



**Universidad
Europea**

UNIVERSIDAD EUROPEA DE MADRID

**ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO
GRADO EN INGENIERÍA INFORMÁTICA**

PROYECTO FIN DE GRADO

**Analítica de Datos IoT para predicción del
Síndrome de Burnout.**

DIEGO ALBERTO ABDUL-MASSIH LÓPEZ

Dirigido por

Dr. DIEGO GACHET PÁEZ

CURSO 2019-2020

TÍTULO: Analítica de Datos IoT para predicción del Síndrome de Burnout.

AUTOR: DIEGO ALBERTO ABDUL-MASSIH LÓPEZ

TITULACIÓN: GRADO EN INGENIERÍA INFORMÁTICA

DIRECTOR DEL PROYECTO: Dr. DIEGO GACHET PÁEZ

FECHA: JUNIO de 2020

RESUMEN

Este trabajo de fin de grado se focaliza en encontrar la relación entre encuestas del síndrome de Burnout y los datos fisiológicos recogidos por pulseras Fitbit, logrando un sistema que permita visualizar y predecir datos relacionados con este síndrome.

En primer lugar, se realizará un proceso ETL (Extracción, Transformación y Carga) de los datos, utilizando Google Dataprep. Luego de contar con los datos limpios, se deberán extraer para su análisis; debido al gran volumen de los datos, se utilizará SPARK como herramienta principal para el aprendizaje automático de estos.

Por último, se finalizará construyendo una herramienta de visualización de datos, a través de la librería para Python Dash, en la que se incluirá un apartado donde se podrá visualizar el Sistema de Detección del síndrome de Burnout.

Palabras clave: Google Dataprep, SPARK, ETL, Dash, Aprendizaje Automático.

ABSTRACT

This end-of-degree work focuses on finding the relationship between surveys of Burnout syndrome and the physiological data collected by Fitbit bracelets. Achieving a system that allows visualizing and predicting data related to this syndrome.

First, an ETL (Extract, Transform and Load) process of the data will be carried out, using Google Dataprep, once we have the data clean; we extract it for analysis using SPARK as the main tool for machine learning due to the great volume.

We finished building a data visualization tool using the Python Dash library, including a section where you will find the Burnout Syndrome Detection System.

Key words: Google Dataprep, SPARK, ETL, Dash, Machine Learning.

AGRADECIMIENTOS

Después de unos meses intensos de trabajo fuerte para sacar adelante esta investigación, escribo por último los agradecimientos para dar finalizada una etapa que comenzó en Venezuela y por motivos varios termina en Madrid, España.

Ha sido una experiencia única, agradezco al Doctor Diego Gachet Páez por su apoyo, ha sido una persona clave para el desarrollo de este proyecto, consiguió la inversión de Google para hacer viable este proyecto cuestión clave.

Quiero agradecer a mi novia, a mis padres y mi hermana por siempre estar allí con toda la comprensión posible y un apoyo incondicional.

TABLA RESUMEN

DATOS	
Nombre y apellidos:	Diego Alberto Abdul-Massih López
Título del proyecto:	Analítica de Datos IoT para predicción del Síndrome de Burnout.
Directores del proyecto:	Dr. Diego Gachet Páez
El proyecto se ha realizado en colaboración de una empresa o a petición de una empresa:	SI
El proyecto ha implementado un producto: (esta entrada se puede marcar junto a la siguiente)	SI
El proyecto ha consistido en el desarrollo de una investigación o innovación: (esta entrada se puede marcar junto a la anterior)	SI
Objetivo general del proyecto:	Estudiar la relación entre el síndrome de burnout y alteraciones en los baremos de la salud, para la realización de un modelo predictivo con la máxima exactitud posible.

Índice

RESUMEN.....	3
ABSTRACT	3
TABLA RESUMEN	5
Capítulo 1. Introducción	11
1.1 Planteamiento del problema	11
1.1.1 Dimensiones del síndrome de burnout	11
1.2 Objetivos del proyecto	12
1.2.1 Objetivo Global	12
1.2.2 Objetivos Concretos	12
1.3 Áreas de conocimiento	12
1.4 Metodología utilizada.....	12
1.5 Recursos utilizados.....	14
1.5.1 Google Cloud Platform	14
1.5.2 Lenguajes de programación.....	15
1.5.3 Control de Versiones	15
Capítulo 2. Estado del arte	16
2.1 Estado del Arte	16
2.1.1 La Autoestima y el Burnout	16
2.1.2 Barómetro del Burnout.....	17
2.1.3 Investigación de Agotamiento en una Muestra de Médicos Generales Británicos	
17	
2.2 Contexto y justificación	18
2.3 Planteamiento del problema	18
Capítulo 3. Diseño del Sistema.....	20
3.1 Descripción del funcionamiento del sistema.....	20
3.2 Requisitos	21
3.2.1 Requisitos funcionales.....	21
3.2.2 Requisitos de calidad.....	21
a. Usabilidad	21
3.2.3 Prioridades de los requisitos.....	22

3.3	Casos de uso	23
3.3.1	Descripción de los actores	24
3.3.2	Descripción de los casos de uso	24
3.4	Diseño del sistema.....	24
3.4.1	Entorno del sistema y patrón de diseño.....	25
Capítulo 4.	Desarrollo e Implantación del Sistema.....	26
4.1	Introducción a Google Cloud Platform	26
4.1.1	Características de Google Cloud Platform	26
4.2	Servicios	27
4.3	Precio de Google Cloud Platform	28
4.4	Uso de GCP en el desarrollo del sistema	28
4.4.1	Instancia de Compute Engine.....	29
4.4.2	Descripción de los datos.....	30
4.4.3	Dataprep – Dataflow	36
4.4.4	Cloud Data Storage	41
4.4.5	Clúster Dataproc.....	41
Capítulo 5.	Algoritmo de Aprendizaje Automático Aplicado al Síndrome de Burnout	46
5.1	Conceptos básicos	46
5.2	Análisis de los datos.....	46
5.2.1	Burnout Dataset.....	47
5.2.2	Análisis de los principales componentes.....	48
5.3	Algoritmos Machine Learning aplicados	49
5.3.1	Preparación de los datos	50
5.3.2	CrossValidation	53
5.3.3	Algoritmo Regresión Logística	53
5.3.4	Algoritmo Árbol de Decisión	56
5.3.5	Algoritmo Bosques Aleatorios	59
5.3.6	Conclusión.....	61
Capítulo 6.	Desarrollo de una Herramienta Propia de Visualización	62
6.1.1	Importancia de la Herramienta	62
6.1.2	Componentes utilizados	62
6.1.3	Datos alojados en Big Query	63
6.1.4	Código de la herramienta de visualización.....	63
Capítulo 7.	Conclusiones	74
Capítulo 8.	Futuras Líneas de Trabajo	75

PRESUPUESTO	76
BIBLIOGRAFÍA.....	77

Índice de Figuras

Ilustración 1: Ciclo de desarrollo ágil SCRUM	13
Ilustración 2:Diagrama Gantt de las fases del proyecto.	14
Ilustración 3:Bono de créditos de Google Cloud Platform	15
Ilustración 4: Barómetro.....	17
Ilustración 5:Casos de uso.....	23
Ilustración 6:Entorno del sistema.....	25
Ilustración 7:Principales empresas en Google Cloud.....	27
Ilustración 8:Empresas en Google Cloud.....	27
Ilustración 9:Servicios de Google Cloud Platform.....	27
Ilustración 10: Entorno del Sistema	28
Ilustración 11:Instancia en Google Cloud Platform	29
Ilustración 12: Usuarios CSV	30
Ilustración 13: Detalle Usuarios csv.....	31
Ilustración 14: CSV - Estados de Ánimo	31
Ilustración 15: Detalle Estados de Ánimo csv.....	32
Ilustración 16: CSV – Encuestas	32
Ilustración 17: Detalle Encuestas csv.....	32
Ilustración 18: CSV Raw Ahí.....	33
Ilustración 19: Detalle Raw - Ahi csv	33
Ilustración 20: CSV - Raw-Sleep	34
Ilustración 21: Detalle Raw-Sleep csv	35
Ilustración 22: Flujo de Trabajo Dataprep en Google Cloud Platform	36
Ilustración 23:Datos Despierto Dataprep	37
Ilustración 24:Datos Dormido Dataprep	37
Ilustración 25:Datos de usuarios	38
Ilustración 26:Datos de usuarios Burnout	38
Ilustración 27:Datos Fisiológicos Dataprep	39
Ilustración 28:Datos de estados de ánimo	39
Ilustración 29:Datos unión	40
Ilustración 30:Datos unión 2	40
Ilustración 31:Exportación de los datos	41
Ilustración 32:Dataproc en Google Cloud Platform.....	42
Ilustración 33:Dataproc en Google Cloud Platform.....	42
Ilustración 34:Conexión SSH Dataproc	43
Ilustración 35:Dataproc Spark-Submit.....	43
Ilustración 36: Dataproc Regresión Logística.....	44
Ilustración 37: Dataproc Bosques Aleatorios	44
Ilustración 38: Dataproc Árbol de Decisión.....	44
Ilustración 39:Guardado de modelos.....	45
Ilustración 40: Esquema principal del Dataset	47
Ilustración 41:Principales Componentes	48
Ilustración 42:Variables para el estudio	49

Ilustración 43:Código de selección de variables según su importancia	50
Ilustración 44:Codificación de variables categóricas	50
Ilustración 45:Codificación de variables numéricas	51
Ilustración 46:Pipeline	51
Ilustración 47:Código de RandomSplit	52
Ilustración 48:Código de preparación de la data.....	52
Ilustración 49:Ecuación Regresión Logística.....	53
Ilustración 50:Ecuación de Regresión Logística predicción	54
Ilustración 51: Regresión Logística (https://www.javatpoint.com/logistic-regression-in-machine-learning)	54
Ilustración 52: Código Regresión Logística.....	54
Ilustración 53: Predicciones Regresión Logística	56
Ilustración 54: Árbol de decisión	57
Ilustración 55: Código del árbol de decisión.....	57
Ilustración 56: Predicciones Árbol de Decisión	58
Ilustración 57: Código Bosques Aleatorios.....	59
Ilustración 58: Predicciones Bosques Aleatorios	60
Ilustración 59: Datos alojados en Big Query.....	63
Ilustración 60: Querys ejecutadas para las gráficas.....	63
Ilustración 61: Código de la herramienta de visualización	64
Ilustración 62: Código del panel de navegación	64
Ilustración 63: Código de tablas dinámicas.....	65
Ilustración 64: Página principal del Sistema Burnout Study.....	65
Ilustración 65: Formulario del sistema de detección.....	66
Ilustración 66: Predicción del Sistema Burnout Study.....	67
Ilustración 67: Importancia de las características.....	67
Ilustración 68: Datos fisiológicos por paciente	68
Ilustración 69: Burnout por Subescalas.....	69
Ilustración 70: Burnout por Subescalas Individuales	69
Ilustración 71: Burnout por Especialidad	70
Ilustración 72: Burnout por Área de trabajo.....	70
Ilustración 73: Burnout por Sexo	71
Ilustración 74: Burnout por Estado Civil	71
Ilustración 75: Burnout por Tipo de Contrato	72
Ilustración 76: Burnout por Número de Hijos.....	72
Ilustración 77: Burnout por Edad	73
Ilustración 78: Presupuesto	76

Capítulo 1. Introducción

En este proyecto se pretende analizar los datos fisiológicos de varios funcionarios de dos hospitales de Madrid, que participaron en un estudio sobre síndrome de Burnout. Lo que se pretende es analizar dichos datos, obtener la relación existente entre ellos y luego realizar un modelo predictivo a partir de los resultados.

1.1 Planteamiento del problema

El término burnout fue determinado inicialmente por Freudenberger (1974) para referirse al desarrollo de decadencia en los cuidados y aplicación profesional a los usuarios de las organizaciones de servicios (organizaciones de voluntariado, sanitarias, servicios sociales, educativos, etc.).

Sin embargo, existen otras definiciones del término burnout, como es la proposición por Maslach y Jackson (1981/1986) y la elaborada por (Gil-Monte y Peiró, 1999). Estos autores definen el burnout como la manifestación inadecuada al estrés emocional crónico, que resulta de una discrepancia entre los ideales individuales y la realidad de la vida ocupacional diaria, requiriéndose al menos seis meses de periodo de adaptativo.

1.1.1 Dimensiones del síndrome de burnout

El síndrome de burnout involucra tres medidas. Estas son:

1. **Agotamiento emocional** (pérdida o desgaste de recursos emocionales: anergia, agotamiento, fatiga)
2. **Deshumanización o despersonalización** (actitudes negativas, cínicas e insensibles hacia los pacientes, familias, compañeros)
3. **Falta de realización personal en el trabajo** (tendencia a evaluar el propio trabajo de forma negativa: sentimientos de inadecuación o fracaso).

Este proyecto, será de utilidad para comprender cuáles son los efectos del síndrome de burnout en la salud, a través del análisis de datos reales, recopilados al amparo de un proyecto de investigación interno de la UEM en dos hospitales de España. En este sentido, la población de estudio está compuesta por los médicos de dos unidades: Urgencias y Psiquiatría.

A su vez, existen también otro conjunto de datos, que hace referencia a los datos personales de cada sujeto, donde se diferencian tres categorías: (Datos personales, Datos laborales, hábitos sociales y de tiempo de ocio).

En este orden de ideas, es importante mencionar que a través de la utilización de la Inteligencia Artificial y la herramienta SPARK, la cual funciona de manera distribuida gracias a Google Cloud Platform, se logrará realizar una predicción del síndrome de burnout, con un

modelo propio y se creará una herramienta de visualización de datos para posteriores análisis de expertos.

1.2 Objetivos del proyecto

1.2.1 Objetivo Global

El objetivo principal del proyecto es estudiar la relación entre el síndrome de burnout y alteraciones en los baremos de la salud, para la realización de un modelo predictivo con la máxima exactitud posible.

1.2.2 Objetivos Concretos

Se parte de un conjunto de datos que se han obtenido mediante un proyecto de investigación de la Universidad Europea de Madrid, y se van a tratar para conseguir:

1. Limpieza y carga de datos en una plataforma de computación distribuida como Google Cloud, utilizando el servicio Google Dataprep.
2. Categorización de los sujetos según las encuestas en dos grandes grupos “Verdadero” y “Falso”, verdadero para positivo en burnout y falso para negativo en burnout.
3. Construir un sistema que sea capaz de visualizar estos datos observando las diferencias entre todas las variables, tanto del conjunto de datos personales, hábitos y datos fisiológicos, entre los sujetos que den “Verdadero” y los que den “Falso”, para conseguir una relación.
4. Realizar un análisis de importancia de características.
5. Realizar el sistema de predicciones categórico con una interfaz web de acceso, estudiando qué tipo de algoritmo arroja mejores resultados.

1.3 Áreas de conocimiento

Las áreas de conocimiento que intervienen en este proyecto son las siguientes:

- Ciencias de la computación.
- Estadística y ciencia de datos.
- Psiquiatría y urgencias.

1.4 Metodología utilizada

La metodología utilizada para el desarrollo de este proyecto de fin de grado se caracteriza por ser ágil, teniendo ciclos conocidos como “Sprint”, en el que se destacan las siguientes fases:

1. *Análisis de requisitos:* en esta fase se realizó una reunión para la licitación primaria de requisitos, donde se destacaron las principales características que debía tener el sistema final, así como los principales focos a resolver.
2. *Diseño:* con los requisitos adquiridos en la reunión, se inició el diseño de la solución, el cual se caracterizó por ser dinámico y con validación continua hacia el director del proyecto.
3. *Desarrollo:* con el diseño se empezó el desarrollo de la solución. Esta fase se caracterizó por ser repetitiva, ya que, al alcanzar un hito, se volvió a rectificar mediante reuniones con los diferentes actores los objetivos alcanzados para continuar con los siguientes.
4. *Control de calidad:* una vez superado el proceso de desarrollo con todos los objetivos alcanzados, se hizo una revisión exhaustiva para comprobar que el sistema cumplía con los estándares de calidad requeridos.
5. *Despliegue:* Al contar con los estándares de calidad requeridos, se procedió a desplegar una versión del sistema en la web.

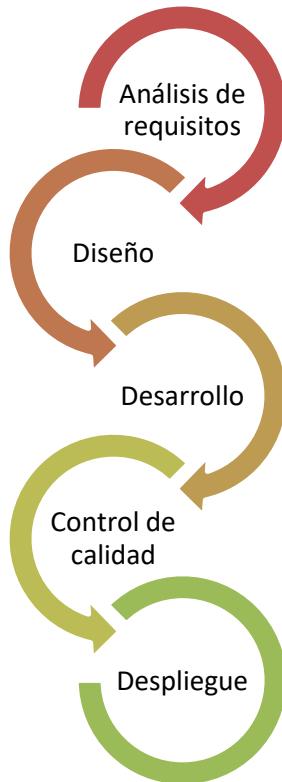


Ilustración 1: Ciclo de desarrollo ágil SCRUM.

La planificación en forma de diagrama Gantt fue la siguiente:

ACTIVIDADES	ENERO				FEBRERO				MARZO				ABRIL				MAYO	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Limpiar y cargar los diferentes conjuntos de datos a Google Cloud, para su posterior analítica.																		
Categorizar a cada sujeto dependiendo de las reglas para diagnosticar el síndrome de burnout y unir estos datos a los anteriores.																		
Analizar los datos observando las diferencias entre todas las variables, tanto del conjunto de datos personales, hábitos y Fitbit, entre los sujetos que den “VERDADERO” y los que den “FALSO”, para conseguir una relación																		
Obtención de las características más importantes del conjunto de datos.																		
Realización del Sistema de Detección probando con diferentes algoritmos, utilizando SPARK ML																		
Realización de un sistema de visualización de datos con la herramienta Dash incorporado con el Sistema de Detección.																		
Conclusiones																		
Documentar																		

Fuente :

Datos obtenidos por los autores.

Leyenda:

Tiempo estimado



Tiempo estimado en semanas: 17 semanas

Tiempo estimado en horas: se trabaja aprox. 25 h x Semana, para un total de = 425 horas

Ilustración 2:Diagrama Gantt de las fases del proyecto.

1.5 Recursos utilizados

El desarrollo del sistema para detectar el síndrome de burnout, como la visualización de los datos de todo el proceso ETL, son tareas que requieren el aprendizaje de nuevas tecnologías y la investigación activa, razón por la cual es importante describir los recursos utilizados.

1.5.1 Google Cloud Platform

En Google Cloud Platform se ha utilizado **Google Dataprep** para la realización del proceso ETL, el **Big Query** como base de datos principal para la gestión de los datos y el **Google Dataproc** para ejecutar el código generado en SPARK de manera distribuida. Además, se ha empleado **una instancia** para el despliegue de la solución web y **Google Storage** para guardar los modelos.

