.88 AUC** con el conjunto de datos de características de la frase con el clasficador MLP con 10 neuronas y demás parámetros por defecto. Para los experimentos con los espectros los resultados son un poco peores que para los *embeddings*, pero con poca diferencia significativa. Conseguimos un **0.861 AUC** con el conjunto de datos de características

Exp.	Embedd	MFCC
Experimenter	0.888	0.829
GS - SVM	0.876	0.861
GS - VT - AdaBoost	0.772	0.818
VT - varios	0.832	0.831
KBest - varios	0.760	0.761
GS - KBest - AdaBoost	0.765	0.788
GS - KBest - Rand For	0.798	0.842
GS - KBest - GBC	0.804	0.805
GS - KBest - MLP	0.856	0.768

Tabla 5.6: Resumen resultados fase 3.

de la vocal A con el clasficador SVM con búsqueda de parámetros C y lambda. Cabe destacar que para las características de los MFCC y espectros extraídas con VGGish, para todos los experimentos se ha obtenido el mejor resultado con las características de la vocal A. Esto se aproxima más a lo que dice el estado del arte, que las vocales sostenidas, en concreto la A, pueden ser suficientes para esta tarea de clasificación.

5.8. Conclusión de experimentos con clasificadores

Tras la realización de los experimentos podemos llegar a una serie de conclusiones. Llegamos a resultados que, si no fuera por la comparación con los del estado del arte, serían significativos. Sin embargo, al tener como referencia los resultados de [25], se han realizado infinidad de experimentos con el objetivo de conseguir acercarse a los resultados. Como hemos ido viendo, en total hemos realizado experimentos con 108 conjuntos de datos diferentes. Con cada conjunto hemos realizado una serie de experimentos, y la mayoría de experimentos incluían internamente en entrenamiento de varios clasificadores. Por ello, se considera abordado la etapa de construcción de clasificadores, no llegando a saber exactamente porqué no se llegan a los resultados del estado del arte, habiendo seguido el proceso casi con similitud total. Cabe destacar que, sí coincidimos con el estado del arte en que, en la mayoría de experimentos, conseguimos los mejores resultados con las características extraídas de la frase, o con las características extraídas de las vocales, aunque, paradójicamente, justo el mejor resultado se consigue

con las características extraídas de la palabra campana. Los resultados de todos los experimentos, con medida AUC y con qué dataset se obtiene la mejor medida están plasmados en 5.7.

Con cada modificación a los datos originales de Disvoice (adición de edad y sexo, separación por sexo), se han conseguido mejoras significativas en algunos de los experimentos. Concretamente, los resultados más altos se obtienen a la hora de separar por sexos, siendo el resultado más alto de 0.9 AUC para el conjunto de datos de articulación de la palabra campana extraído con Disvoice para las mujeres, con selección de características con K best y clasificador AdaBoost. La novedad de usar VGGish, no destaca por su rendimiento, sino que destaca por ser un método novedoso en la extracción de características, con el que conseguimos igualar los resultados con las otras técnicas (nuestros resultados).

Destacamos el siguiente aspecto: como hemos dicho antes, no llegamos a los rendimientos del estado del arte siguiendo el mismo proceso. Nuestro proceso de entrenamiento ha sido revisado varias veces y está realizado de manera correcta. La herramienta Disvoice es una herramienta del propio grupo de investigación de [25]. Sin embargo, un aspecto nos destaca bastante. Cuando las características MFCC y los espectros los extraemos con el preprocesado de VGGish en vez de con Disvoice, justo en el experimento de búsqueda de parámetros SVM con kernel gaussiano (GS - Orozco - SVM en nuestras tablas de resultados), sufrimos un incremento de el AUC de manera considerable, comparado con el mismo experimento con otros datasets. Aparte, la búsqueda de parámetros definida en [25], dice que buscan el parámetro C y λ en los rangos de [1, 10000]. Sin embargo, en los experimentos de la tercera fase, hemos añadido el valor *auto* para λ , que en *Scikit-Learn* se corresponde con 1/número_atribs. Por ello, también se llega a la conclusión, y se sugiere, que en experimentos futuros se realice la búsqueda del parámetro λ en el rango [1/1000, 1] en vez en el rango [1, 10000].

Por ello concluimos que el mejor clasificador es el entrenado con el conjunto de datos de articulación de la palabra campana extraído con Disvoice para las mujeres, con selección de características con K best y clasificador AdaBoost. También queremos destacar el clasificador MLP de 10 neuronas, entrenado con las características *embeddings* de la frase, que sin dividir por sexos los datos llega a un rendimiento de 0.888 AUC.

Resultados	Disvoice		Disvoice + Edad y Sexo		Disvoice + Edad - Mujeres	
	Dataset	Auc	Dataset	Auc	Dataset	Auc
Ex - Atribs Defecto	*ADA\fon_rt*	0,836	*ADA\fon_rt*	0,836	*ADA\art_w_campana*	0,883
Ex - Atribs Norm	*ADA\fon_rt*	0,836	*ADA\fon_rt*	0,836	*GBC\art_w_campana*	0,894
GS - Orozco (SVM)	*fon_v_U*	0,744	*fon_v_U*	0,721	*fon_w_gato*	0,725
GS PCA - SVM	*fon_v_U*	0,768	-	-	-	-
GS - VT + AdaBoost	*fon_rt*	0,832	*fon_rt*	0,86	*art_rt*	**0,905**
VT - varios	*ADA\fon_rt*	**0,884**	*ADA\fon_rt*	**0,884**	*ADA\art_w_campana*	0,883
Kbest - varios	*RF\fon_w_gato*	0,78	*SVM\fon_v_u*	0,759	*GBC\art_w_gato*	0,816
GS - Kbest - Ada	*fon_rt*	0,744	-	-	*art_w_campana*	**0,908**
GS - Kbest - RF	-	-	*art_rt*	0,748	*fon_v_A*	0,82
GS - Kbest - GBC	-	-	-	-	*art_w_campana*	0,88
GS - Kbest - MLP	-	-	-	-	-	-
Media de todos		0,803		0,806		0,8571

Resultados	Disvoice + Edad - Hombres		VGGish Embeddings		VGGish MFCC-Espectros	
	Dataset	Auc	Dataset	Auc	Dataset	Auc
Ex - Atribs Defecto	*ADA\art_w_campana_ON*	0,813	*MLP\vocal_A*	0,822	*MLP\vocal_A*	0,843
Ex - Atribs Norm	*ADA\art_w_campana_ON*	0,813	*MLP\read_text*	**0,888**	*MLP\vocal_A*	0,829
GS - Orozco (SVM)	*fon_v_I*	0,769	*read_text*	0,876	*vocal_A*	**0,861**
GS PCA - SVM	-	-	-	-	-	-
GS - VT + AdaBoost	*art_w_campana_ON*	0,838	*vocal_A*	0,772	*vocal_A*	0,818
VT - varios	*ADA\art_w_campana_ON*	0,813	*MLP\read_text*	0,832	*GBC\vocal_A*	0,831
Kbest - varios	*GBC\art_w_gato_ON*	0,838	*MLP\read_text*	0,76	*MLP\vocal_A*	0,761
GS - Kbest - Ada	*art_w_campana_ON*	0,816	*vocal_A*	0,765	*vocal_A*	0,788
GS - Kbest - RF	*art_w_braso*	0,816	*read_text*	0,798	*vocal_A*	0,842
GS - Kbest - GBC	*art_w_atleta_OF*	**0,867**	*vocal_A*	0,804	*vocal_A*	0,805
GS - Kbest - MLP	-	-	*read_text*	0,856	*vocal_A*	0,768
Media de todos		0,8203333		0,8173		0,8146

Figura 5.7: Resumen de los resultados de los experimentos.

Roc

Se ha realizado la muestra de la curva ROC del mejor experimento que hemos realizado. La codificación de este proceso se encuentra en la ruta `src/ROC - Best.ipynb`. Podemos ver la curva en 5.8.

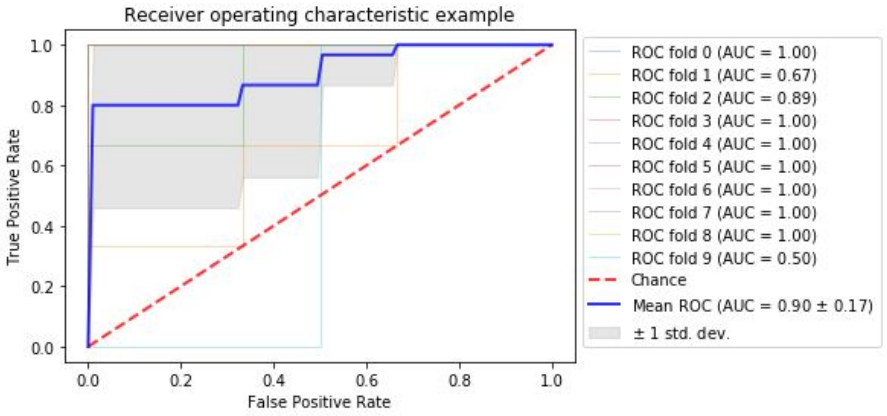


Figura 5.8: Curva ROC del mejor experimento realizado.

5.9. Aplicación de escritorio

Para concluir, se ha realizado una aplicación de escritorio, llamada PDDetection, con interfaz gráfica, utilizando tkinter de Python, con el objetivo de cómo podría aplicarse el resultado de investigación a un producto final. En ella se podrá analizar un audio y, tras procesarlo, que nos diga la probabilidad de que la persona de ese audio tenga parkinson o no, y se muestren las gráficas de amplitud de onda y de espectrograma de frecuencias. Éstas últimas gráficas, son importantes a la hora de analizar un audio, tal y como se explica en [12], la variación de la amplitud y la frecuencia en pacientes de Parkinson, es más abrupta que en personas sanas.

Para ello, se ha realizado una única pantalla, en la que nos deje realizar las siguientes funciones:

- Cargar y borrar audios formato wav en la aplicación.
- Reproducir los audios mediante un reproductor típico (reproducir, pausar, volumen..).
- Diagnosticar si la persona de un audio tiene Parkinson y con qué probabilidad.
- Introducir el sexo y la edad de la persona del audio.
- Mostrar gráfica de ayuda al facultativo: amplitud de la onda del audio.
- Mostrar gráfica de ayuda al facultativo: espectrograma de frecuencias del audio.