En este sentido, resultó conveniente utilizar Google Cloud Platform, debido al volumen importante de los datos, ya que estos pasan de los 3 millones de líneas en el fichero csv.

Es importante mencionar que Google es un gran promotor de los objetivos que se pretenden conseguir con el proyecto, ya que otorgó un bono de créditos en su plataforma.

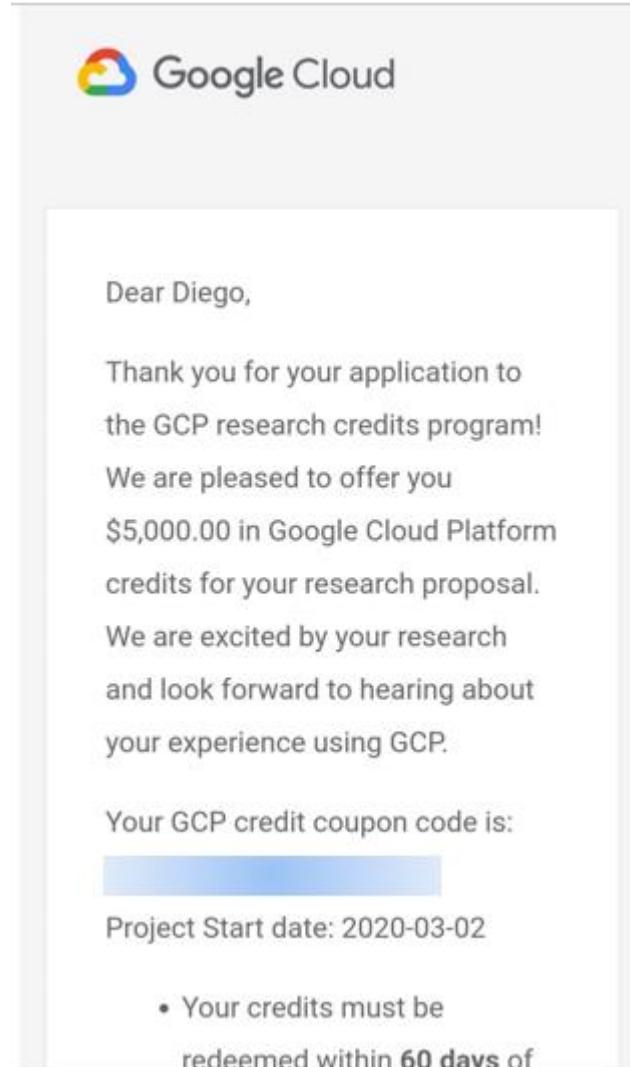


Ilustración 3:Bono de créditos de Google Cloud Platform

1.5.2 Lenguajes de programación

Los lenguajes de programación utilizados en el proyecto son los siguientes:

- HTML5 y CCS3 en la librería Dash de Python para el Sistema Web.
- Python – Spark para la creación de modelos de aprendizaje.
- SQL para la interacción con Big Query.
- Interacción con la Shell de Google Cloud Platform y Ubuntu distribuido en Dataproc.

1.5.3 Control de Versiones

Con respecto al control de versiones, se utilizó un repositorio de GitHub.

Capítulo 2. Estado del arte

El siguiente proyecto se elabora en colaboración con los hospitales Infanta Sofía y Son Llàtzer, tomando como punto de inicio un proyecto de la Universidad Europea de Madrid, el cual se centró en la recolección de los datos fisiológicos de los diferentes médicos participantes de ambos hospitales.

2.1 Estado del Arte

Se pretende estudiar la relación entre los datos fisiológicos recogidos y el síndrome de estar quemado (burnout), mediante el desarrollo de un sistema de inteligencia artificial que es capaz de detectar este síndrome.

Anteriores investigaciones sobre burnout que se han encontrado destacan las siguientes:

2.1.1 La Autoestima y el Burnout

En este trabajo de investigación titulado “La autoestima como variable protectora del "burnout" en estudiantes de fisioterapia” realizado por (González Cabanach, Ramón; Souto-Gestal, Antonio; Freire Rodríguez, Carlos; Fernández-Cervantes, Ramón; y González Doniz, Luz).

Se busca la relación entre el síndrome de burnout y el nivel de autoestima de 487 estudiantes, utilizando la encuesta Maslach Burnout Inventory (MBI). En él se concluye que;

“Cuando los estudiantes tienen una baja autoestima, aparecen manifestaciones más claras de burnout. En cambio, cuanto mayor es la valoración que hacen de sí mismos, menores síntomas de agotamiento y despersonalización y mayores niveles de realización personal manifiestan los estudiantes.”.

Encontraremos el trabajo de investigación al que hacemos referencia en el siguiente enlace.

González Cabanach, Ramón; Souto-Gestal, Antonio; Freire Rodríguez, Carlos; Fernández-Cervantes, Ramón; y González Doniz, Luz. (2016). La autoestima como variable protectora del "burnout" en estudiantes de fisioterapia. 2020, de Universidade da Coruña Sitio web: <https://ruc.udc.es/dspace/handle/2183/16897>

2.1.2 Barómetro del Burnout

En el proyecto “Burnout Barometer” realizado por Aaron Sachs, es una herramienta para registrar burnout, asignándole ciertos niveles.

Para su funcionamiento debemos escribir alguna frase o sentimiento, como una especie de diario, y añadirle un número para que el sistema guarde como nos sentíamos en ese momento.

Nivel	Descripción
5	Las cosas van bien y realmente puedes decir que eres feliz. Acabo de cenar con tus amigos más cercanos, en un viaje que 5 habías planeado hace meses, una palmada en el trabajo, etc.
4	Estás deseando algo: volver a casa los fines de semana, un viernes por la noche con amigos, etc. Las cosas se están 4 poniendo optimistas y te sientes muy bien.
3	No es ni bueno ni malo. Recomendamos realizar controles puntuales frecuentes durante todo el día para practicar un 3 cierto nivel de atención plena.
2	Algo sucedió que te molestó. Estás nervioso por lo que sucederá más tarde durante el día, un comentario molesto que no 2 te gustó, etc.
1	Han sucedido cosas malas y necesitas una avenida para salir. Si ha sentido muchos 1s durante los días, le recomendamos 1 que se comunique con alguien.

Ilustración 4: Barómetro

Está orientado para ser usado con la aplicación Slack, de manera que constantemente se encuentra guardando todo lo que escribimos, con los sentimientos que le asignamos; el autor detalla que sus usuarios principales serían equipos de IT, que se comuniquen a través de esta aplicación “Slack”.

Encontraremos el proyecto al que hacemos referencia en el siguiente enlace.

Aaron Sachs. (2018). Burnout Barometer. 2020, de GitHub Sitio web:
<https://ljvmiranda921.github.io/burnout-barometer/>

2.1.3 Investigación de Agotamiento en una Muestra de Médicos Generales Británicos

En esta investigación realizada por (Michael Kirwan y David Armstrong), se destaca, “Los cambios recientes en el contrato de medicina general, han producido un aumento en la carga de trabajo y estrés, peor salud mental y reducción de la satisfacción. Estos factores pueden combinarse, para aumentar el nivel de burnout entre los profesionales generales.”

El estudio se centraba en enviar encuestas a 295 profesionales de la salud, de la provincia de Northamptonshire en Reino Unido.

Los resultados destacan que no hubo asociación entre la edad y el nivel de burnout, aunque se encontró una pequeña correlación negativa entre la edad y la personalización de otras subescalas.

Igualmente destacan que los practicantes generales a tiempo parcial muestran niveles más bajos de estar quemado que los practicantes a tiempo completo. Destacan

también que no lograron conseguir una relación entre el sexo y el síndrome de estar quemado.

Michael Kirwan y David Armstrong. (1995). Investigación de Agotamiento en una Muestra de Médicos Generales Británicos . 2020, de British Journal of General Practice Sitio web:

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1239232/pdf/brjgenprac00020-0039.pdf>

2.2 Contexto y justificación

A pesar de existir varias investigaciones y proyectos relacionados con el burnout, no se ha encontrado un sistema que estudie la relación entre los datos fisiológicos y el síndrome de estar quemado, teniendo como objetivo detectar este síndrome utilizando la inteligencia artificial.

El síndrome de estar quemado afecta en gran medida al personal de salud, por lo que siguiendo con la línea de desarrollo de sistemas que ayuden a simplificar las pruebas y obtener mayores facilidades en el ámbito médico, este sistema también ayudará a la detección sencilla y rápida del síndrome. Igualmente, pretende proveernos de una herramienta de visualización, que permitirá a los expertos sanitarios estudiar con mayor facilidad y exactitud dicho síndrome.

2.3 Planteamiento del problema

El término burnout fue determinado inicialmente por Freudenberger (1974) para referirse al desarrollo de decadencia en los cuidados y aplicación profesional a los usuarios de las organizaciones de servicios (organizaciones de voluntariado, sanitarias, servicios sociales, educativos, etc.).

Sin embargo, existen otras definiciones del término burnout, como es la proposición por Maslach y Jackson (1981/1986) y la elaborada por (Gil-Monte y Peiró, 1999). Estos autores definen el burnout como la manifestación inadecuada al estrés emocional crónico, que resulta de una discrepancia entre los ideales individuales y la realidad de la vida ocupacional diaria, requiriéndose al menos seis meses de periodo de adaptativo.

El burnout aparece en individuos sin historia de trastornos psicológicos o psiquiátricos, se desarrolla gradualmente y no está presente cuando se inicia un nuevo empleo (Moss et al, 2016). El progreso del cuadro se relaciona con la exceso del trabajo, pero el exceso de tarea no provoca sin más el síndrome, es más significativo la desmotivación emocional y cognitiva por el abandono de intereses que habían sido importantes, por la discordancia entre el esfuerzo y lo conseguido.

En los últimos años se observa un aumento de las prácticas de medicina defensiva, los médicos han perdido su rol clásico de cuidador.

Se ha descrito la aparición del burnout cuando el clínico considera que el cuidado que está proporcionando no se adecúa a sus valores o su conocimiento personal, debido a la

diferencia entre la cantidad o calidad del cuidado y el pronóstico, los pacientes no cumplidores, la falta de información, no respetar los deseos de los pacientes o dar un cuidado que se percibe de poca calidad.

Capítulo 3. Diseño del Sistema

En este capítulo mostraremos el desarrollo del sistema, comenzando por su funcionamiento, continuando con las especificaciones de requisitos, casos de uso y terminando con el diseño final desplegado.

3.1 Descripción del funcionamiento del sistema

El sistema de visualización y detección de burnout es un sistema orientado al uso por personas con experiencia en la investigación de este síndrome. La idea es desarrollar un sistema, a través del cual los expertos puedan comparar, visualizar y evaluar las diferentes variables del síndrome, con la facilidad que un entorno web puede proveer. Los datos que se presentan en el sistema son:

1. Datos fisiológicos (parámetros de sueño, ritmo cardíaco, etcétera) capturados con pulseras Fitbit Alta HR por un proyecto anterior de la Universidad Europea.
2. Datos de las encuestas de la escala Maslach, realizadas por el personal médico en un proyecto anterior de la Universidad Europea.

De esta manera, el sistema web contiene las siguientes informaciones:

1. Un Dashboard que refleja gran cantidad de subcategorías de clasificación solicitadas por el investigador final, algunas de las cuales son categorizaciones por (sexo, especialidad, número de hijos, estado civil, etcétera), todas estas relacionadas con el síndrome.
2. Sistema de detección del síndrome de burnout mediante un cuestionario web.

Una vez hecha la descripción del funcionamiento del sistema, se estudiarán con más detalle los requisitos (Pressman,2005) , los cuales se clasifican en:

- Requisitos Funcionales: Definen las funciones del software o de sus componentes.
- Requisitos No Funcionales: Describen aquellas funciones que debe cumplir un determinado sistema.

3.2 Requisitos

Una vez descrita la funcionalidad del sistema, se deberán definir los requisitos, categorizándolos y asignándoles una prioridad.

3.2.1 Requisitos funcionales

En esta sección se detallarán las funcionalidades del sistema, con el objetivo de que los usuarios puedan utilizarlo correctamente.

a. Acceso al sistema

F-01. El usuario podrá acceder mediante la dirección web al sistema.

b. Sistema de detección

F-02. El usuario del sistema podrá llenar el formulario para conocer si tiene o no el síndrome.

F-03. El usuario debe recibir una recomendación según el resultado del sistema de detección.

c. Visualización de datos

F-04. El sistema le debe proporcionar la capacidad al usuario de visualizar el conjunto de datos por sus diversas categorías, pudiendo filtrar y seleccionar correctamente las gráficas.

d. Descarga de datos

F-05. El sistema debe estar capacitado para descargar los datos en bruto y realizar posteriores análisis (respetando la confidencialidad de estos).

F-06. El sistema debe permitir la descarga de los diferentes gráficos.

3.2.2 Requisitos de calidad.

En esta sección se especificarán los requisitos de calidad que debe cumplir el sistema

a. Usabilidad

US-01. El sistema debe facilitar su utilización permitiendo al usuario alcanzar cualquier funcionalidad en un máximo de 3 pasos, según las normas generales.

b. Mantenibilidad

MAN-01. El sistema debe cuidar las fases de diseño, desarrollo y control de calidad para fomentar su sencillo mantenimiento.

c. **Accesibilidad**

AC-01. La interfaz web del sistema debe cumplir con características adecuadas para ser accesible con facilidad.

d. **Capacidad**

CAP-01. El sistema debe soportar un número de conexiones simultáneas.

e. **Seguridad.**

SEG-01. El sistema no perderá los datos.

f. **Robustez.**

ROB-01. El sistema gestionará de forma correcta los datos introducidos por el usuario de forma inválida.

3.2.3 Prioridades de los requisitos

En este apartado mostraremos todos los requisitos con su nivel de importancia, donde 1 significa la máxima importancia y 5 la mínima.

Nombre del requisito	Prioridad
F-01. El usuario podrá acceder mediante la dirección web al sistema.	1
F-02. El usuario del sistema podrá llenar el formulario para conocer si tiene o no el síndrome.	1
F-03. El usuario debe recibir una recomendación según el resultado del sistema de detección.	2
F-04. El sistema le debe proporcionar la capacidad al usuario de visualizar el conjunto de datos por sus diversas categorías, pudiendo filtrar y seleccionar correctamente las gráficas.	1
F-05. El sistema debe estar capacitado para descargar los datos en bruto y realizar posteriores análisis (respetando la confidencialidad de estos).	3
F-06. El sistema debe permitir la descarga de los diferentes gráficos.	3
US-01. El sistema debe facilitar su utilización permitiendo al usuario alcanzar cualquier funcionalidad en un máximo de 3 pasos, según las normas generales.	2

MAN-01. El sistema debe cuidar las fases de diseño, desarrollo y control de calidad para fomentar el sencillo mantenimiento.	2
AC-01. La interfaz web del sistema debe cumplir con características adecuadas para ser accesible con facilidad.	2
CAP-01. El sistema debe soportar un número de iteraciones simultáneas.	2
SEG-01. El sistema no perderá los datos.	1
ROB-01. El sistema gestionará de forma correcta los datos introducidos por el usuario de forma inválida.	1

Tabla 1: Importancia de requisitos.

3.3 Casos de uso

Un modelo de casos de uso determina la funcionalidad que el sistema ofrece desde el punto de vista de los usuarios.

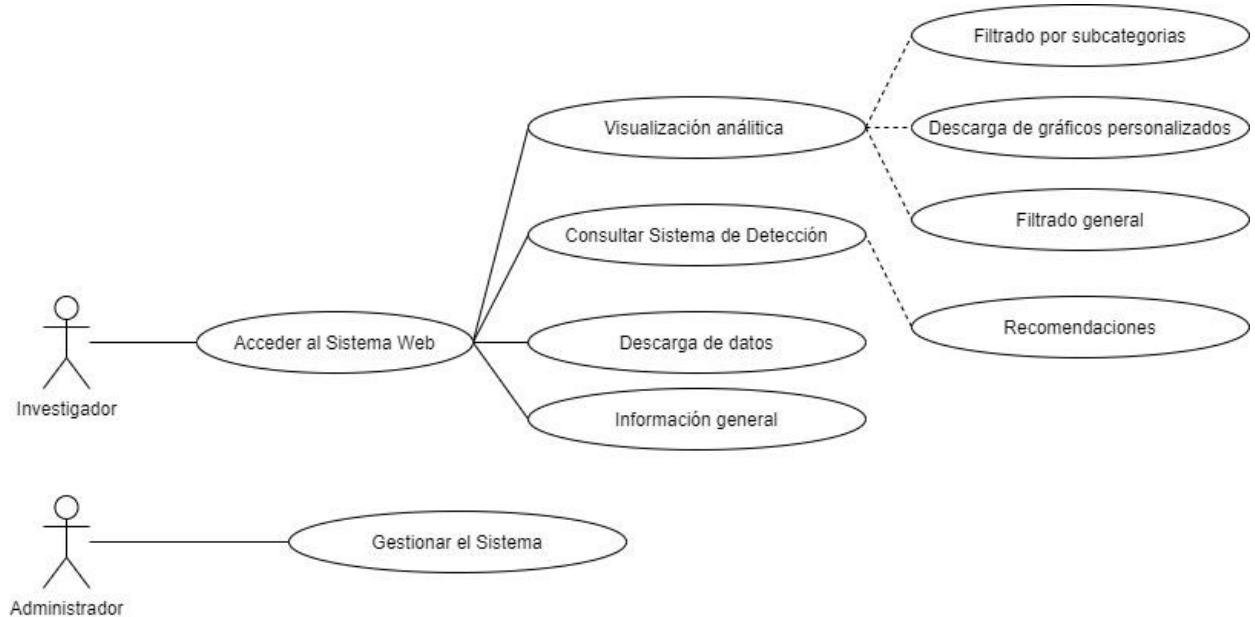


Ilustración 5: Casos de uso.

3.3.1 Descripción de los actores

Los actores que pueden utilizar el sistema son los siguientes:

- Investigador: representa a un usuario experto en el síndrome de burnout; no es un requisito indispensable ser investigador, sin embargo, el sistema se centra en este tipo de perfil.
- Administrador: representa al usuario encargado de mantener el correcto funcionamiento del sistema.

3.3.2 Descripción de los casos de uso

Se deberá realizar la relación de los casos de uso con su respectivo requisito:

- Acceder al Sistema Web: Esto es indispensable para poder utilizar los beneficios del sistema.

Requisito funcional: **F-01**.

- Visualización analítica: con esta herramienta el usuario podrá visualizar el conjunto de datos por sus diversas categorías, pudiendo filtrar y seleccionar correctamente las gráficas.