El funcionamiento de una ejecución ejemplo sería el siguiente: cargar el audio (o los audios) a la aplicación (guardar la ruta en una lista para su predicción o reproducción), insertar sexo y edad del paciente, predecir (se extraen los *embeddings* del audio con VGGish, se añaden los datos edad y sexo y se predice con el clasificador), mostrar resultados de predicción junto con ambas gráficas. Podemos ver las ventanas en la siguiente captura 5.9.

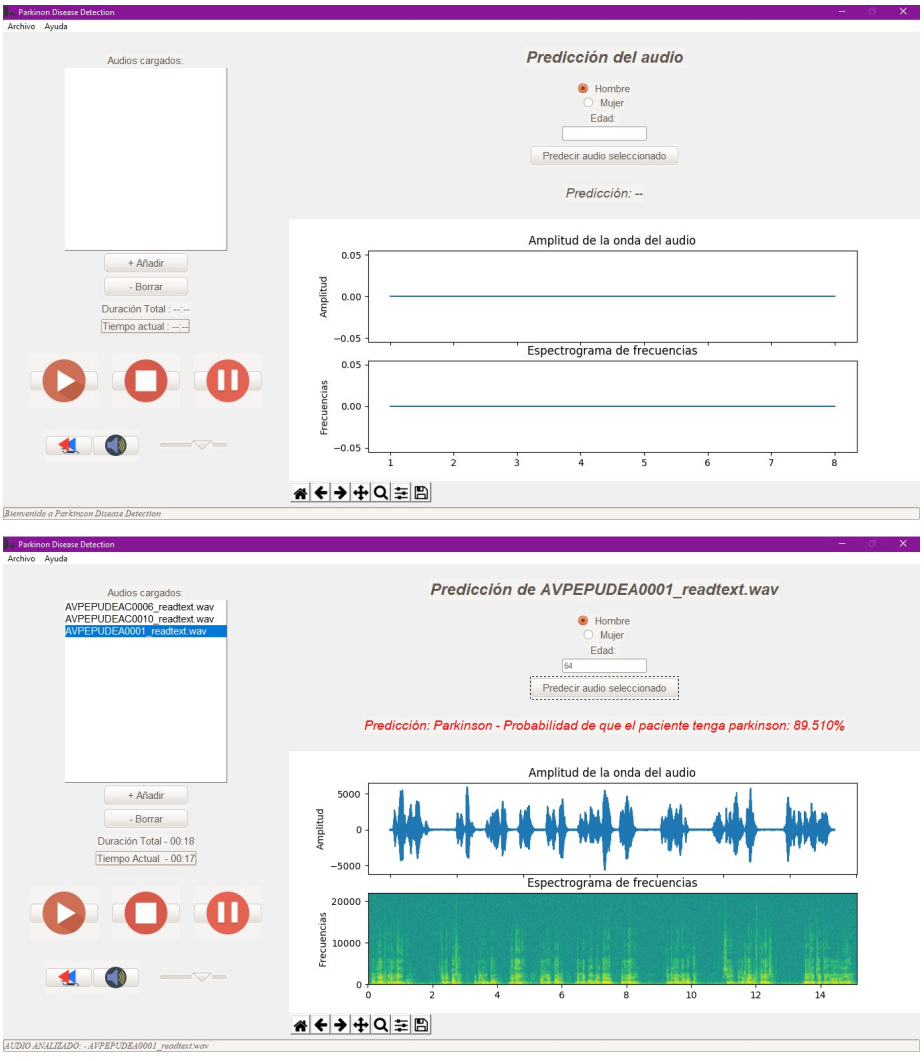


Figura 5.9: Ventanas incial y tras ejecución de la aplicación

Se ha realizado siguiendo el patrón de diseño mediador fachada. Cabe destacar que esta fase del proyecto, aunque obviamente es la más vistosa, no es el grueso principal del mismo. Esta aplicación es un ejemplo de cuales

son las aplicaciones de esta investigación, por lo que será una aplicación base, sin demasiados detalles de perfeccionamiento para la explotación.

El clasificador utilizado ha sido el clasificador MLP de 10 neuronas, entrenado con los **embeddings** extraídos de la frase, que tiene un AUC de 0.888. Hemos escogido este clasificador, ya que es el que mejor se comporta con los audios de ambos sexos y porque la extracción de características con la red VGGish, se realiza de manera mucho más rápida que con la herramienta Disvoice. Como hemos comentado esta es una aplicación base, ya que, en vez de utilizar el mismo clasificador para ambos sexos, se podría aver generado una sentencia condicional en la que, dependiendo del sexo del paciente y del tipo de audio (vocal palabra o frase), clasificara con diferentes clasificadores.

Trabajos relacionados

En este apartado se comentará el estado del arte de la materia y algunos trabajos relacionados.

Para comenzar, comentaremos que hay dos grupos de investigación importantes en este campo de estudio. El primer grupo sería el encabezado por el autor M. A. Litlle de la Universidad de Aston, Birmingham, Inglaterra. Este autor a su vez encabeza un proyecto para la obtención de una gran base de datos de 10000 audios obtenidos de pacientes con Parkinson. Esta iniciativa es conocida como Parkinson Voice Inititative[12]. Este conjunto de autores tiene varios artículos [33], [34] y el más citado [20]. Estos artículos aportan un buen análisis sobre las características a extraer de los audios (lineares y no lineares) y algoritmos de selección de características. Los aspectos más relevantes de cada artículo son los siguientes:

- *Suitability of dysphonia measurements for telemonitoring of Parkinson's disease* [20].
  Este artículo trata sobre la aplicación de diferentes medidas estándares (lineales) y no estándares (no lineales) de disfonía para la clasificación automática de personas con Parkinson y personas sanas. Además, añaden otra nueva medida, calculada en este mismo artículo, llamada PPE, *pitch period entropy*. Se centra en responder a la pregunta de qué características son las mejores para la detección del Parkinson. Para ello utiliza únicamente audios de vocales sostenidas (28 PD y 8 HC). Se calculan un total de 17 medidas de disfonía de cada audio usando diferentes software como Praat [2]. Posteriormente, se hace un

[12] http://www.parkinsonsvoice.org/

análisis de correlación entre las medidas obtenidas, y de aquellos pares que tienen un coeficiente de correlación mayor del 95 %, se elimina una. Después de este análisis se obtienen 10 características: *jitter* medido diferencia absoluta, *jitter* medido como media entre ciclos, APQ, *shimmer* (calculado como la diferencia absoluta promedio entre las amplitudes de los períodos consecutivos), HNR, NHR, RPDE, DFA, la dimensión de correlación y el comentado PPE.

En el siguiente paso del proceso, utilizan el algoritmo de aprendizaje supervisado SVM con kernel de base radial, para construir el modelo. Esto se realiza con cada uno de los 1023 diferentes subconjuntos posibles de las 10 características, $\sum_{i=1}^{10} \binom{10}{i}$, para encontrar el mejor subconjunto posible. Esto es debido a que los autores consideran que es un número pequeño de conjuntos y se pude hacer búsqueda exhaustiva. Se llega a la conclusión que el **mejor conjunto de características es el formado por HNR, RPDE, DFA, y PPE** que devuelve una precisión del **91 %**, seguido en precisión por el conjunto completo de las 10 características.

- *Accurate Telemonitoring of Parkinsons Disease Progression by Noninvasive Speech Tests* [33].
  En este artículo trata sobre cómo **el objetivo de la clasificación del Parkinson puede ser de regresión**. En vez de clasificar entre personas sanas (HC) y con Parkinson (PD), lo que se hace es medir el nivel de parkinson de una persona usando la escala UPDRS (*Unified Parkinson's Disease Rating Scale*). Para ello saca un total de 16 características de audios con vocales sostenidas, de las que se hace un análisis de correlación pero no se elimina ninguna.

  Para la construcción de modelos se utilizan 4 técnicas diferentes: 3 de ellas de regresión lineal (LS, IRLS y LASSO) y una de regresión no lineal (CART's). Se llega a la conclusión de que los métodos lineales no dan malos resultados, siendo el IRLS el mejor de los 3. Sin embargo, el que mejor precisión da es el método CART. Los errores son de 8.47 ± 0.17 para UPDRS total con IRLS y de 7.52 ± 0.25 usando el método CART. A parte de estos resultados, en el artículo se realiza un análisis de la correlación de las características fijándose en los coeficientes devueltos por el método LS. En ellos se puede ver como las características altamente correlacionadas tienen magnitud similar y signo opuesto.

- *Novel Speech Signal Processing Algorithms for High-Accuracy Classification of Parkinson's Disease* [34].

En este artículo el objetivo es clasificar entre personas sanas y con PD a partir de audios de vocales sostenidas **utilizando un gran conjunto de medidas de disfonía extraídas de los audios**. Un punto novedoso de este experimento, es que hasta ese momento, se habían elegido conjuntos de pocas (<20) medidas de disfonía y medido su correlación. En este artículo, trata un total de 132 medidas lineares y no lineares de cada audio a las que se aplicará diferentes algoritmos de selección de características (algoritmos FS) para elegir varios conjuntos (uno por cada algoritmo). Los algoritmos de selección de características han sido: LASSO, RELIEF, mRMR y LLBFS. Cada algoritmo de selección de características ha elegido un conjunto diferente, siendo básicamente medidas como *jitter* y *shimmer*, variantes de medidas de ruido, MFCCs y medidas no lineales. Se ha analizado el número de características que comprende cada conjunto y se ha llegado a la conclusión de que cada conjunto tendrá 10 medidas únicamente. Según se explica, esto es debido a que usando más de 22 características se tiende al sobreajuste y que tampoco hay mucha mejoría usando 22 características en lugar de 10.

Posteriormente cada uno de esos conjunto de características ha sido usado para construir dos clasificadores con dos métodos distintos: random forest y SVM con kernel Gaussiano. Se obtienen resultados que mejoran cualquier artículo de de accurracy del 97.7 % utilizando las 132 características con SVM. Sin embargo y como acabamos de explicar anteriormente, en el mismo artículo acusa del accuracy tan alto a un posible sobreajuste al usar tantas caractarísticas. Por lo explicado anteriormente, cada algoritmo FS escoge un conjunto de 10 medidas elegidas. El mejor resultado se da para el subconjunto de características escogido por el algoritmo RELIEF y usando SVM, presentando una accuracy del 98.6 % frente al RF que con el mismo conjunto llega a un accuracy del 93.5 %.