Requisito funcional: **F-04 y F-06**

- Consultar Sistema de Detección: Este caso de uso permitirá consultar si un usuario tiene burnout, arrojando diversas recomendaciones a partir de los resultados.

Requisito funcional: **F-02 y F-03**

- Descarga de datos: este caso de uso permite descargar y visualizar los datos utilizados para el aprendizaje del algoritmo.

Requisito funcional: **F-05**

3.4 Diseño del sistema

Luego de especificar los requisitos, casos de uso y diseño en general, resulta necesario explicar de forma visual, el funcionamiento del entorno del sistema y patrón de diseño.

3.4.1 Entorno del sistema y patrón de diseño

Mostraremos una visión general del entorno del sistema y su interacción.

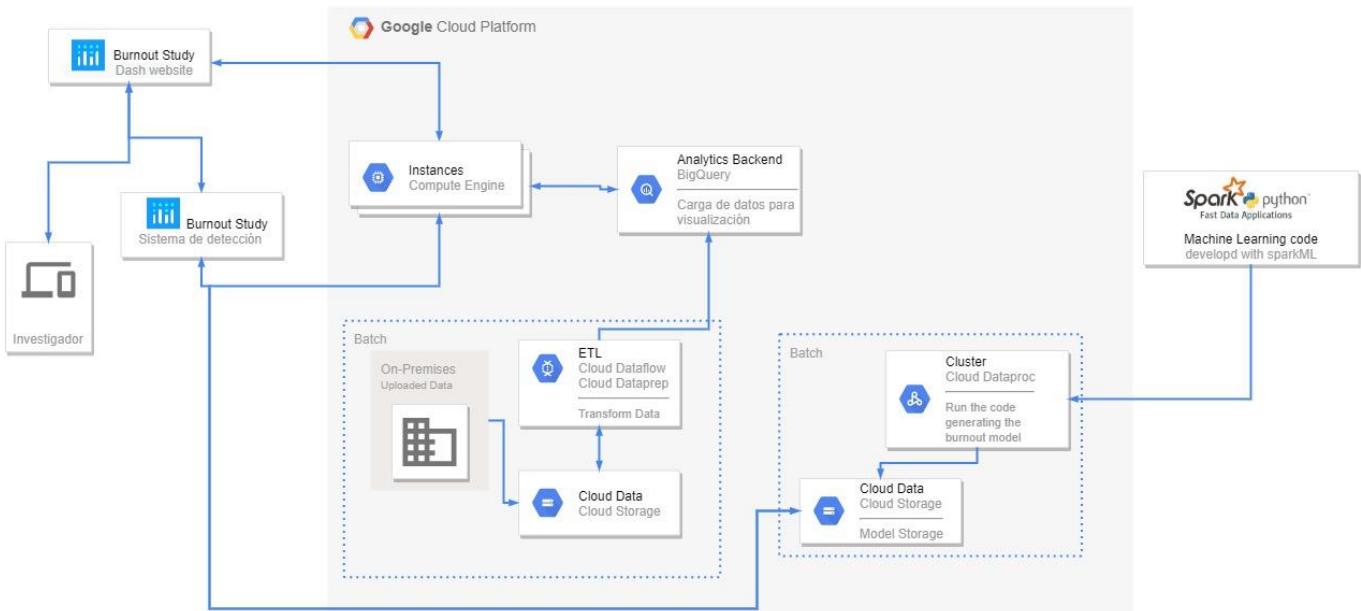


Ilustración 6:Entorno del sistema.

En cuanto al patrón de diseño, se ha utilizado MVC (Modelo Vista Controlador). Estas son algunas de sus características principales:

- El modelo contiene los datos con los que la aplicación trabaja, permitiendo la interacción y representación de estos.
- Las vistas constituyen todos los elementos visuales del sistema.
- El controlador es el encargado de gestionar las peticiones entre las vistas y el modelo.

Capítulo 4. Desarrollo e Implementación del Sistema

En este capítulo se explicarán las bondades de Google Cloud Platform y de qué manera se han aplicado en el desarrollo del sistema. Asimismo, resulta importante destacar que Google hizo una inversión a través de créditos, por un valor total de 5000\$ en sus plataformas, lo que representa la razón principal del uso de Google Cloud Platform y no de alguna otra plataforma.

4.1 Introducción a Google Cloud Platform

GCP es una infraestructura totalmente escalable, rentable y de grandes capacidades tecnológicas, donde Google proporciona a particulares y empresas servicios de computación, seguridad, bases de datos y herramientas Big Data.

4.1.1 Características de Google Cloud Platform

Google Cloud Platform es en su sector, una de las nubes más utilizadas por grandes empresas, debido a que sus características generan un valor agregado.

Algunas características generales de GCP son:

- Dispone de una infraestructura desplegada a nivel mundial, por lo que hay fácil elección de región para situar nuestros datos o servicios.
- Dispone de una inmensa cantidad de poder tecnológico, permitiendo efectuar tareas de alto rendimiento.
- Ofrecen altos estándares de seguridad.
- La curva de aprendizaje para hacerse con GCP es más sencilla que con otros proveedores.

Es interesante observar los datos de GCP sobre quienes están utilizando sus servicios, se puede visualizar que el mundo ha cambiado en dirección a la nube y que el hacer outsourcing de estas infraestructuras, muchas veces representa un costo importante para las empresas. Es por ello por lo que se puede afirmar que hoy en día es indispensable tener conocimientos entorno a este mundo en auge.



Ilustración 7:Principales empresas en Google Cloud

Ilustración 8:Empresas en Google Cloud

4.2 Servicios

Entre los principales servicios que ofrece Google Cloud Platform, se destacan: el alojamiento de algún servidor, almacenamiento con Google Storage, bases de datos relacionales, no relacionales y las complejas herramientas de Streaming, Machine Learning, etc.

En la siguiente figura se puede visualizar los diversos servicios que ofrece Google Cloud Platform según su área de aplicación.



Ilustración 9:Servicios de Google Cloud Platform

4.3 Precio de Google Cloud Platform

El precio de Google Cloud Platform suele ser pago por uso, es decir que se les cobra a los clientes lo que están utilizando actualmente, ofreciendo así la capacidad de hacer crecer y decrecer la infraestructura, según la carga de trabajo que se tenga.

Google Cloud Platform ofrece una calculadora en la que se pueden simular los servicios utilizados por hora, calculando cuál sería su coste.

(<https://cloud.google.com/products/calculator>).

Igualmente, para empresas, GCP ofrece un contacto directo con su departamento de ventas, en el cual se puede negociar el uso de su infraestructura por un coste previamente establecido.

4.4 Uso de GCP en el desarrollo del sistema

Como hemos mencionado anteriormente en el esquema del entorno, los servicios que utilizamos son los siguientes:

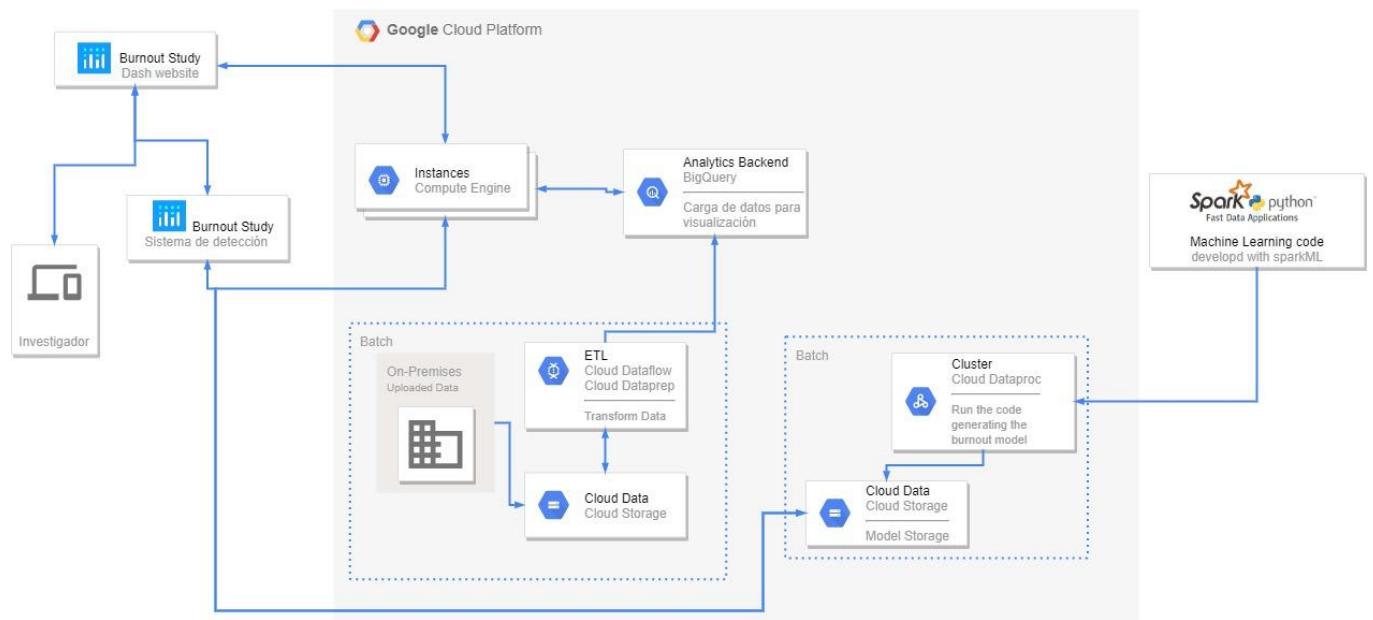


Ilustración 10: Entorno del Sistema

Cada uno de los bloques detallados en la ilustración, están enumerados con una referencia a su apartado explicativo.

- 4.4.1 Instancia de Compute Engine
- 4.4.3 Cloud Dataprep – Dataflow
- 4.4.4 Cloud Storage
- 4.4.5 Cloud Dataproc

4.4.1 Instancia de Compute Engine

Esta instancia se ha utilizado para desplegar la solución; en ella se ha establecido un entorno que se compone de los siguientes requerimientos:

- a. Python 3.7 (Anaconda)
- b. Librería Python dash 1.7.0
- c. Google-Auth 1.63
- d. Pyspark 2.4.5
- e. Spark 2.4.5
- f. Java 1.8.0_231
- g. Windows Server 2018.

Dicha instancia está funcionando en la zona “europe-west4-c” (Puerto de Ems, Países Bajos) debido a su cercanía con España, de manera que disminuimos el lag que se pueda presentar, si la colocamos en Estados Unidos, por ejemplo.

Actualmente está funcionando con 1 Core de CPU y 3.75 GB de Ram, es una configuración de ahorro, se observa que Google avisa que deberíamos asignarle más recursos, una vez puesta al uso masivo esto se puede cambiar fácilmente.

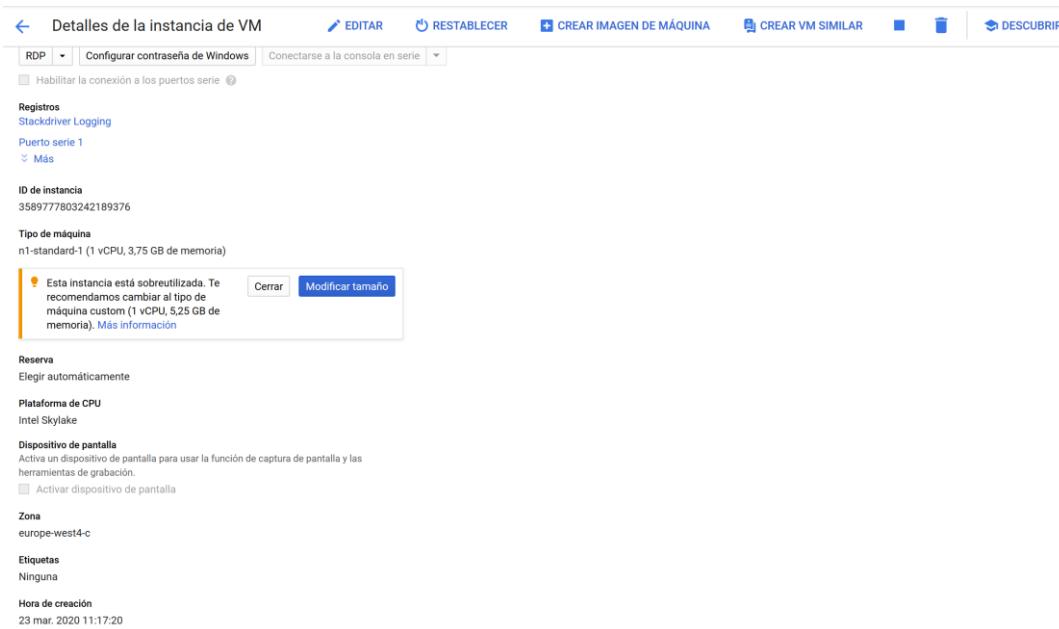


Ilustración 11:Instancia en Google Cloud Platform

4.4.2 Descripción de los datos

En este apartado se pretende dar una visión global de las características de cada fichero utilizado para el desarrollo con Dataprep y sus consecutivas aplicaciones en algoritmos de inteligencia artificial.

4.4.2.1 CSV – Usuarios

Guarda la información relativa a los usuarios que participaron en las encuestas y la recolección de datos por otro proyecto de la Universidad Europea de Madrid.



RBC	Name	# Value.altura
11fCM64PIJ0y0qX230jMMacFe0j2	132	2
1pNdNk2sUGgX8rvEp3xPBmaov5z1	164	3
2Ui5pXEP7eWD8WAuZhwJ6UDPIQe2	167	
2eTjl0E2EAgfS1d9KRvp0f3Wme32	188	
2oLP1U9xBifOACOrUoo2NI2Ycid2	163	3
2yEsCDTUyyMzFvlRJ9X6mBUyJAx1	174	
6i8XvcRUPLLUxf3cZHkDvGW2L33	160	

Type GCS
Location gs://dataprep-staging-fdf310fb-5ba0-4b7e-80c5-cf4e3f4a907e/CSV/usuarios.csv
File Size 20.77kB
Size 65 columns · 4 types

Ilustración 12: Usuarios CSV

Tiene 65 columnas y 4 tipos de datos diferentes, un vistazo de ellos.

RBC	Name	#	Value.altura	#	Value.anosresidente	RBC	Value.edad	RBC	Value.ejerciciofisico	E	Value.email	RBC	Value.especialidad	RBC
11fCM64PIJ0y0qXz30jMMacFe0j2		132		2		20-24	ocasionalmente			uemusuario@gmail.com		Análisis Clínicos		
1pNdh2sUg0Xr0vpx3xPbmav0sz1		164		3		25-29	habitualmente			Hs56q.com		Psiquiatría		
2U15pXEP7#DBDAuZhWJ0UDP1Qe2		167				40-44	habitualmente			Hs81q.com		Otros		
2eTj10E2Af5f1d9KrVp0f3Wm32		188				25-29	habitualmente			luisEse.com		Otros		
2oLP1U9XbFOCoUoo2N2Yci2d		163				40-44	habitualmente			esteban@gmail.com		Análisis Clínicos		
2yEc0DfUyMfFvIRJ9X6nBlyJax1		174				45-49	habitualmente			Hs78q.com		Medicina Familiar y Comunitaria		
618XvRPUUxxsf3zKhDvGWL3		168				40-44	ocasionalmente			bPq.com		Psiquiatría		
6zezcOB1sdv03w5170Rnb4002		132				25-29	ocasionalmente			1Pq.com		Anatomía Patológica		
RBC	Value.estudios	#	Value.hijos	RBC	Value.id	RBC	Value.leg	RBC	Value.musica	RBC	Value.password	#	Value.peso	RBC
habitualmente	1		11fCM64PIJ0y0qXz30jMMacFe0j2		ocasionalmente		habitualmente			uer2018.		42		nunca
habitualmente	0		1pNdh2sUg0Xr0vpx3xPbmav0sz1		ocasionalmente		habitualmente			uer2018.		52		habitualmente
ocasionalmente	2		2U15pXEP7#DBDAuZhWJ0UDP1Qe2		ocasionalmente		habitualmente			uer2018.		58		ocasionalmente
ocasionalmente	0		2eTj10E2Af5f1d9KrVp0f3Wm32		nunca		ocasionalmente			ABC120		75		ocasionalmente
habitualmente	3		2oLP1U9XbFOCoUoo2N2Yci2d		ocasionalmente		habitualmente			123456		71		ocasionalmente
habitualmente	2		2yEc0DfUyMfFvIRJ9X6nBlyJax1		nunca		habitualmente			uer2018.		79		ocasionalmente
ocasionalmente	1		618XvRPUUxxsf3zKhDvGWL3		ocasionalmente		ocasionalmente			123456		58		ocasionalmente
RBC	Value.sexo	#	Value.tiempoplazaactual	#	Value.tiempovidalaboral	RBC	Value.tipocontrato	RBC	Value.tipotrabajo	#	Value.ulimaencuesta.ae			
hombre	2		2		residente		ambulatorio					31		
mujer	3		3		residente		hospitalario					19		
mujer	13		16		adjunto		urgencias					21		
hombre	3		4		adjunto		hospitalario					8		
mujer	3		4		residente		ambulatorio					54		
mujer	14		17		adjunto		urgencias					37		
			17											
#	Value.ulimaencuesta.d	RBC	Value.ulimaencuesta.q	#	Value.ulimaencue		Value.viajes	#	Value.horasActCuidados	#	Value.horasActFisica			
19			Mejor sueldo		32							8		
15			Mejorar el trabajo entre profesionales		44							1		
12			"Disminuir las horas de trabajo a la >		32									
0					0									
24			Segunda encuesta Esteban Horarios, retribuciones, incentivos, etc.		0									
16					26							4		
#	Value.horasActGratificantes	#	Value.horasActSocial	RBC	Value.contratoadju	RBC	Value.contratoadjunto	RBC	Columna31	RBC	Columna32	RBC	column31	
12		12												
2		3		eventual		eventual								
			interino			interino								
5		1		fijo		fijo								
				fijo		fijo								

Ilustración 13: Detalle Usuarios csv

Se observa que los datos necesitan un proceso de limpieza, para eliminar columnas vacías, por ejemplo, a partir de la columna 31, igualmente es necesario crear un id para identificar a cada usuario, eliminar algunas columnas innecesarias como "Value.password", este proceso se realizará con Dataprep y se detalla en el próximo apartado.

4.4.2.2 CSV – Estados de Ánimo

Guarda la información relativa al estado de ánimo de usuarios que participaron en la encuesta.



RBC	column2	RBC	column3	ABC
Name	Value.estado.estadoAnimo	Value		
#2NOMBRE?	triste	Cbpdi		
#2NOMBRE?	contento	Cmce		
#2NOMBRE?	triste	JTDsi		
#2NOMBRE?	normal	yXkVf		
#2NOMBRE?	contento	SKlgh		
-L2ZXERyI2YfBrcao-9A	triste	cOoD		

Type GCS
Location gs://dataprep-staging-fdf310fb-5ba0-4b7e-80c5-cf4e3f4a907e-/CSV/EstadoDeAnimo.csv
File Size 427B
Size 5 columns · 1 type

Ilustración 14: CSV - Estados de Ánimo

Tiene 5 columnas y un tipo de datos, un vistazo de ellos.

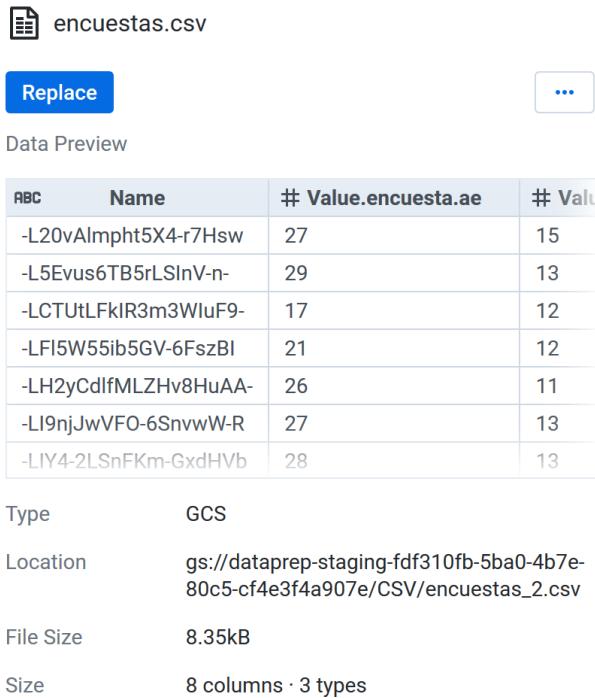
RBC	column2	RBC	column3	RBC	column4	RBC	column5	RBC	column6
Name			Value.estado.estadoAnimo		Value.id		Value.estado.estadoAnimo2		Value.estado.estadoAnimo3
#_NOMBRE?	triste		CbpdnySjRcXUfh0m75ZzeufUAvn2				triste		normal
#_NOMBRE?	contento		CmecJsdZwUpTcmtoYapsjUvZG02				contento		triste
#_NOMBRE?	triste		JTDsm0oZt40Sk7zCwVzfA10jc93						
#_NOMBRE?	normal		yXKVApp12nfwnCeC9e2rn6R0i1S2						
_NOMBRES			oXfLhQqkTQnOjLcAjLcQyOxOe2						

Ilustración 15: Detalle Estados de Ánimo csv

Se observa que el nombre de las columnas esta en la primera fila de los datos, por lo cual hay que convertir esa primera fila en cabecera, luego se observa que no tienen identificador simple, por lo que habrá que relacionar “Value.id” con su contraparte en los demás csv, este proceso se realizará con Dataprep y se detalla en el próximo apartado.

4.4.2.3 CSV – Encuestas

Guarda la información relativa a las encuestas de los usuarios.



RBC	Name	# Value.encuesta.ae	# Value.encuesta.d	# Value.encuesta.q	# Value.encuesta.rp	Value
-L20vAlmpht5X4-r7Hsw	27	15	24	32	32	2018-06-24T09:22:29.541Z
-L5Evus6TB5rLSInV-n-	29	13	23	30	30	2018-07-10T11:27:27.928Z
-LCTUtLFkIR3m3WluF9-	17	12	37	37	37	2018-07-24T05:42:01.277Z
-LFI5W55ib5GV-6FszBI	21	12	27	27	27	2018-07-28T22:48:16.225Z
-LH2yCdIfMLZHv8HuAA-	26	11	26	26	26	2018-08-29T11:28:03.675Z
-LI9njJwVF0-6SnvwW-R	27	13	34	34	34	2018-08-29T11:28:03.679Z
-LIY4-2LSnFKm-GxdHVb	28	13				

Type: GCS
Location: gs://dataprep-staging-fdf310fb-5ba0-4b7e-80c5-cf4e3f4a907e/CSV/encuestas_2.csv
File Size: 8.35kB
Size: 8 columns · 3 types

Ilustración 16: CSV – Encuestas

Se observa que tiene 8 columnas con 3 tipo de datos diferentes, un vistazo de ellos.

RBC	Name	#	Value.encuesta.ae	#	Value.encuesta.d	RBC	Value.encuesta.q	#	Value.encuesta.rp	Value
-L20vAlmpht5X4-r7Hsw	27		15		subir los sueldos a los medicos a minimo 4000 euros		24			2018-06-24T09:22:29.541Z
-L5Evus6TB5rLSInV-n-	29		13		Dismuir las horas de trabajo a la semana o lo q es lo min		32			2018-07-10T11:27:27.928Z
-LCTUtLFkIR3m3WluF9-	17		12		Evitar conductas persecutorias control de horarios libran		37			2018-07-24T05:42:01.277Z
-LFI5W55ib5GV-6FszBI	21		12		Escuchar los problemas de los trabajadores		27			2018-07-28T22:48:16.225Z
-LH2yCdIfMLZHv8HuAA-	26		11		Cuidar los espacios de consulta de descanso para el profes		26			2018-08-29T22:48:16.229Z
-LI9njJwVF0-6SnvwW-R	27		13		Menos horas de trabajo,mantener el equilibrio		34			2018-08-29T11:28:03.675Z
Value.encuesta.created_at										
2018-06-24T09:22:29.541Z										
2018-07-10T11:27:27.928Z										
2018-07-24T05:42:01.277Z										
2018-07-28T22:48:16.225Z										
2018-08-29T22:48:16.229Z										
2018-08-29T11:28:03.675Z										
2018-08-29T11:28:03.679Z										

Ilustración 17: Detalle Encuestas csv

Los datos necesitan ser relacionados con el csv de usuarios, para poder asociar la encuesta a cada uno de los participantes, generaremos un identificador, se eliminarán columnas de fechas, este proceso se realizará con Dataprep y se detalla en el próximo apartado.

4.4.2.4 CSV – Raw-Ahí

Guarda la información relativa a los datos fisiológicos de los usuarios.

RBC	Name	Value.created_at
-LCvhTeHz0vEHoiRmhyD		2018-05-17T04:01:48.866Z

Type: GCS
Location: gs://dataprep-staging-fdf310fb-5ba0-4b7e-80c5-cf4e3f4a907e/CSV/raw-ahi.csv
File Size: 128.85MB
Size: 14 columns · 4 types

Ilustración 18: CSV Raw Ahí

Se observa que tiene 14 columnas con 4 tipo de datos diferentes, un vistazo de ellos.

RBC	Name	Value.created_at	Value.email	Value.raw.activities-heart.dateTime	Value.raw.activities-heart.value.hearRateZones.caloriesOut
-LCvhTeHz0vEHoiRmhyD		2018-05-17T04:01:48.866Z	mti.jgaytan@gmail.com	2018-05-29	454,43334
-LCvhTeHz0vEHoiRmhyD		2018-05-17T04:01:48.866Z	mti.jgaytan@gmail.com	2018-05-29	454,43334
-LCvhTeHz0vEHoiRmhyD		2018-05-17T04:01:48.866Z	mti.jgaytan@gmail.com	2018-05-29	454,43334
-LCvhTeHz0vEHoiRmhyD		2018-05-17T04:01:48.866Z	mti.jgaytan@gmail.com	2018-05-29	454,43334
-LCvhTeHz0vEHoiRmhyD		2018-05-17T04:01:48.866Z	mti.jgaytan@gmail.com	2018-05-29	454,43334
-LCvhTeHz0vEHoiRmhyD		2018-05-17T04:01:48.866Z	mti.jgaytan@gmail.com	2018-05-29	454,43334
-LCvhTeHz0vEHoiRmhyD		2018-05-17T04:01:48.866Z	mti.jgaytan@gmail.com	2018-05-29	454,43334
# Value.raw.activities-heart.value.hearRateZones.max		# Value.raw.activities-heart.value.hearRateZones.max	# Value.raw.activities-heart.value.hearRateZones.minutes	# Value.raw.activities-heart.value.hearRateZones.minutes	# Value.raw.activities-heart.value.hearRateZones.minutes
89		30	338	338	Out of Range
89		30	338	338	Out of Range
89		30	338	338	Out of Range
89		30	338	338	Out of Range
89		30	338	338	Out of Range
89		30	338	338	Out of Range
89		30	338	338	Out of Range
# Value.raw.activities-heart.value.restingHeartRate		# Value.raw.activities-heart.intraday.dataset.time	# Value.raw.activities-heart.intraday.dataset.value	# Value.raw.activities-heart.intraday.dataset.value	# Value.raw.activities-heart.intraday.dataset.value
63		00:00:00	62	1	1
63		00:01:00	61	1	1
63		00:02:00	63	1	1
63		00:03:00	66	1	1
63		00:04:00	66	1	1
63		00:05:00	63	1	1
63		00:06:00	63	1	1
RBC	Value.raw.activities-heart.intraday.datasetType		62	1	1
minute					

Ilustración 19: Detalle Raw - Ahí csv

Se observa que en la frecuencia de medición es por minuto, los datos necesitan ser relacionados con el csv de usuarios, para poder asociar los datos fisiológicos a cada uno de los participantes, generaremos un identificador, se eliminarán columnas de fechas, este proceso se realizará con Dataprep y se detalla en el próximo apartado.

4.4.2.5 CSV – Raw-Sleep

Guarda la información relativa a los datos de sueño de los usuarios.

Data Preview			
ABC	Name	Value.created_at	
-LCvhThtUFL3JSvASlyv		2018-05-18T04:01:48.866Z	mt
Type	GCS		
Location		gs://dataprep-staging-fdf310fb-5ba0-4b7e-80c5-cf4e3f4a907e/CSV/raw-sleep.csv	
File Size	18.8MB		
Size	34 columns · 5 types		

Ilustración 20: CSV - Raw-Sleep

Se observa que tiene 34 columnas con 5 tipo de datos diferentes, un vistazo de ellos

#	Value.raw.sleep.levels.summary.deep.thirtyDayAvgMinutes	v	#	Value.raw.sleep.levels.summary.light.count	v	#	Value.raw.sleep.levels.summary.light.minutes	v
	0			7			121	
	0			7			121	
	0			7			121	
	0			7			121	
	0			7			121	
	0			7			121	
#	Value.raw.sleep.levels.summary.light.thirtyDayAvgMinutes	v	#	Value.raw.sleep.levels.summary.rem.count	v	#	Value.raw.sleep.levels.summary.rem.minutes	v
	0			2			48	
	0			2			48	
	0			2			48	
	0			2			49	
	0			2			48	
	0			2			48	
	0			2			48	
#	Value.raw.sleep.levels.summary.rem.thirtyDayAvgMinutes	v	#	Value.raw.sleep.levels.summary.wake.count	v	#	Value.raw.sleep.levels.summary.wake.minutes	v
	0			8			35	
	0			8			35	
	0			8			35	
	0			8			35	
	0			8			35	
	0			8			35	
#	Value.raw.sleep.levels.summary.wake.thirtyDayAvgMinutes	v	#	Value.raw.sleep.logId	v	#	Value.raw.sleep.minutesAfterWakeUp	v
	0			18264053339			0	
	0			18264053339			0	
	0			18264053339			0	
	0			18264053339			0	
	0			18264053339			0	
	0			18264053339			0	
#	Value.raw.sleep.minutesAwake	v	#	Value.raw.sleep.minutesToFallAsleep	v	(Value.raw.sleep.startTime	v
	35			0	2018-05-19T16:09:30.000			RBC
	35			0	2018-05-19T16:09:30.000			Value.raw.sleep.type
	35			0	2018-05-19T16:09:30.000			
	35			0	2018-05-19T16:09:30.000			
	35			0	2018-05-19T16:09:30.000			
	35			0	2018-05-19T16:09:30.000			
	35			0	2018-05-19T16:09:30.000			

Ilustración 21: Detalle Raw-Sleep csv

Se observa que en la frecuencia de medición es por minuto, los datos necesitan ser relacionados con el csv de usuarios, para poder asociar los datos de sueño a cada uno de los participantes, generaremos un identificador, se eliminarán columnas de fechas, este proceso se realizará con Dataprep y se detalla en el próximo apartado.

4.4.3 Dataprep – Dataflow

Dataprep es un servicio que ofrece Google Cloud Platform para hacer un proceso de ETL (Extracción, Transformación y Carga) de un conjunto de datos, permitiendo realizar operaciones costosas en tiempo y recursos, levantando además instancias con **Dataflow**. De esta manera, se pueden procesar conjuntos de datos muy grandes con una interfaz amigable y con un gran poder de cómputo detrás.

Para realizar esto, se utiliza el siguiente flujo de trabajo:

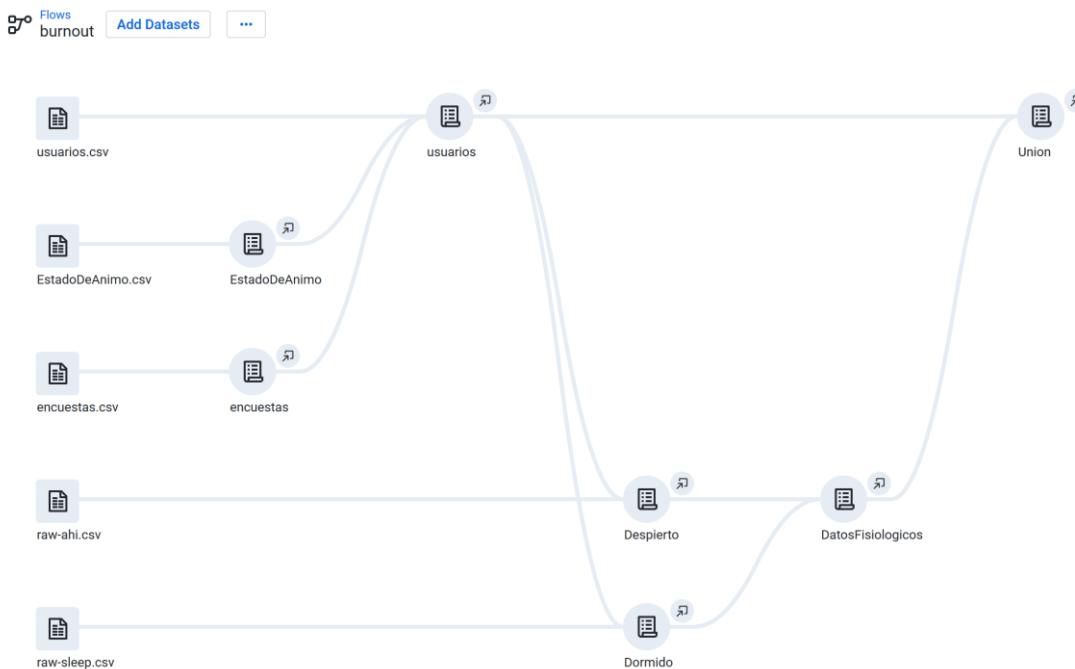


Ilustración 22: Flujo de Trabajo Dataprep en Google Cloud Platform

En Dataprep se cargan los csv como datos de entrada y se aplican distintos *récipes*¹, es decir, transformaciones, para limpiar y unir los datos según lo que se quiera obtener.

Anteriormente detallamos cada uno de los ficheros csv a utilizar, con esta herramienta procederemos a realizar el proceso ETL que detallaremos a continuación para cada uno de ellos.

Este proceso ETL (Extracción, Transformación y Carga) se realiza debido a la necesidad de limpiar los ficheros de entrada, es decir, para el desarrollo del proyecto es necesario obtener un solo fichero csv, con todos los datos para poder crear un algoritmo de aprendizaje, que logre predecir el síndrome de burnout.

Igualmente necesitamos crear una columna adicional, donde se categorice los datos de cada uno de los pacientes, según la lógica de las encuestas Maslach, asignando Verdadero o Falso para el síndrome de burnout; esto nos permitirá luego utilizar el dataset resultante de entrada para el algoritmo de aprendizaje automático.

¹ Récipe es la columna que contiene los conjuntos de transformaciones para limpiar los datos.

El récipe “Despierto” contiene los datos fisiológicos principales, los unimos con cada uno de sus usuarios, renombramos variables y validamos tipos.

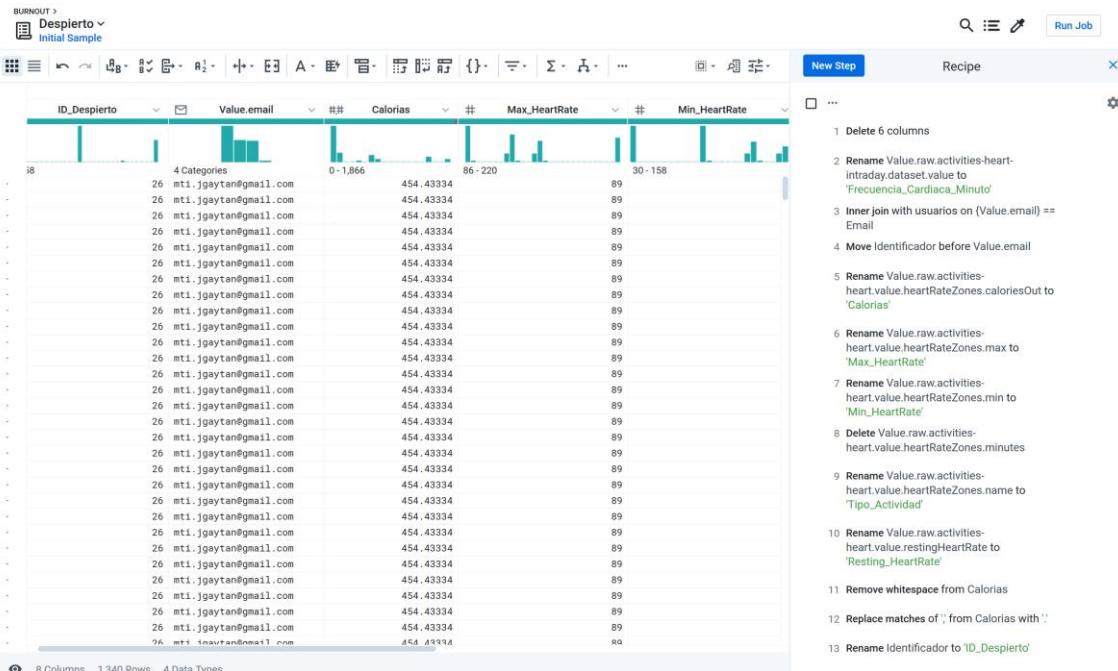


Ilustración 23: Datos Despierto Dataprep

El récipe “Dormido” contiene los datos de sueño, los unimos con cada uno de sus usuarios, renombramos variables y validamos tipos.