Para terminar con el resumen de este artículo, indicar que la importancia está en que se trabaja con un número grande de medidas (132), llegando a la conclusión de que los subconjuntos idóneos tendrán alrededor de 10 características ya que tienen resultados parecidos a los conjuntos grandes pero generalizarán mejor que éstos.

El segundo grupo de investigación y para nosotros más importante, ya que seguimos los pasos para la extracción de características, es el liderado por J. R. Orozco-Arroyave de la Universidad de Antioquía, Colombia. Estos artículos son más recientes que los de el anterior grupo de investigación y,

aunque tienen menos repercusión, son autores que trabajan y colaboran con el MIT (Massachusetts Institute of Technology) el cual tiene gran prestigio. El artículo más importante del grupo y que aporta una nueva visión a esta materia es [25]. Esta nueva visión es debido a que toma en cuenta más audios aparte de únicamente de la vocal sostenida. También tiene en cuenta varios idiomas. Es precisamente de este artículo del que hemos querido seguir los pasos a la hora de determinar qué características sacar de los audios y cómo sacarlas. También valorar y agradecer a este grupo de investigación la cesión de los audios descritos en [24], un corpus de audios de vocales sostenidas, monólogos, frases y palabras recopilados de un total de 100 personas (50 sanas y 50 PD) que contiene aproximadaamente 4200 audios. Analizamos el artículo más importante de este grupo:

- *Automatic detection of Parkinson's disease in running speech spoken in three different languages* [25].
  En este artículo clasifica entre personas sanas y con PD y se incorporan a la materia varias novedades relacionadas con los tipos de audios a utilizar.La primera de ellas es utilizar audios de monólogos, texto leído y palabras aparte únicamente de la vocal sostenida. Esto es debido a que la mayoría de los experimentos se basan en vocales sostenidas, obviando los consejos de los neurólogos respecto a que las consonantes requieren un mejor control de los movimientos de órganos (la lengua por ejemplo) y de ahí se puede sacar mucha más información. Por ello se utiliza un método para la caracterización de las señales de voz, basado en la segmentación automática de las expresiones en cuadros con voz y sin voz.

  La otra gran novedad que incorpora es la utilización de corpus de audios de varios idiomas a la vez, en este caso 3 diferentes. Esto se utiliza, por ejemplo, para lo que ellos mismos llaman *cross-language experiments*. En estos experimentos, el entrenamiento del modelo se hace en un idioma y el testeo con otro idioma diferente. En este artículo se realizan diferentes experimentos, construyendo varios modelos para cada tipo de audio y así evaluar qué tipo de audio es el mejor para la clasificación. Se comienza haciendo un preprocesado de los audios donde los silencios iniciales y finales son eliminados. Se prosigue dividiendo los audios en partes con voz y partes sin voz (estos frames con voz y sin voz se corresponden a los frames donde el software Praat [2] detecta que hay discurso y donde no lo hace). Para acabar el preprocesado se eliminan los fragmentos menores a 40 ms.

  A continuación empiezan con la extracción de características de los

audios. Se sacan 3 grupos de medidas diferentes con las que se construyen 3 modelos diferentes: 2 sacando diferentes tipos de medidas de los fragmentos con voz y 1 sacando medidas de los fragmentos sin voz. De los fragmentos con voz, se sacan por un lado 64 medidas prosódicas (*jitter*, *shimmer*...) con sus estadísticos (media, desviación, coeficiente de asimetría y curtosis). Por otro se sacan NHR, NNE, GNE, *F1*, *F2* y las 12 MFCC con sus estadísticos. De los frames sin voz se hace un análisis cepstral y de energía para calcular las 12 MFCC y 25 BBE con sus estadísticos.

Después se hace la clasificación utilizando SVM con kernel Gaussiano, que utiliza *grid search* para los parámetros C que mide la estrictez del margen y γ que es un parámetro del kernel Gaussiano utilizando como medida objetivo *accuracy*. Se utiliza una técnica de validación cruzada 10-fold.

Como resultado obtenemos 3 modelos diferentes para cada tipo de audio. De cada modelo saca 4 medidas para poderles comparar: *accuracy*, *sensitivity*, *specificity* y *AUC* (área bajo la curva de ROC). En el artículo compara los experimentos por tipo de audio. El mejor rendimiento lo consiguen, el texto leído, ya que consigue, para el idioma castellano y con el modelo construido con las medidas extraídas de los frames sin voz, un accuracy del 97 % y un AUC del 99 %.

Ahora trataremos dos trabajos que utilizan un conjunto de audios que se nos ha prestado, aunque finalmente no hemos trabajado con el porque [24] es más completo. Estos trabajos corresponden a Giovanni Dimauro de la Universidad de Bari, Italia. En el segundo de estos trabajos se aborda una perspectiva diferente para la diagnosis de Parkinson, utilizando sistemas Speech-To-Text:

- *VoxTester, software for digital evaluation of speech changes in Parkinson disease* [6].
  Este artículo describe la realización de una herramienta software para la ayuda en la evaluación de los cambios y variaciones en la voz de pacientes con la enfermedad del Parkinson. El funcionamiento de la herramienta es el siguiente: insertamos una serie de audios del paciente, la herramienta analiza el audio y nos devuelve 4 gráficas que un experto deberá interpretar para la evaluación de la enfermedad. Las gráficas de salida son las siguientes: gráfica del DDK *rate*, gráfica de la intensidad vocal y duración del discurso, gráfica del espectro de frecuencias vocal y gráfica del nivel de presión vocal. Estos parámetros

Artículos	Clasificación PD/HC	Regresión Mapeo UPDRS	Otro análisis
[20] [34] [25]	X		
[33]		X	
[6] [5]			X

Tabla 6.7: Objetivo de cada artículo

sacados de la voz son importantes ya que pueden ser indicadores de la enfermedad del Parkinson. Por ejemplo, un rango de frecuencias de la voz pequeño, una frecuencia fundamental baja o un decaimiento notable de la intensidad de la voz en el discurso serán una de las características del discurso para personas con esta enfermedad.

En este experimento no se utiliza en ningún momento técnicas de clasificación ni tampoco la herramienta discierne por ella misma si la persona del audio tiene una enfermedad o en qué grado. Lo único que hace es analizar la onda sonora y mostrar algunas características, por lo que no se alinea del todo con nuestros objetivos.

- *Assessment of Speech Intelligibility in Parkinson's Disease Using a Speech-To-Text System* [5].
  Este experimento trata de resolver el problema del anterior (valuación de los cambios y variaciones en la voz de pacientes con la enfermedad del Parkinson) con otra línea. Para ello utilizará sistemas de reconocimiento de voz y transcripción a texto (*Speech-To-Text Systems*) para evaluar la inteligibilidad del discurso de las personas con Parkinson. El proceso será el siguiente: la persona con Parkinson habla a un sistema STT, el texto de la transcripción se pasa a un programa que evalúa su acierto y finalmente ese programa devuelve el porcentaje de fallo en el reconocimiento de voz respecto a la frase objetivo. El objetivo de este proyecto ha sido comparar los fallos de reconocimiento en las palabras entre 3 grupos de personas: jóvenes sanos, mayores sanos y pacientes de la enfermedad del Parkinson. La conclusión a la que llegan los investigadores es que había mucho mas fallo en el reconocimiento de palabras en los pacientes de la enfermedad del Parkinson.

Conclusiones y Líneas de trabajo futuras

7.1. Conclusión

Respecto a los requisitos iniciales del proyecto, pienso que se han cumplido en su mayor medida. Dado a que este proyecto ha sido un proyecto de investigación, hay determinados objetivos que han podido ir cambiando a lo largo del proyecto. Por ello, los objetivos principales de realizar una investigación extensa y documentada y, finalmente, realizar una aplicación base de ejemplo, se han cumplido.

Cabe destacar que aunque evidentemente todo el proceso de minería de datos y realización de experimentos fue arduo, los primeros pasos de la investigación les considero como los más complejos. Esto es debido a que se quería construir una buena base bien documentada de la investigación y así, poder dar los pasos de manera más firme. Es por eso que las tareas de resumen del estado del arte, identificación de metodologías y adquisición de un conjunto de audios con el que poder trabajar fueron muy costosos por diversos motivos (dificultad de encontrar metodologías, sin respuesta de dueños de las bases de datos...).

En lineas generales, en este trabajo se ha desarrollado una serie de experimentos sobre el tema relacionado de análisis de audios para la detección del Parkinson bastante extenso. Hemos obtenido muchos conjuntos de características de los audios (108 diferentes, extraídos de diferentes formas) con los que se han hecho muchos experimentos. Cada paso que hemos ido dando en el proyecto, ha sido con el objetivo de ir mejorando los resultados obtenidos por nosotros.

Cabe destacar que los resultados por si solos no son malos, incluso están en la medida de los resultados de algunos artículos (tabla de resultados del estado del arte recogida en [34]). Sin embargo, al estar siguiendo en este proyecto el proceso de [25], con el mismo conjunto de audios y las mismas herramientas, cualquier resultado que estuviera lejos del 0,98 o 0,99 AUC nos ha parecido 'bajo'. Destacamos que la base de datos es pequeña ya que, aunque haya 2000 audios, como ya se ha comentado no se pueden entrenar modelos con todos los audios a la vez, hay que dividirlos por tipos, lo que hace que tengamos conjuntos de datos de 50 a 300 instancias.

Si nos olvidamos de la diferencia de resultados con ese artículo concreto, ha sido una labor de experimentación en la labor de Minería de Datos muy satisfactoria. Se han comprendido todas las etapas de la misma, desde la recopilación de datos y la extracción de características, hasta las labores finales de interpretar los resultados. La gran cantidad de diversos experimentos y técnicas utilizadas han servido para aprender en profundidad cómo funcionan los diferentes algoritmos y herramientas de este campo.

En resumen, tanto con la labor de investigación y su documentación, como con la labor de realización de experimentos, como con la realización aplicación final estoy satisfecho.