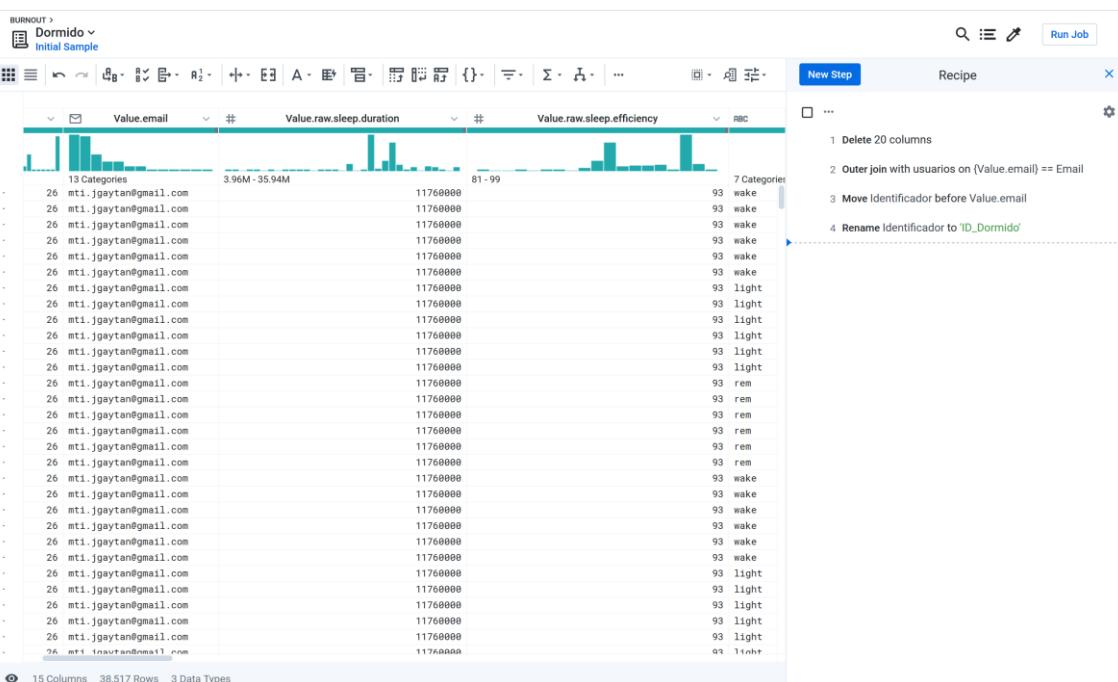


Ilustración 24: Datos Dormido Dataprep

En el récipe “usuarios”, se eliminan columnas con datos repetidos y se crea un ID único para cada usuario, se renombran variables y se construye la columna de Verdadero y Falso para burnout, siguiendo la lógica de las encuestas Maslach.

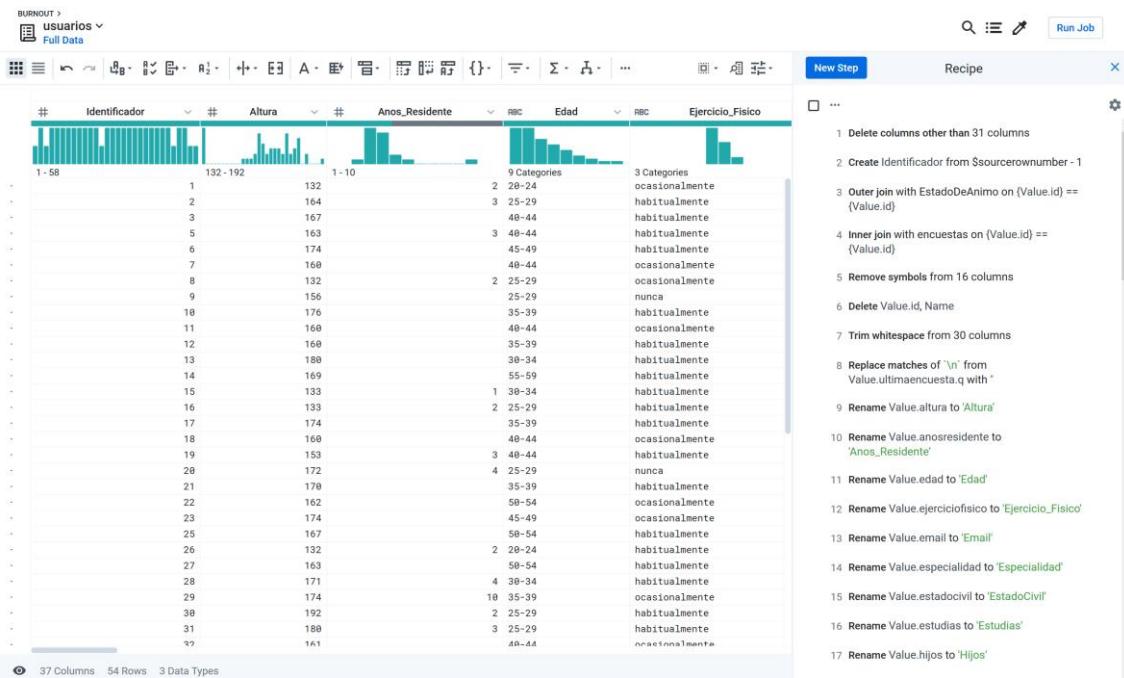


Ilustración 25: Datos de usuarios

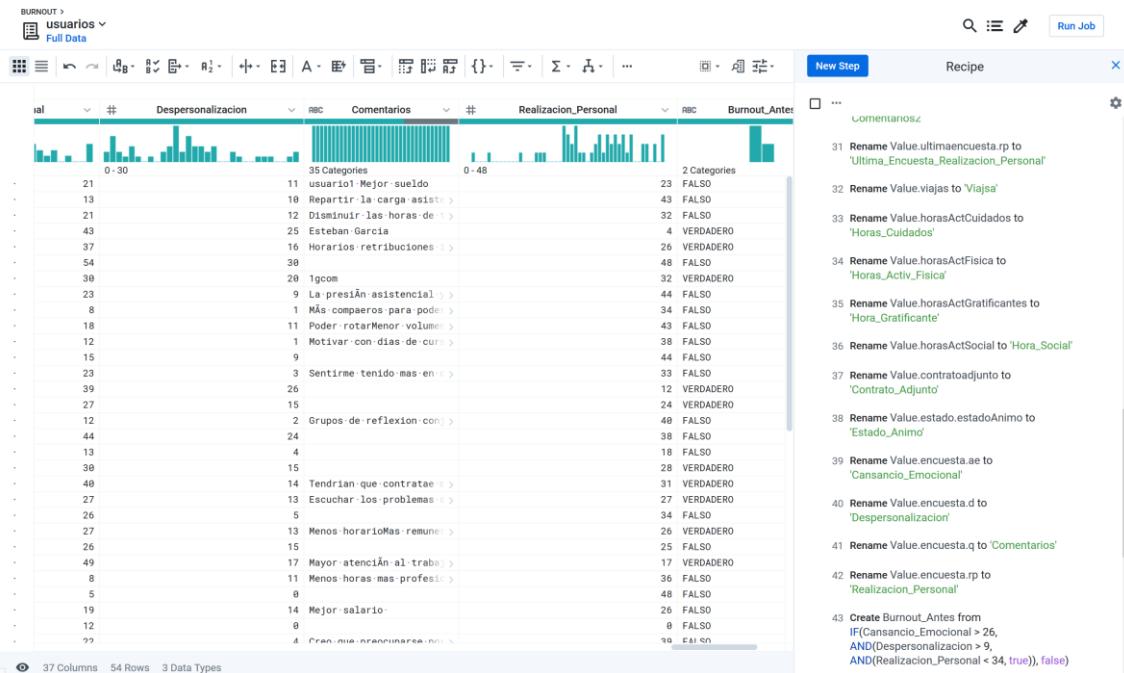


Ilustración 26: Datos de usuarios Burnout

En el récipe “datos fisiológicos”, se unen los datos de sueño con los datos fisiológicos, las transformaciones que se aplicaron, son eliminar algunas columnas repetidas en ambos datasets y unirlos a través del ID que generamos utilizando el ID de los usuarios.

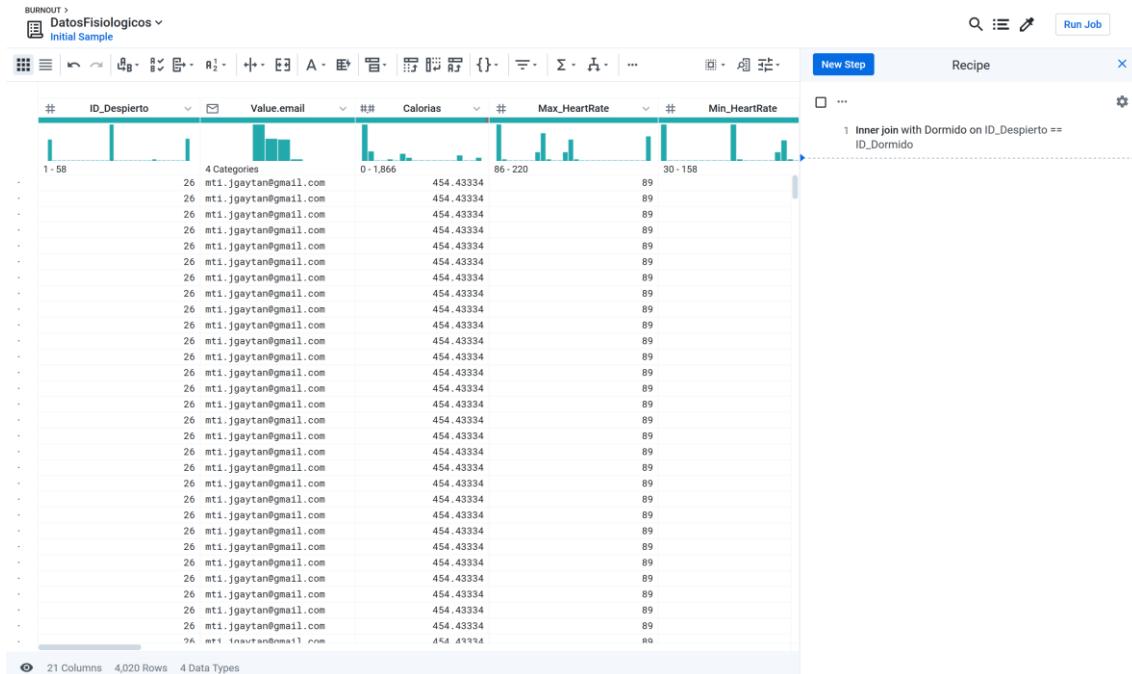


Ilustración 27: Datos Fisiológicos Dataprep

En el récipe “estados de ánimo” se realiza una limpieza de ciertas columnas y se convierte la primera línea en encabezado.

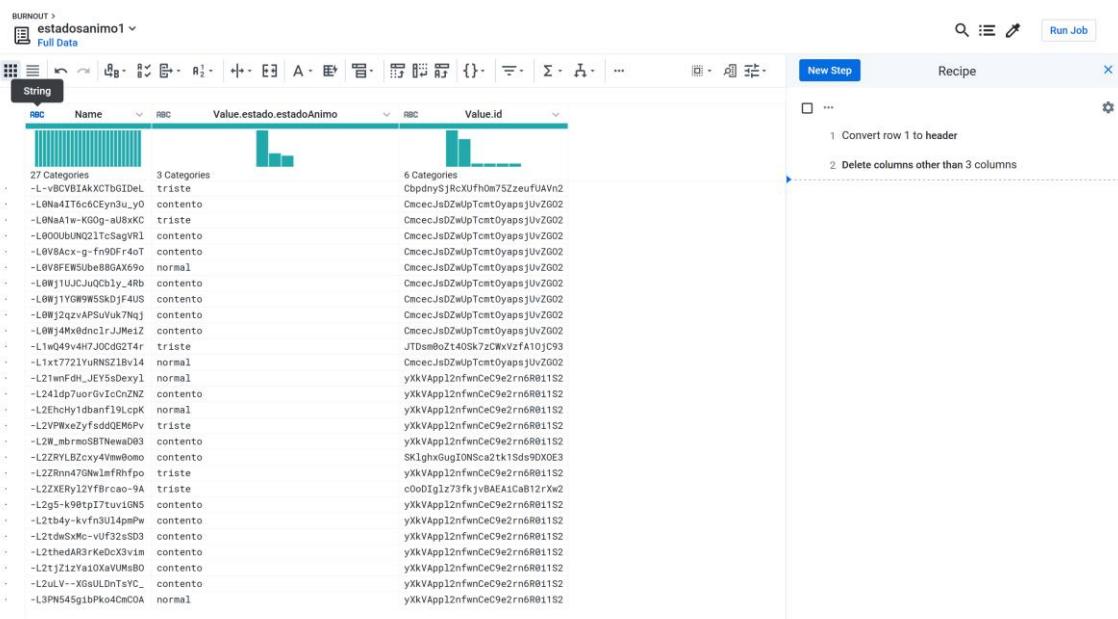


Ilustración 28: Datos de estados de ánimo

En el récipe “Unión” se hace una unión de las encuestas con los diferentes usuarios, con sus estados de ánimo y con sus datos fisiológicos que contienen los datos del sueño.

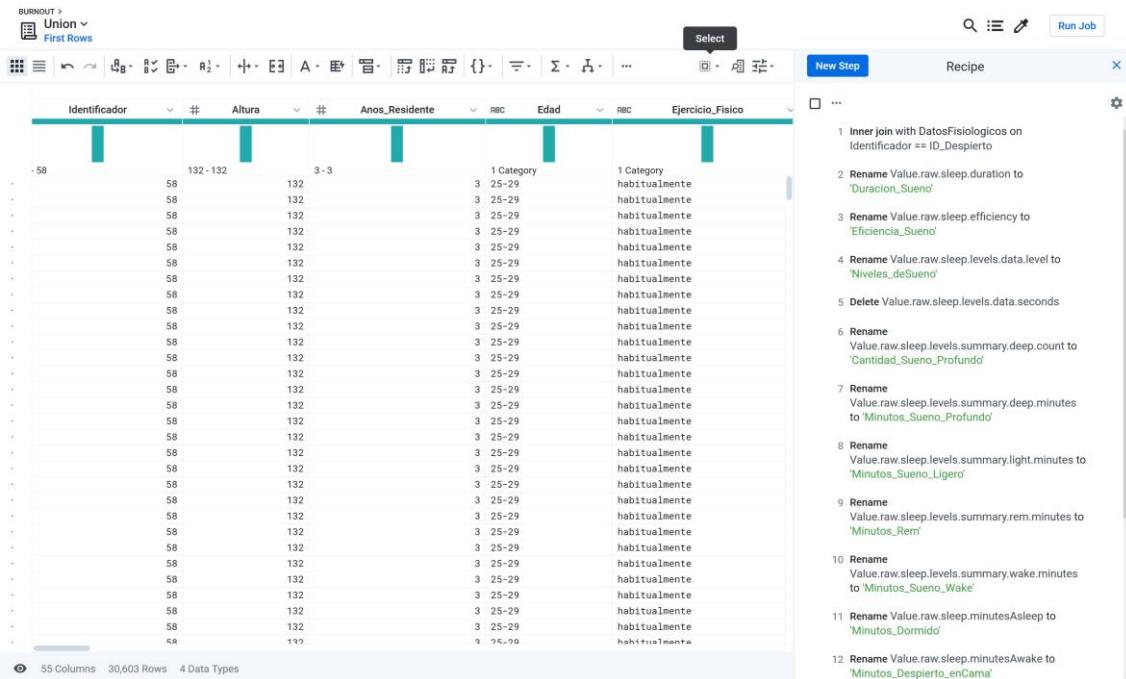


Ilustración 29: Datos unión

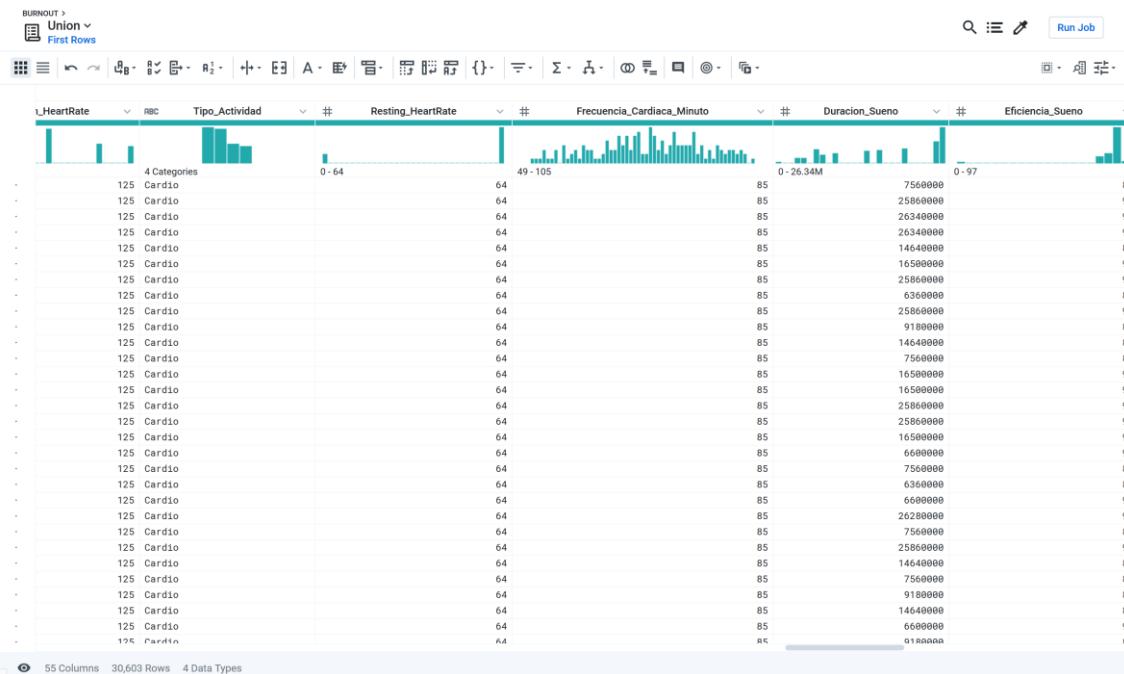


Ilustración 30: Datos unión 2

Una vez esté todo terminado, se realiza la exportación del dataset a Google Storage, con el nombre: “Burnout_Data.csv”, el cual contiene 55 columnas, 3 millones de filas y un peso de 1.03 GB.

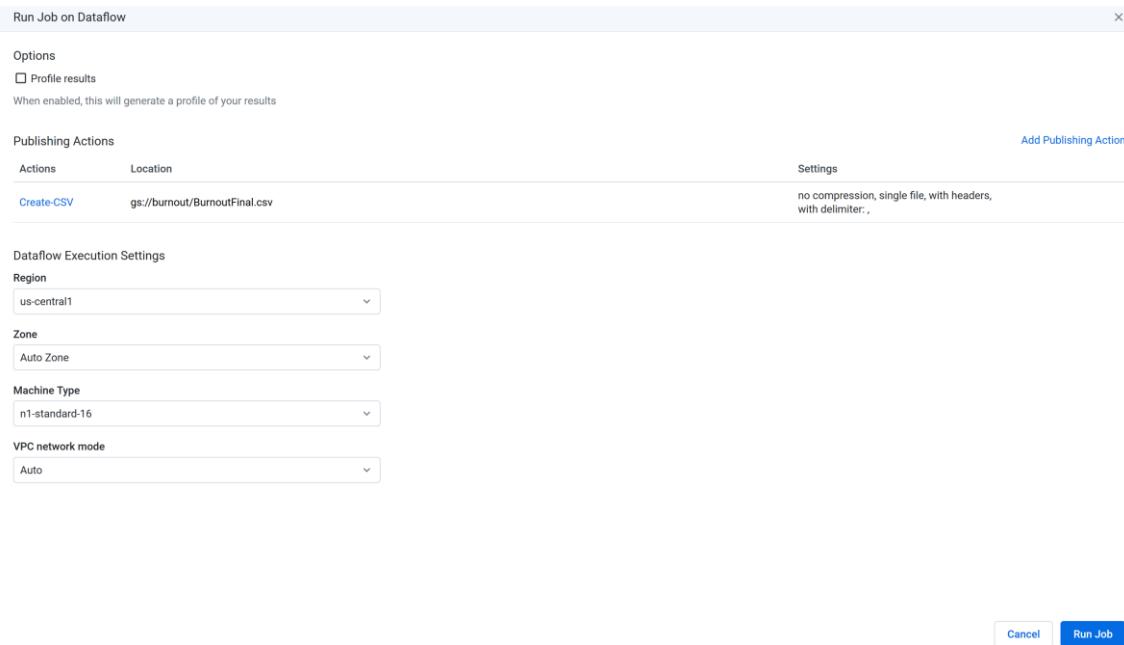


Ilustración 31: Exportación de los datos

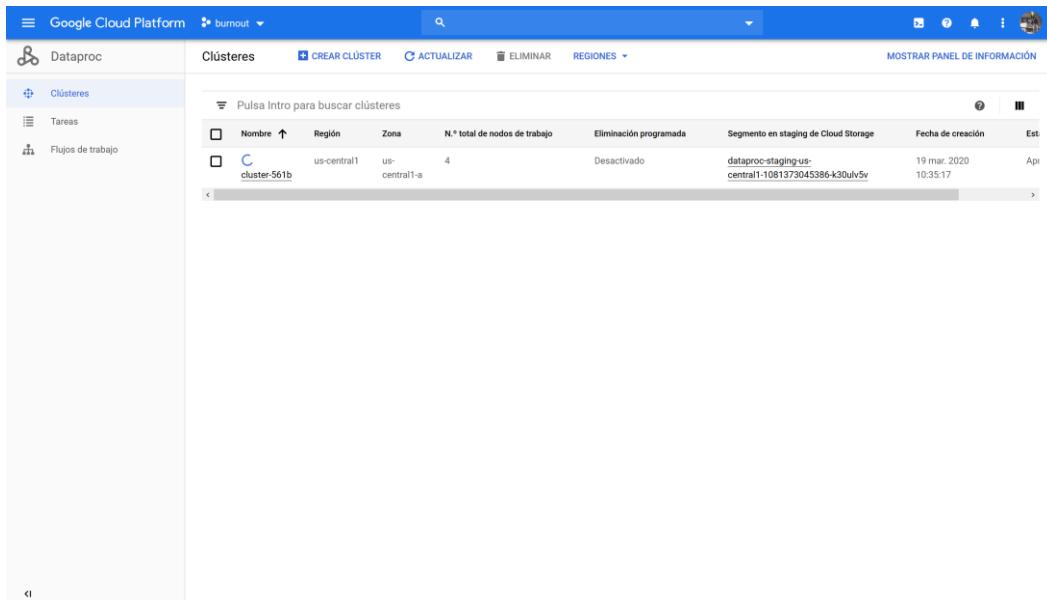
4.4.4 Cloud Data Storage

Cloud Storage es un servicio que nos ofrece GCP para guardar cualquier tipo de dato. Este ha sido utilizado para guardar ficheros antes del ETL, después del ETL y modelos una vez generados. Para ello se creó un segmento llamado burnout.

4.4.5 Clúster Dataproc

En este servicio es en el que más se ha invertido tiempo y créditos, ya que para entrenar los modelos de manera distribuida se ha utilizado Dataproc, debido a que este permite ejecutar aplicaciones de Spark de manera distribuida. Asimismo, las características de la instancia que se utilizó es un **nodo maestro** de 4 CPU y 15 GB de RAM con 500 GB de disco duro. Luego de esto, se configuraron los **nodos de trabajo** de 4 CPU y 15 GB de RAM con 500 GB de disco duro. En total, se utilizaron 4 nodos de trabajo con estas características.

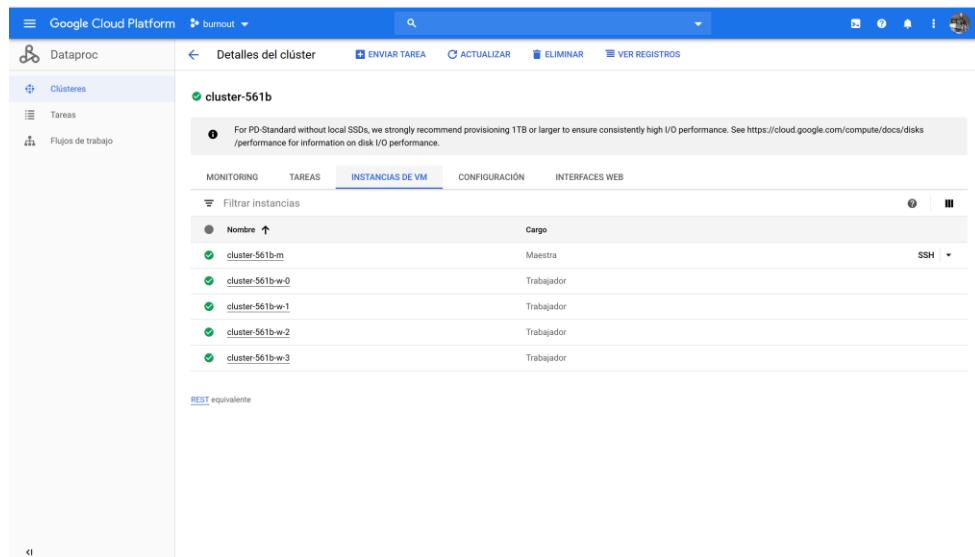
Posterior a esto, se colocó como imagen para esta configuración (Ubuntu 18.04 LTS, Hadoop 2.9 y Spark 2.4).



The screenshot shows the Google Cloud Platform DataProc interface. In the top navigation bar, there are tabs for 'Clústeres' (Clusters), 'CREAR CLÚSTER' (Create Cluster), 'ACTUALIZAR' (Update), 'ELIMINAR' (Delete), and 'REGIONES' (Regions). A search bar is also present. On the left sidebar, under the 'DataProc' section, there are links for 'Clústeres', 'Tareas', and 'Flujos de trabajo'. The main content area displays a table for clusters. The table has columns: Nombre (Name), Región (Region), Zona (Zone), N.º total de nodos de trabajo (Total number of worker nodes), Eliminación programada (Scheduled deletion), Segmento en staging de Cloud Storage (Cloud Storage staging segment), Fecha de creación (Creation date), and Est. (Status). One cluster, 'cluster-561b', is listed with the following details:

Nombre	Región	Zona	N.º total de nodos de trabajo	Eliminación programada	Segmento en staging de Cloud Storage	Fecha de creación	Est.
cluster-561b	us-central1	us-central1-a	4	Desactivado	dataproc-staging-us-central1-1081373045386-k30ulv5v	19 mar. 2020 10:35:17	Apagado

Ilustración 32: DataProc en Google Cloud Platform

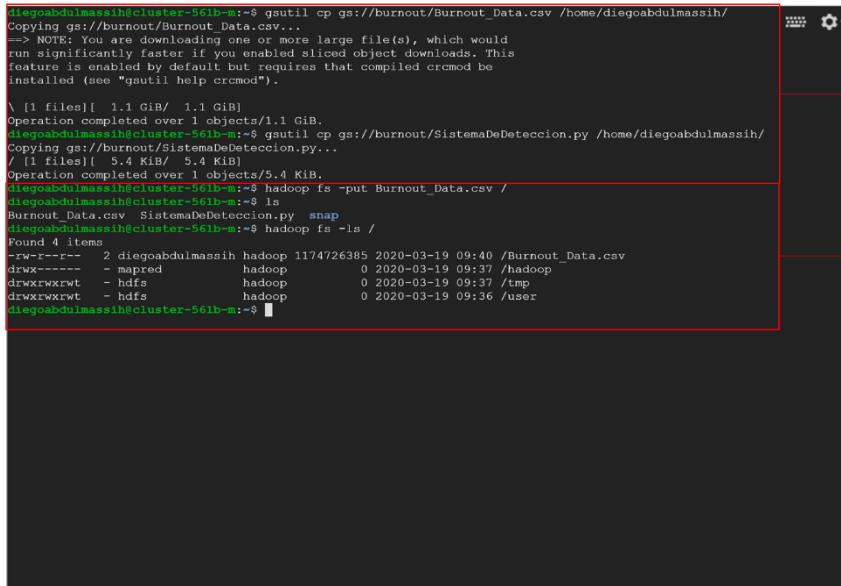


The screenshot shows the 'Detalles del clúster' (Cluster Details) page for 'cluster-561b'. At the top, there are buttons for 'ENVIAR TAREA' (Send Task), 'ACTUALIZAR' (Update), 'ELIMINAR' (Delete), and 'VER REGISTROS' (View Logs). Below this, the cluster name 'cluster-561b' is displayed with a green checkmark icon. A note states: 'For PD-Standard without local SSDs, we strongly recommend provisioning 1TB or larger to ensure consistently high I/O performance. See https://cloud.google.com/compute/docs/disk/performance for information on disk I/O performance.' The main content area is titled 'INSTANCIAS DE VM' (VM Instances). It includes tabs for 'MONITORING', 'TAREAS', 'INSTANCIAS DE VM' (selected), 'CONFIGURACIÓN', and 'INTERFACES WEB'. A filter for 'Nombre' (Name) is shown. The table lists five instances:

Nombre	Cargo	SSH
cluster-561b-m	Maestra	SSH
cluster-561b-w-0	Trabajador	
cluster-561b-w-1	Trabajador	
cluster-561b-w-2	Trabajador	
cluster-561b-w-3	Trabajador	

Ilustración 33: DataProc en Google Cloud Platform

Una vez que el clúster funcione, se procede a utilizar una conexión SSH para ejecutar el código (el detalle del código se podrá visualizar en el Capítulo 4).



```

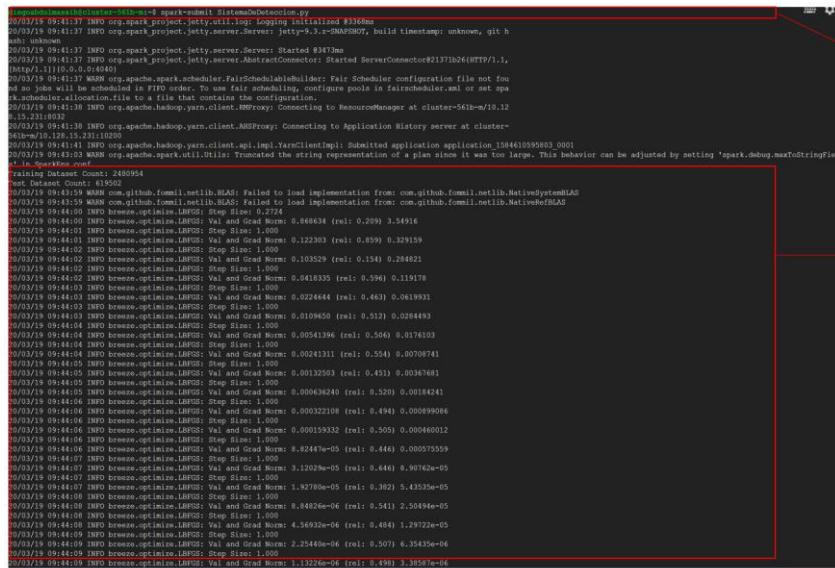
diegoabdulmassih@cluster-561b-m:~$ gsutil cp gs://burnout/Burnout_Data.csv /home/diegoabdulmassih/
Copying gs://burnout/Burnout_Data.csv...
=> NOTE: You are downloading one or more large file(s), which would
run significantly faster if you enabled sliced object downloads. This
feature is enabled by default but requires that compiled crcmod be
installed (see "gsutil help crcmod").
\ [1 files]! 1.1 GiB/ 1.1 GiB
Operation completed over 1 objects/1.1 GiB.
diegoabdulmassih@cluster-561b-m:~$ gsutil cp gs://burnout/SistemaDeDeteccion.py /home/diegoabdulmassih/
Copying gs://burnout/SistemaDeDeteccion.py...
\ [1 files]! 5.4 KiB/ 5.4 KiB
Operation completed over 1 objects/5.4 KiB.
diegoabdulmassih@cluster-561b-m:~$ hadoop fs -put Burnout_Data.csv /
diegoabdulmassih@cluster-561b-m:~$ ls
Burnout_Data.csv SistemaDeDeteccion.py
diegoabdulmassih@cluster-561b-m:~$ hadoop fs -ls /
Found 4 items
-rw-r--r-- 2 diegoabdulmassih hadoop 1174726385 2020-03-19 09:40 /Burnout_Data.csv
drwx----- mapred hadoop 0 2020-03-19 09:37 /hadoop
drwxrwxrwt - hdfs hadoop 0 2020-03-19 09:37 /tmp
drwxrwxrwt - hdfs hadoop 0 2020-03-19 09:36 /user
diegoabdulmassih@cluster-561b-m:~$ 

```

Con gsutil descargamos los ficheros Burnout_Data.csv y SistemaDeDeteccion.py de Google Storage a nuestro Clúster

Ejecutando el comando "hadoop fs -put "Burnout_Data.csv /" colocamos los datos del Burnout de manera distribuida en Hadoop, para su posterior consumo por el fichero Python.

Ilustración 34:Conexión SSH Dataproc



```

diegoabdulmassih@cluster-561b-m:~$ spark-submit SistemaDeDeteccion.py
20/03/19 09:41:37 INFO org.spark.project.jetty.server.Server: Logging initialized #3160ms
20/03/19 09:41:37 INFO org.spark.project.jetty.server.Server: Started #3473ms
20/03/19 09:41:37 INFO org.spark.project.jetty.server.AbstractConnector: Started ServerConnector@2137fb26[HTTP/1.1,
20/03/19 09:41:37 WARN org.apache.spark.scheduler.FairSchedulableBuilder: Fair Scheduler configuration file not found
no job will be scheduled in FIFO order. To use fair scheduling, configure pools in fairscheduler.xml or set spa
20/03/19 09:41:37 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at cluster-561b-m/10.12
4.15.231.80:2388
20/03/19 09:41:38 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to Application History server at cluster-
4.15.231.102:2000
20/03/19 09:41:41 WARN org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Submitted application application_184461059503_0001
20/03/19 09:41:41 INFO org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.debug.maxToStringFields'
20/03/19 09:41:42 INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat: Training Dataset Count: 2480954
20/03/19 09:41:59 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from com.github.fommil.netlib.NativeSystemBLAS
20/03/19 09:43:59 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from com.github.fommil.netlib.NativeRefBLAS
20/03/19 09:44:00 INFO breeze.optimization.LBFGS: Step Size: 0.2734
20/03/19 09:44:00 INFO breeze.optimization.LBFGS: Val and Grad Norm: 8.868434 (rel: 0.209) 3.54918
20/03/19 09:44:01 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:01 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.122303 (rel: 0.059) 0.329139
20/03/19 09:44:02 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:02 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.103529 (rel: 0.154) 0.284821
20/03/19 09:44:02 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:03 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:03 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.0418335 (rel: 0.594) 0.119178
20/03/19 09:44:03 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:03 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.0224444 (rel: 0.463) 0.0619931
20/03/19 09:44:04 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:04 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.0109450 (rel: 0.512) 0.0284493
20/03/19 09:44:04 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:04 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.00541396 (rel: 0.506) 0.0176103
20/03/19 09:44:04 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:04 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.002421331 (rel: 0.554) 0.0079741
20/03/19 09:44:05 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:05 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.00132503 (rel: 0.453) 0.00387648
20/03/19 09:44:05 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:05 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.000636240 (rel: 0.520) 0.00184241
20/03/19 09:44:06 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:06 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.000322108 (rel: 0.494) 0.000699006
20/03/19 09:44:06 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.000159332 (rel: 0.505) 0.000460012
20/03/19 09:44:06 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:06 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.000159332 (rel: 0.446) 0.000575559
20/03/19 09:44:07 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:07 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.000125000 (rel: 0.446) 0.000575559
20/03/19 09:44:07 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:07 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.000125000 (rel: 0.382) 5.47535e-05
20/03/19 09:44:08 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:08 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.000125000 (rel: 0.341) 2.50494e-05
20/03/19 09:44:08 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:08 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.000125000 (rel: 0.484) 1.29722e-05
20/03/19 09:44:09 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:09 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.000125000 (rel: 0.507) 0.356135e-06
20/03/19 09:44:09 INFO breeze.optimization.LBFGS: Step Size: 1.000
20/03/19 09:44:09 INFO breeze.optimization.LBFGS: Val and Grad Norm: 0.000125000 (rel: 0.499) 3.3858e-06
20/03/19 09:44:09 INFO breeze.optimization.LBFGS: Step Size: 1.000

```

Ejecutamos nuestro fichero de entrenamiento con el comando "spark-submit SistemaDeDeteccion.py"

Vemos la ejecución del código que detallaremos en el capítulo 4.

Esta parte corresponde a la ejecución del Pipeline donde hacemos la preparación de los datos, así como la división en un 70% para el entrenamiento y 30% para las pruebas.

También vemos que comienza la ejecución del primer algoritmo que hemos probado que es el LogisticRegression.