7.2. Líneas futuras

Se sugieren varios caminos diferentes de investigación, así como varios detalles a mejorar y desarrollar del proyecto:

- Primeramente, se sugiere que, en vez de realizar una tarea de clasificación entre PD y HC, se realicen **experimentos de regresión sobre la escala UPDRS**. Es decir, en vez de únicamente decir si una persona tiene Parkinson o no, realizar una regresión para ver que nivel de Parkinson tiene en la escala UPDRS. Abordar el problema de esta manera, se empezó a ver en [33]. Para esta tarea de regresión se podrá experimentar con todos los conjuntos los conjuntos de características extraídos, incluidos los *embeddings*. En principio, iba a ser también uno de los objetivos del proyecto. Sin embargo, al no llegar al rendimiento esperado en clasificación, se optó por realizar más experimentos de clasificación en vez de comenzar con regresión.

- Con objetivo de mejorar los resultados de los experimentos, se podrán probar diferentes elementos que mejoren su resultado. Uno de ellos es

probar otros selectores de atributos. En este proyecto hemos probado PCA, *Select K Best* y *Variance Treshold*. PCA es un reductor de dimensionalidad. Los otros dos son algoritmos de selección de atributos que se basan en cada atributo individual para su elección. Por ello, como mejora se podrían elegir nuevos selectores de atributos, basados en modelos de aprendizaje (i.e. *SelectFromModel* de *Scikit-Learn*) o basados en grupos de atributos (i.e. *Recursive feature elimination* de *Scikit-Learn*).

- Otra perspectiva diferente puede ser **tratar de manera diferente los *embeddings***. Como hemos comentado, VGGish saca un vector de características para cada fragmento de 0.975s de cada audio. Nuestra solución ha sido hacer la media y desviación para obtener un único vector de características de todos los fragmentos de cada audio. Sin embargo se podrán probar otras perspectivas.

  - La primera que se sugiere es, además de sacar la media y desviación, obtener también la **curtosis y el coeficiente de asimetría** (ver sección 3.3). Esto ayudará a no desechar tanta información, y puede ser que ayude a mejorar los resultados.
  - Otra perspectiva que se sugiere es la siguiente: clasificar cada vector devuelto por VGGish por separado, como si fuera un audio diferente, y devolver para ese audio la predicción mayoritaria de las predicciones de sus segmentos. Es decir, para un audio, clasificar cada 0.975s que analiza VGGish por separado de manera independiente y devolver la predicción mayoritaria.

- Por otro lado, de VGGish únicamente hemos usado la red de extracción de características. También podemos probar a **modificar la red de clasificación** del proyecto VGGish para que saque 2 salidas, en vez de las 600 que ahora tiene, y probarla sobre nuestro problema.

- Una mejora que se puede añadir a los experimentos actuales es la siguiente. En el experimento que imita al realizado en [25], se utiliza un SVM con kernel gaussiano. La búsqueda de parámetros definida en dicho artículo, dice que buscan el parámetro C y λ en los rangos de [1, 10000]. Sin embargo, en los experimentos de la tercera fase, hemos añadido el valor *auto* para λ , que en *Scikit-Learn* se corresponde con 1/número_atribs. Por ello, también se llega a la conclusión, y se sugiere, que en experimentos futuros se realice la **búsqueda del parámetro** λ en el rango [1/1000, 1] $\rightarrow$ (1/1000, ..., 1/100, ..., 1/10, ...,1), en vez en el rango [1, 10000].

- Para terminar, se sugiere la mejora de la aplicación. A parte de mejoras estéticas, se sugiere el uso de **diferentes clasificadores en función de los detalles** del paciente de entrada y del audio (vocal, palabra o frase). Actualmente utilizamos un clasificador único. La mejora sería derivar la clasificación a un clasificador u a otro en función del sexo, edad y tipo de audio introducido.

Bibliografía

[1] *Speech Processing, Transmission and Quality Aspects (STQ); Distributed speech recognition; Front-end feature extraction algorithm; Compression algorithms*, volume ETSI ES 201 108 V1.1.3. European Telecommunications Standards Institute, September 2003.

[2] Paul Boersma and David Weenink. Praat, a system for doing phonetics by computer. *Glot international*, 5:341–345, 01 2001.

[3] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] Gavin C Cawley and Nicola LC Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11(Jul):2079–2107, 2010.

[5] Giovanni Dimauro, Vincenzo Di Nicola, Vitoantonio Bevilacqua, Danilo Caivano, and Francesco Girardi. Assessment of speech intelligibility in parkinson's disease using a speech-to-text system. *IEEE Access*, pages 22199–22208, 2017.

[6] Giovanni Dimauro, Vincenzo Di Nicola, Vitoantonio Bevilacqua, Francesco Girardi, and Vito Napoletano. Voxtester, software for digital evaluation of speech changes in parkinson disease. *Proc. IEEE Int. Symp. Med. Meas. Appl. (MeMeA)*, pages 1–6, 2016.

[7] Theodoros Evgeniou and Massimiliano Pontil. Support vector machines: Theory and applications. volume 2049, pages 249–257, 01 2001.

[8] SRLE Fahn. Unified parkinson's disease rating scale. *Recent development in Parkinson's disease*, 1987.

[9] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[10] Salvador García, Sergio Ramírez-Gallego, Julián Luengo, José Manuel Benítez, and Francisco Herrera. Big data preprocessing: methods and prospects. *Big Data Analytics*, 1, 2016.