Ilustración 35:Dataproc Spark-Submit

```

20/03/19 09:44:07 INFO breeze.optimize.LBFGS: Val and Grad Norm: 3.12029e-05 (rel: 0.646) 8.90762e-05
20/03/19 09:44:07 INFO breeze.optimize.LBFGS: Val and Grad Norm: 1.92780e-05 (rel: 0.382) 5.43535e-05
20/03/19 09:44:08 INFO breeze.optimize.LBFGS: Step Size: 1.000
20/03/19 09:44:08 INFO breeze.optimize.LBFGS: Val and Grad Norm: 8.0826e-06 (rel: 0.541) 2.50494e-05
20/03/19 09:44:08 INFO breeze.optimize.LBFGS: Step Size: 1.000
20/03/19 09:44:08 INFO breeze.optimize.LBFGS: Val and Grad Norm: 4.46932e-06 (rel: 0.484) 1.29722e-05
20/03/19 09:44:09 INFO breeze.optimize.LBFGS: Step Size: 1.000
20/03/19 09:44:09 INFO breeze.optimize.LBFGS: Val and Grad Norm: 2.25440e-06 (rel: 0.507) 6.35435e-06
20/03/19 09:44:09 INFO breeze.optimize.LBFGS: Step Size: 1.000
20/03/19 09:44:09 INFO breeze.optimize.LBFGS: Val and Grad Norm: 1.13226e-06 (rel: 0.498) 3.38587e-06
20/03/19 09:44:09 INFO breeze.optimize.LBFGS: Converged because gradient converged
20/03/19 09:44:10 INFO breeze.optimize.LBFGS: Step Size: 1.000
20/03/19 09:44:10 INFO breeze.optimize.LBFGS: Val and Grad Norm: 2.05050e-07 (rel: 0.300) 7.25984e-07
20/03/19 09:44:10 INFO breeze.optimize.LBFGS: Step Size: 1.000
20/03/19 09:44:10 INFO breeze.optimize.LBFGS: Val and Grad Norm: 1.92920e-07 (rel: 0.101) 3.92573e-07
20/03/19 09:44:11 INFO breeze.optimize.LBFGS: Step Size: 1.000
20/03/19 09:44:11 INFO breeze.optimize.LBFGS: Val and Grad Norm: 6.68175e-08 (rel: 0.0659) 1.90694e-07
20/03/19 09:44:11 INFO breeze.optimize.LBFGS: Step Size: 1.000
20/03/19 09:44:11 INFO breeze.optimize.LBFGS: Val and Grad Norm: 3.39754e-08 (rel: 0.0299) 9.11731e-08
20/03/19 09:44:12 INFO breeze.optimize.LBFGS: Step Size: 1.000
20/03/19 09:44:12 INFO breeze.optimize.LBFGS: Val and Grad Norm: 2.33100e-08 (rel: 0.00971) 2.27193e-07
20/03/19 09:44:12 INFO breeze.optimize.LBFGS: Step Size: 1.000
20/03/19 09:44:12 INFO breeze.optimize.LBFGS: Val and Grad Norm: 7.0598e-09 (rel: 0.0148) 4.79376e-08
20/03/19 09:44:12 INFO breeze.optimize.LBFGS: Step Size: 1.000
20/03/19 09:44:13 INFO breeze.optimize.LBFGS: Step Size: 1.000
20/03/19 09:44:13 INFO breeze.optimize.LBFGS: Val and Grad Norm: 4.47457e-09 (rel: 0.00235) 2.73781e-08
20/03/19 09:44:13 INFO breeze.optimize.LBFGS: Step Size: 1.000
20/03/19 09:44:13 INFO breeze.optimize.LBFGS: Val and Grad Norm: 2.08679e-09 (rel: 0.00217) 1.08865e-08
20/03/19 09:44:13 INFO breeze.optimize.LBFGS: Step Size: 1.000
20/03/19 09:44:13 INFO breeze.optimize.LBFGS: Val and Grad Norm: 1.09123e-09 (rel: 0.000906) 4.98539e-09
20/03/19 09:44:13 INFO breeze.optimize.LBFGS: Converged because gradient converged
-----+-----+-----+-----+-----+-----+
|Burnout_Antes|label| rawPrediction|prediction| probability|
+-----+-----+-----+-----+-----+
| VERDADERO| 0.0|[21, 3395211419853,...]| 0.0|[0.999999999083901...|
-----+-----+-----+-----+-----+
only showing top 10 rows
Test Area Under ROC 1.0

```

La ejecución del algoritmo LogisticRegression termina y se visualiza que contamos con un ROC del 1.0

Ilustración 36: Dataproc Regresión Logística

```

-----+-----+-----+-----+-----+
|Burnout_Antes|label| rawPrediction|prediction| probability|
+-----+-----+-----+-----+-----+
| VERDADERO| 0.0|[18.0,0.0,0,0,0]| 0.0|[1.0,0.0,0,0,0]|
-----+-----+-----+-----+-----+
only showing top 10 rows
Test Area Under ROC 1.0

```

El segundo algoritmo que se ejecuta es el RandomForest el cual no genera logs por lo que solo se visualiza la tabla con las probabilidades y la predicción con un resultado igualmente de ROC 1.0

Ilustración 37: Dataproc Bosques Aleatorios

```

-----+-----+-----+-----+-----+
|Burnout_Antes|label| rawPrediction|prediction| probability|
+-----+-----+-----+-----+-----+
| VERDADERO| 0.0|[3.34643370674057...]| 0.0|[0.99876181213634...|
-----+-----+-----+-----+-----+
only showing top 10 rows
Test Area Under ROC 1.0

```

El tercer algoritmo que se ejecuta es el DecisionTree el cual no genera logs por lo que solo se visualiza la tabla con las probabilidades y la predicción con un resultado igualmente de ROC 1.0

Ilustración 38: Dataproc Árbol de Decisión

Una vez que se cuenta con los modelos entrenados (proceso que tardó 2 días en ejecutarse, debido al CrossValidation que se verá en el capítulo 4), se procede a guardarlos en Google Cloud Storage.

<input type="checkbox"/>  modelo_DecisionTree/	— Carpeta	—	—	Sujeto a las LCA de cada objeto	—	—
<input type="checkbox"/>  modelo_LogisticRegression/	— Carpeta	—	—	Sujeto a las LCA de cada objeto	—	—
<input type="checkbox"/>  modelo_Pipeline/	— Carpeta	—	—	Sujeto a las LCA de cada objeto	—	—
<input type="checkbox"/>  modelo_RandomForest/	— Carpeta	—	—	Sujeto a las LCA de cada objeto	—	—

Ilustración 39:Guardado de modelos

En este sentido, se puede visualizar también, que se genera un “modelo_Pipeline”, el cual hace referencia al tratamiento de los datos, por lo cual es necesario guardarlo para poder hacer predicciones de datos nuevos.

Capítulo 5. Algoritmo de Aprendizaje Automático Aplicado al Síndrome de Burnout

En este capítulo se detallará el código desarrollado en Spark, utilizando el dataset que se ha pre-procesado con Google Dataprep, explicado en el Capítulo 3.

El objetivo principal es conseguir desarrollar un modelo de Machine Learning que logre identificar el síndrome de burnout a través de las variables fisiológicas y no desde la encuesta Maslach. A su vez, también se han tomado en consideración los datos personales, datos laborales, hábitos sociales y de tiempo de ocio. Estas variables están incluidas dentro del dataset.

5.1 Conceptos básicos

Los conceptos básicos que se deben tener presentes para continuar con la lectura de este capítulo son palabras referentes al aprendizaje automático.

- Dataset: es el conjunto de datos o colección, habitualmente tabulada en donde cada columna representa una variable y cada fila un valor para dicha variable.
- Característica: son los diversos atributos que tiene un dataset, por lo cual normalmente se realiza un análisis de importancia.
- Modelo: surge una vez se ha entrenado el sistema y su finalidad es realizar predicciones partiendo de nuevos datos. Una de las fases más importantes para lograr esto, es la fase conocida como training o entrenamiento, en la que se ejecuta algoritmos para conseguir los patrones necesarios para crear el modelo.

5.2 Análisis de los datos

El análisis de los datos se realizó en el capítulo anterior, en el que se utilizó la herramienta Dataprep para la carga y el proceso ETL (Extracción, Transformación y Carga), donde se obtuvo entonces el dataset “Burnout_Data.csv”, el cual será utilizado en el entrenamiento.

Una de las primeras tareas que se deben plantear luego de haber realizado el proceso ETL, es un análisis de sus principales características.

5.2.1 Burnout Dataset

El dataset representa los datos fisiológicos, así como las variables de datos personales, datos laborales, hábitos sociales y de tiempo de ocio del personal médico de los hospitales Infanta Sofía y Son Llàtzer, recogidos en el periodo que comprende del 31 de mayo al 31 de agosto de 2018. Este se compone de 3.100.000 filas y 55 columnas.

Para un vistazo del dataset tenemos la figura 17 y figura 18, una vez cargado en Spark para su análisis, vemos que el esquema del dataset es el siguiente:

```
root
|-- Identificador: integer (nullable = true)
|-- Altura: integer (nullable = true)
|-- Anos_Residente: integer (nullable = true)
|-- Edad: string (nullable = true)
|-- Ejercicio_Fisico: string (nullable = true)
|-- Email: string (nullable = true)
|-- Especialidad: string (nullable = true)
|-- EstadoCivil: string (nullable = true)
|-- Estudios: string (nullable = true)
|-- Hijos: integer (nullable = true)
|-- Lectura: string (nullable = true)
|-- Musica: string (nullable = true)
|-- Password: string (nullable = true)
|-- Peso: integer (nullable = true)
|-- Sales_Social: string (nullable = true)
|-- Sexo: string (nullable = true)
|-- Tiempo_PlazaActual: integer (nullable = true)
|-- Tiempo_Vida_Laboral: integer (nullable = true)
|-- Tipo_Contrato: string (nullable = true)
|-- Tipo_Trabajo: string (nullable = true)
|-- Ultima_Encuesta_Cansancio_Emocional: integer (nullable = true)
|-- Ultima_Encuesta_Despersonalizacion: integer (nullable = true)
|-- Comentarios2: string (nullable = true)
|-- Ultima_Encuesta_Realizacion_Personal: integer (nullable = true)
|-- Viajas: integer (nullable = true)
|-- Horas_Cuidados: integer (nullable = true)
|-- Horas_Activ_Fisica: integer (nullable = true)
|-- Hora_Gratificante: integer (nullable = true)
|-- Hora_Social: integer (nullable = true)
|-- Contrato_Adjunto: string (nullable = true)
|-- Estado_Amino: string (nullable = true)
|-- Cansancio_Emocional: integer (nullable = true)
|-- Despersonalizacion: integer (nullable = true)
|-- Comentarios: string (nullable = true)
|-- Realizacion_Personal: integer (nullable = true)
|-- Burnout_Antes: string (nullable = true)
|-- Burnout_Despues: string (nullable = true)
|-- Calorias: double (nullable = true)
|-- Max_HeartRate: integer (nullable = true)
|-- Min_HeartRate: integer (nullable = true)
|-- Tipo_Actividad: string (nullable = true)
|-- Resting_HeartRate: integer (nullable = true)
|-- Frecuencia_Cardiaca_Minuto: integer (nullable = true)
|-- Duracion_Sueno: integer (nullable = true)
|-- Eficiencia_Sueno: integer (nullable = true)
|-- Niveles_deSueno: string (nullable = true)
|-- Cantidad_Sueno_Profundo: integer (nullable = true)
|-- Minutos_Sueno_Profundo: integer (nullable = true)
|-- Minutos_Sueno_Ligero: integer (nullable = true)
|-- Minutos_Rem: integer (nullable = true)
|-- Minutos_Sueno_Wake: integer (nullable = true)
|-- Minutos_Dormido: integer (nullable = true)
|-- Minutos_Despierto_enCama: integer (nullable = true)
|-- Minutos_paraDormir: integer (nullable = true)
|-- Tiempo_enCama: integer (nullable = true)
```

Ilustración 40: Esquema principal del Dataset

En el dataset hay 55 columnas de las cuales 33 son variables de tipo integer, 1 columna es de tipo float y 21 columnas son de tipo String.

Como se puede observar, se cuenta con una gran cantidad de datos, por lo que es necesario conocer cuáles son las características más importantes, si se quiere obtener una predicción certera. Es por esta razón que, a partir de este dataset se realizó el análisis de sus principales componentes.

5.2.2 Análisis de los principales componentes

El problema que se planteó hace referencia a la descripción de la información contenida en estos datos, mediante algún conjunto de variables menor que el conjunto de variables originales.

Es necesario determinar entonces, si hay variables en función de otras, ya que esto indicaría que hay información redundante.

Por tanto, si las p variables observadas están fuertemente correlacionadas, será posible sustituirlas por menos variables sin gran pérdida de “información”. (Aurea Grané)

Para ello, debido al gran volumen de los datos, se utilizó un servicio de Google Tables que arrojó con gran exactitud los principales componentes y su importancia. Dicho conjunto de datos, se guardan en un fichero CSV, a partir del cual se grafica y coloca como referencia en el sistema.

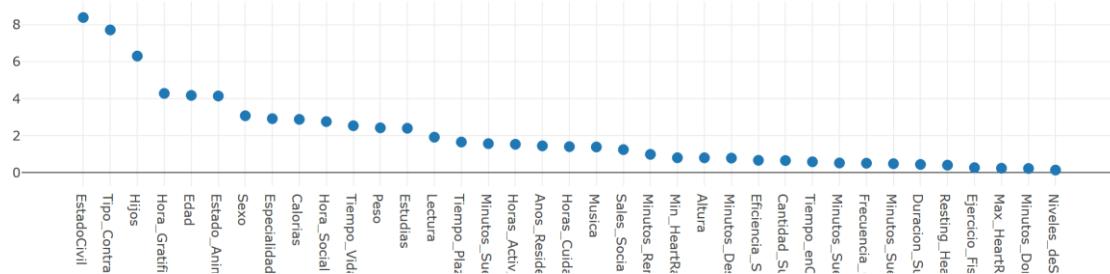


Ilustración 41:Principales Componentes

Debido a su inmensidad de columnas no se aprecian todas como leyenda, pero si se puede visualizar una representación de ellas gracias a los puntos.

Partiendo de este resultado, para posteriores análisis y para la construcción de los modelos, se ha decidido tomar las variables cuya importancia $p>1$. Estas son:

```
root
|-- Tiempo_PlazaActual: integer (nullable = true)
|-- EstadoCivil: string (nullable = true)
|-- Burnout_Antes: string (nullable = true)
|-- Hora_Social: integer (nullable = true)
|-- Horas_Cuidados: integer (nullable = true)
|-- Calorías: double (nullable = true)
|-- Peso: integer (nullable = true)
|-- Contrato_Adjunto: string (nullable = true)
|-- Musica: string (nullable = true)
|-- Sexo: string (nullable = true)
|-- Estudios: string (nullable = true)
|-- Sales_Social: string (nullable = true)
|-- Edad: string (nullable = true)
|-- Estado_Animo: string (nullable = true)
|-- Tiempo_Vida_Laboral: integer (nullable = true)
|-- Hijos: integer (nullable = true)
|-- Lectura: string (nullable = true)
|-- Hora_Gratificante: integer (nullable = true)
|-- Horas_Activ_Fisica: integer (nullable = true)
```

Ilustración 42: Variables para el estudio

Se pasa entonces de 55 variables a 19 fuertemente relacionadas con el objetivo.

5.3 Algoritmos Machine Learning aplicados

En este apartado se va a desarrollar y explicar cada uno de los algoritmos que se han utilizado, así como el proceso de preparación de los datos, para la ingesta por estos algoritmos, por ello vamos a seguir el siguiente análisis.

- Preparación de los datos
- Regresión Logística
- Bosques Aleatorios
- Árbol de Decisión

El tipo de clasificación que se va a realizar es una clasificación **Binaria**, debido a que el valor que puede tomar la variable **Burnout** es “VERDADERO” o “FALSO”.

La clasificación binaria es utilizada en análisis de fraude bancario, donde una transacción se intenta clasificar en fraudulenta o no; en correo fraudulento donde un correo puede ser catalogado como fraudulento o no.

5.3.1 Preparación de los datos

En este apartado se van a cargar en Spark el dataset “Burnout_Data.csv”, seleccionando las variables con importancia $p>1$ y luego se ejecutarán una serie de pasos para el tratamiento de las variables según su tipo.

```
spark = SparkSession.builder.appName('SistemaDeDetencion').master("local[*]").getOrCreate() #Creamos la sesión de spark
data = spark.read.csv("Burnout_Data.csv",header=True, inferSchema=True) #Cargamos el dataset
data = data.select('Tiempo_PlazaActual', 'EstadoCivil', 'Burnout_Antes', 'Hora_Social', 'Horas_Cuidados', 'Calorías', 'Peso', 'Contrato_Adjunto', 'Musica',
                   'Sexo', 'Estudios', 'Sales_Social', 'Edad', 'Estado_Animo', 'Tiempo_Vida_Laboral', 'Hijos', 'Lectura', 'Hora_Gratificante', 'Horas_Activ_Física')
#Nos quedamos con las columnas de importancia p>1 según el análisis de componentes
cols = data.columns #Guardamos en una variable los nombres de las columnas
```

Ilustración 43:Código de selección de variables según su importancia

5.3.1.1 Variables categóricas:

Para las variables categóricas se utiliza la herramienta de Spark “OneHotEncoderEstimator”. Dicha herramienta lo que hace es asignarle un valor numérico a los diferentes valores que contiene cada variable categórica, con el objetivo de añadirles valores en un vector numérico. Es necesario hacer este proceso, debido a que la mayoría de los algoritmos solo entienden valores numéricos y no caracteres.

“La codificación única asigna una característica de tipo categórica, representada como un índice de etiqueta, a un vector binario. Esta codificación permite que los algoritmos que esperan características continuas, como la regresión logística, utilicen características de tipo categóricas.” (Spark - <https://spark.apache.org/docs/latest/ml-features>)

Una vez procesadas las variables categóricas, se añade el vector resultante en una variable llamada “stages”, que no es más que una lista, con los pasos que más adelante ejecutara el Pipeline².

```
from pyspark.ml.feature import OneHotEncoderEstimator, StringIndexer, VectorAssembler #importamos las librerías necesarias para convertir datos categóricos
# en datos tratables por los algoritmos, es decir transformandolos a números
categoricalColumns = ['Contrato_Adjunto', 'Musica', 'Sexo', 'Estudios', 'Sales_Social', 'Edad', 'Estado_Animo', 'Lectura', 'EstadoCivil']
stages = [] #en esta variable guardaremos cada uno de los pasos para luego aplicarlos en el Pipeline
for categoricalCol in categoricalColumns: #indexamos para cada una de las variables categóricas de la lista
    stringIndexer = StringIndexer(inputCol = categoricalCol, outputCol = categoricalCol + 'Index')
    encoder = OneHotEncoderEstimator(inputCols=[stringIndexer.getOutputCol()], outputCols=[categoricalCol + "classVec"])
    #una vez indexadas utilizamos el OneHotEncoderEstimator que le asigna a cada valor de la variable categórica un número
    stages += [stringIndexer.setHandleInvalid("keep"), encoder]
#guardamos este proceso en la variable stages indicandole que si hay valores inválidos los conserve
```

Ilustración 44: Codificación de variables categóricas

5.3.1.2 Variables Numéricas

Para las variables numéricas se utiliza “VectorAssembler”, donde se toman las variables numéricas y el vector resultante de las variables categóricas, uniéndolos para obtener como resultado la columna “features”.