[11] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[12] JI Godino-Llorente, S Shattuck-Hufnagel, JY Choi, L Moro-Velázquez, and JA Gómez-García. Towards the identification of idiopathic parkinson's disease from the speech. new articulatory kinetic biomarkers. *PloS one*, 12(12):e0189583, 2017.

[13] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. Cnn architectures for large-scale audio classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 131–135. IEEE, 2017.

[14] Aileen K Ho, Robert Iansek, Caterina Marigliani, John L Bradshaw, and Sandra Gates. Speech impairment in a large sample of patients with parkinson's disease. *Behavioural neurology*, 11(3):131–137, 1999.

[15] Margaret M Hoehn and Melvin D Yahr. Parkinsonism: onset, progression, and mortality. *Neurology*, 17(5):427–427, 1967.

[16] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

[17] Rhonda J. Holmes, Jennifer M. Oates, Debbie J. Phyland, and Andrew J. Hughes. Voice characteristics in the progression of parkinson's disease. *International Journal of Language & Communication Disorders*, 35(3):407–418, 2000.

[18] Yannick Jadoul, Bill Thompson, and Bart de Boer. Introducing Parselmouth: A Python interface to Praat. *Journal of Phonetics*, 71:1–15, 2018.

[19] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Internet; descargado 26-junio-2019].

[20] M. A. Little, P. E. McSharry, E. J. Hunter, J. Spielman, and L. O. Ramig. Suitability of dysphonia measurements for telemonitoring of parkinson's disease. *IEEE Transactions on Biomedical Engineering*, 56:1015–1022, Enero 2009.

[21] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[22] Janine Möbes, Gregor Joppich, Frank Stiebritz, Reinhard Dengler, and Christine Schröder. Emotional speech in parkinson's disease. *Movement Disorders*, 23(6):824–829, 2008.

[23] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[24] J. R. Orozco-Arroyave, J. D. Arias-Londoño, J. F. Vargas-Bonilla, M. González-Rátiva, and E. Nöth. New spanish speech corpus database for the analysis of people suffering from parkinson's disease. *Proceedings of the 9th Language Resources and Evaluation Conference (LREC)*, pages 342–347, 2014.

[25] J. R. Orozco-Arroyave, F. Hönig, J. D. Arias-Londoño, J. F. Vargas-Bonilla, K. Daqrouq, S. Skodda, J. Rusz, and E. Nöth. Automatic detection of parkinson's disease in running speech spoken in three different languages. *The Journal of the Acoustical Society of America*, 139:481–500, Enero 2016.

[26] Juan Rafael Orozco-Arroyave, Juan Camilo Vásquez-Correa, Jesús Francisco Vargas-Bonilla, Raman Arora, Najim Dehak, Phani S Nidadavolu, Heidi Christensen, Frank Rudzicz, Maria Yancheva, H Chinaei, et al. Neurospeech: An open-source software for parkinson's speech analysis. *Digital Signal Processing*, 77:207–221, 2018.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[28] Kathe S Perez, Lorraine Olson Ramig, Marshall E Smith, and Christopher Dromey. The parkinson larynx: tremor and videostroboscopic findings. *Journal of Voice*, 10(4):354–361, 1996.

[29] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. *Encyclopedia of Database Systems*, 532–538:532–538, 01 2009.

[30] J Rusz, R Cmejla, H Ruzickova, and E Ruzicka. Quantitative acoustic measurements for characterization of speech and voice disorders in early untreated parkinson's disease. *The journal of the Acoustical Society of America*, 129(1):350–367, 2011.

[31] Sabine Skodda, Wenke Visser, and Uwe Schlegel. Vowel articulation in parkinson's disease. *Journal of voice*, 25(4):467–472, 2011.

[32] Glenn T Stebbins and Christopher G Goetz. Factor structure of the unified parkinson's disease rating scale: motor examination section. *Movement disorders: official journal of the Movement Disorder Society*, 13(4):633–636, 1998.

[33] A. Tsanas, M. A. Little, P. E. Mcsharry, and L. O. Ramig. Accurate telemonitoring of parkinsons disease progression by noninvasive speech tests. *IEEE Transactions on Biomedical Engineering*, 57:884–893, Abril 2010.

[34] A. Tsanas, M. A. Little, P. E. Mcsharry, J. Spielman, and L. O. Ramig. Novel speech signal processing algorithms for high-accuracy classification of parkinson's disease. *IEEE Transactions on Biomedical Engineering*, 59:1264–1271, Mayo 2012.

[35] Wikipedia. Latex — wikipedia, la enciclopedia libre, 2015. [Internet; descargado 30-septiembre-2015].

[36] Wikipedia. Jitter — wikipedia, la enciclopedia libre, 2019. [Internet; descargado 16-abril-2019].

[37] Wikipedia. Keras — wikipedia, la enciclopedia libre, 2019. [Internet; descargado 26-junio-2019].

[38] Wikipedia. Mfcc — wikipedia, la enciclopedia libre, 2019. [Internet; descargado 16-abril-2019].

[39] I. H. Witten, E. Frank, M. A. Hall, , and C. J Pal. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 4 edition, 2017.

[40] Eberhard Zwicker and Ernst Terhardt. Analytical expressions for critical-band rate and critical bandwidth as a function of frequency.

The Journal of the Acoustical Society of America, 68(5):1523–1525, 1980.