“VectorAssembler es un transformador que combina una lista dada de columnas en una sola columna vectorial, resulta de gran utilidad para combinar características sin

² Pipeline lo definimos como el conjunto de pasos ordenados que hay que dar para llegar al resultado.

procesar y generadas por diferentes transformadores en un solo vector de características, con el fin de entrenar modelos ML como la regresión logística y los árboles de decisión. VectorAssembler acepta los siguientes tipos de columnas de entrada: todos los tipos numéricos, tipo booleano y tipo vectorial. En cada fila, los valores de las columnas de entrada se concatenarán en un vector en el orden especificado.” (Spark - <https://spark.apache.org/docs/latest/ml-features>)

Para la variable que se está intentando predecir, se llevó a cabo un procedimiento aparte, para el cual se utilizó el “StringIndexer”, obteniendo como resultado la variable “label”.

“StringIndexer codifica una columna de cadena de etiquetas en una columna de índices de etiqueta.” (Spark - <https://spark.apache.org/docs/latest/ml-features>).

```
label_stringIdx = StringIndexer(inputCol="Burnout_Antes", outputCol="label")  
#Indexamos como label la variable que queremos predecir que es el Burnout_Antes cuyos valores  
#Son VERDADERO y FALSO  
stages += [label_stringIdx.setHandleInvalid("keep")]  
# Guardamos este proceso en la variable stages indicandole que si hay valores invalidos los conserve  
  
numericCols = ['Tiempo_PlazaActual', 'Hora_Social', 'Horas_Cuidados', 'Calorias', 'Peso', 'Tiempo_Vida_Laboral', 'Hijos', 'Hora_Gratificante', 'Horas_Activ_Fisica']  
#Con las variables categoricas transformadas a numeros podemos hacer un vector uniendo con las variables numericas.  
assemblerInputs = [c + "classVec" for c in categoricalColumns] + numericCols  
assembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features")  
#Este proceso nos da como resultado las "features" que contienen en objeto vector las variables numericas y categoricas.  
stages += [assembler.setHandleInvalid("keep")]  
# Guardamos este proceso en la variable stages indicandole que si hay valores invalidos los conserve
```

Ilustración 45: Codificación de variables numéricas

5.3.1.3 Pipeline

Estos pasos se fueron guardando en una lista llamada “stages” y ahora en el Pipeline se ejecuta cada uno de ellos, guardando el modelo resultante, debido a que luego se va a necesitar para poder hacer predicciones sobre nuevos datos.

“Un Pipeline se especifica como una secuencia de etapas y cada etapa es un transformador o un estimador. Estas etapas se ejecutan en orden, y el DataFrame de entrada se transforma a medida que pasa por cada etapa. Para la etapa de transformación, el método transform() se instancia en el DataFrame. Para las etapas del Estimador, se llama al método fit() para producir un Transformador (que se convierte en parte del PipelineModel o Pipeline ajustado) (Spark - <https://spark.apache.org/docs/latest/ml-pipeline.html>).

Ilustración 46: Pipeline

```
from pyspark.ml import Pipeline  
pipeline = Pipeline(stages = stages)  
#Inicializamos nuestro Pipeline y le pasamos la lista de pasos que debe ejecutar, que se encuentran en la variable stages.  
pipelineModel = pipeline.fit(data)  
data = pipelineModel.transform(data)  
#Ejecutamos y entreamos el modelo que seria el procesamiento de los datos.  
path = 'modelo_Pipeline'  
os.mkdir(path)  
pipelineModel.save(os.path.join(path, 'Pipeline'))  
#Guardamos este modelo, debido a que para predecir necesitamos aplicar este mismo modelo a los nuevos datos  
selectedCols = ['label', 'features'] + cols  
data = data.select(selectedCols)  
#Seleccionamos la variable label y features, mas la variable cols que contiene las columnas antes de hacer el procesado de datos
```

5.3.1.4 RandomSplit

Por último, en todas estas transformaciones que deben sufrir los datos, se utiliza el “RandomSplit”, con un porcentaje de 70% de entrenamiento y 30% de pruebas. A su vez, es importante mencionar, que el RandomSplit permite dividir el dataset de manera aleatoria, para su posterior uso en los diversos algoritmos de aprendizaje.

Una vez que se ha dividido, se realizará un retorno para que los algoritmos utilicen estos datos para su entrenamiento y pruebas.

```
train, test = data.randomSplit([0.7, 0.3])
#Para el entrenamiento y las pruebas utilizamos entonces un randomSplit para dividir el dataset en un porcentaje 70% entrenamiento y 30% pruebas
print("Training Dataset Count: " + str(train.count()))
print("Test Dataset Count: " + str(test.count()))
#imprimimos la cantidad de filas que tiene cada uno y devolvemos estos datos para su utilización por los algoritmos.
return train,test
```

Ilustración 47:Código de RandomSplit

5.3.1.5 Código completo preparación de los datos

```
def DataPreparation():
    spark = SparkSession.builder.appName('SistemaDeDetecction').master("local[*]").getOrCreate() #Creamos la sesión de spark
    data = spark.read.csv("Burnout_Data.csv",header=True, inferSchema=True) #Cargamos el dataset
    data = data.select('Tiempo_PlazaActual','EstadoCivil','Burnout_Antes','Hora_Social','Horas_Cuidados','Calorias','Peso','Contrato_Adjunto','Musica',
    'Sexo','Estudias','Sales_Social','Edad','Estado_Amino','Tiempo_Vida_Laboral','Hijos','Lectura','Hora_Gratificante','Horas_Activ_Fisica')
    #Nos quedamos con las columnas de importancia por según el análisis de componentes
    cols = data.columns #Guardamos en una variable los nombres de las columnas

    from pyspark.ml.feature import OneHotEncoderEstimator, StringIndexer, VectorAssembler #importamos las librerías necesarias para convertir datos categóricos
    #en datos tratables por los algoritmos, es decir transformándolos a números
    categoricalColumns = ['Contrato_Adjunto','Musica','Sexo','Estudias','Sales_Social','Edad','Estado_Amino','Lectura','EstadoCivil']
    stages = [] #en esta variable guardaremos cada uno de los pasos para luego aplicarlos en el Pipeline
    for categoricalCol in categoricalColumns: #indexamos para cada una de las variables categóricas en la lista
        stringIndexer = StringIndexer(inputCol = categoricalCol, outputCol = categoricalCol + "Index")
        encoder = OneHotEncoderEstimator(inputCols=[stringIndexer.getOutputCol()], outputCols=[categoricalCol + "classVec"])
        #una vez indexadas utilizamos el OneHotEncoderEstimator que le asigna a cada valor de la variable categórica un número
        stages += [stringIndexer.setHandleInvalid("keep"), encoder]
    #Guardamos este proceso en la variable stages indicandole que si hay valores inválidos los conserve

    label_stringIdx = StringIndexer(inputCol="Burnout_Antes", outputCol="label") #Indexamos como label la variable que queremos predecir que es el Burnout_Antes cuyos valores
    #son VERDADERO y FALSO
    stages += [label_stringIdx.setHandleInvalid("keep")]
    # Guardamos este proceso en la variable stages indicandole que si hay valores inválidos los conserve

    numericCols = ['Tiempo_PlazaActual','Hora_Social','Horas_Cuidados','Calorias','Peso','Tiempo_Vida_Laboral','Hijos','Hora_Gratificante','Horas_Activ_Fisica']
    #Con las variables categóricas transformadas a números podemos hacer un vector uniendo con las variables numéricas.
    assemblerInputs = [c + "classVec" for c in categoricalColumns] + numericCols
    assembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features")
    #este proceso nos da como resultado las "features" que contienen en objeto vector las variables numéricas y categóricas.
    stages += [assembler.setHandleInvalid("keep")]
    # Guardamos este proceso en la variable stages indicandole que si hay valores inválidos los conserve

    from pyspark.ml import Pipeline
    pipeline = Pipeline(stages = stages)
    #Inicializamos nuestro Pipeline y le pasamos la lista de pasos que debe ejecutar, que se encuentran en la variable stages.
    pipelineModel = pipeline.fit(data)
    data = pipelineModel.transform(data)
    #ejecutamos y entreamos el modelo que seria el procesamiento de los datos.
    path = "modelo_Pipeline"
    os.mkdir(path)
    pipelineModel.save(os.path.join(path, 'Pipeline'))
    #Guardamos este modelo, debido a que para predecir necesitamos aplicar este mismo modelo a los nuevos datos
    selectedCols = ['label', 'features'] + cols
    data = data.select(selectedCols)
    #Seleccionamos la variable label y features, más la variable cols que contiene las columnas antes de hacer el procesado de datos

    train, test = data.randomSplit([0.7, 0.3])
    #Para el entrenamiento y las pruebas utilizamos entonces un randomSplit para dividir el dataset en un porcentaje 70% entrenamiento y 30% pruebas
    print("Training Dataset Count: " + str(train.count()))
    print("Test Dataset Count: " + str(test.count()))
    #imprimimos la cantidad de filas que tiene cada uno y devolvemos estos datos para su utilización por los algoritmos.
    return train,test
```

Ilustración 48:Código de preparación de la data.

5.3.2 CrossValidation

En este apartado se hará una descripción de lo que es un Cross validation y de sus principales beneficios, debido a que este ha sido utilizado en todos los algoritmos entrenados.

Cross validation o validación cruzada en español, es una técnica que se utiliza para mejorar la capacidad que tienen los algoritmos de generar modelos; la validación cruzada intenta reducir el ruido de un determinado modelo, haciendo una división más exhaustiva de los datos con los que se va a entrenar.

La validación cruzada que se ha utilizado para todos los modelos es la denominada “**K-Fold**”, esta técnica se utiliza cuando se cuentan con suficientes datos para entrenar el modelo, dejando datos amplios tanto para entrenarlos como para probarlos.

En la técnica K-Fold, los datos se dividen en K subconjuntos, ahora el método se ejecuta k veces, es decir, que cada vez se utiliza un subconjunto k diferente de entrenamiento y de prueba.

La estimación del error se hace sacando la media de todos los k para entonces poder obtener la efectividad del modelo, pudiéndose visualizar entonces, que cada conjunto tiene datos únicos, lo que reduce significativamente el sesgo, debido a que se utilizan la mayoría de los datos para construir el modelo y se reduce también la varianza. Esto pasa de la misma forma para el conjunto de validación, puesto a que se utilizan la mayoría de datos para validar el modelo.

Como regla general se suele situar la k entre 5 y 10.

5.3.3 Algoritmo Regresión Logística

El algoritmo Regresión Logística es uno de los métodos más populares de clasificación, ya que combina cada una de las entradas individuales conocidas como “features”, con un peso específico que se genera automáticamente en el proceso de entrenamiento. Dichos pesos se combinan para conseguir la mejor configuración posible para la variable que se quiere predecir.

Estos pesos no son más que la importancia de cada una de las características, lo que, significa entonces que, si una característica tiene un peso elevado, su variación afecta en gran medida al modelo. Sin embargo, en el caso de que su importancia sea baja, su variación no afectaría demasiado en el modelo.

La ecuación de este algoritmo es la siguiente:

$$L(\mathbf{w}; \mathbf{x}, y) := \log(1 + \exp(-y\mathbf{w}^T \mathbf{x})).$$

Ilustración 49:Ecuación Regresión Logística

La ecuación para conseguir una predicción es:

$$f(z) = \frac{1}{1 + e^{-z}}$$

Ilustración 50: Ecuación de Regresión Logística predicción

Donde $Z = W^T x$, por defecto si $Z > 0.5$ el resultado es positivo o negativo, dependiendo de cómo se configure, obteniendo una probabilidad con el resultado.

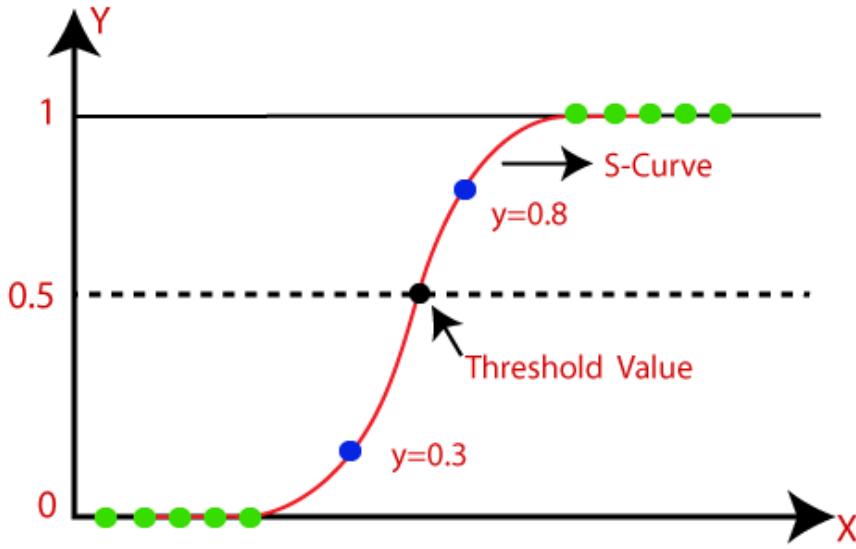


Ilustración 51: Regresión Logística (<https://www.javatpoint.com/logistic-regression-in-machine-learning>)

Esta ecuación lo que intenta es conseguir una línea curva que pase por la mayor cantidad de puntos, para lograr describir el problema y que, al predecir el algoritmo, se coloque el dato que se está insertando en algún punto de esa curva que creó, dependiendo de qué tan cerca quede de una característica u otra, el algoritmo devuelve la pertenencia de ese dato.

La aplicación en Spark realizada pertenece al siguiente código:

```
def LogisticRegression(train,test):
    from pyspark.ml.classification import LogisticRegression #importamos la librería necesaria
    lr = LogisticRegression(featuresCol = 'features', labelCol = 'label', maxIter=100, elasticNetParam=0.5, fitIntercept=False, threshold=0.8) #establecemos parametros para el entrenamiento
    lrModel = lr.fit(train) #entrenamos
    predictions = lrModel.transform(test)
    predictions.select('Burnout_Ants', 'label', 'rawPrediction', 'prediction', 'probability').show(10) #mostramos algunas predicciones
    from pyspark.ml.evaluation import BinaryClassificationEvaluator #importamos el evaluador binario para evaluar nuestro modelo
    evaluator = BinaryClassificationEvaluator()
    print('Test Area Under ROC', evaluator.evaluate(predictions)) #evaluamos e imprimimos el valor ROC

    from pyspark.ml.tuning import ParamGridBuilder, CrossValidator #importamos el CrossValidation
    paramGrid = (ParamGridBuilder()
        .addGrid(lr.maxIter, [100, 500, 1000, 5000, 7000, 10000]) #seleccionamos otro conjunto de características
        .addGrid(lr.elasticNetParam, [0,0.2,0.3,0.4,0.5,0.7,0.8])
        .addGrid(lr.fitIntercept, [True, False])
        .addGrid(lr.threshold, [1,0.9,0.8]))
    .build())
    cv = CrossValidator(estimator=lr, estimatorParamMaps=paramGrid, evaluator=evaluator, numFolds=10) #ejecutamos el crossvalidator donde se van a entrenar varios modelos
    cvModel = cv.fit(train) #de logistic Regression y el se quedara con el que mejor probabilidades tenga
    path = 'modelo_LogisticRegression' #guardamos el modelo
    os.mkdir(path)
    cvModel.save(os.path.join(path, 'modelLogisticRegression'))
    predictions = cvModel.transform(test)
    print('Test Area Under ROC', evaluator.evaluate(predictions)) #probamos nuevamente el ROC
```

Ilustración 52: Código Regresión Logística

En el código se siguen una serie de pasos para conseguir entrenar el modelo, con unos parámetros que se detallarán a continuación:

- featuresCol: es donde se le debe indicar al algoritmo cuál es el nombre de la columna y en donde se encuentran las features. Asimismo, se debe recordar, que al algoritmo se le pasan las variables “train” y “test” que han sido previamente tratadas como se ha descrito en los pasos anteriores.
- labelCol: Se indica la columna de la variable que se quiere predecir.
- maxIter: cuyo parámetro se ha fijado en 100 e indica la cantidad máxima de interacciones que va a tener el algoritmo antes de finalizar.
- elasticNetParam: este es un parámetro que va de 0 a 1, indicando la cantidad de dispersión que se quiere alcanzar en el modelo, por lo tanto, el valor 0 haría que el peso de las características sea arrastrado hacia el 0 pero nunca llegarían a 0, es decir, no crearía dispersión y un valor de 1 haría lo contrario, es decir, crear dispersión, por lo cual es un valor que conviene ir cambiando para evaluar resultados.
- fitIntercept: este parámetro puede ser “true” o “false” y lo que hace es interceptar los datos añadiendo un valor aleatorio a los pesos de las características; normalmente se utiliza en “true” cuando no se han normalizado los datos.
- Threshold: este parámetro va de 0 a 1 y lo que hace es intentar balancear entre los falsos positivos o falsos negativos, por lo que hacer una predicción errónea implica un alto costo en el modelo, por lo cual se debe colocar un número alto.

Luego se ejecuta la validación cruzada, con $k = 10$ y esta técnica va a probar qué modelo tiene mejores posibilidades con cada uno de los siguientes parámetros:

- maxIter: va a ejecutar pruebas con este parámetro en los siguientes valores [100, 500, 1000, 5000, 7000, 10000]
- elasticNetParam: va a ejecutar pruebas con este parámetro en los siguientes valores [0,0.2,0.3,0.4,0.5,0.7,0.8].
- fitIntercept: va a ejecutar pruebas con este parámetro en los siguientes valores: [True, False].
- Threshold: va a ejecutar pruebas con este parámetro en los siguientes valores: [1,0.9,0.8].

Luego de que se hagan todas esas combinaciones, se guardará el modelo que la validación cruzada consiga mejor.

Se puede conseguir más información sobre la ejecución en el apartado 3.4.4 del Capítulo 3.

Para la evaluación de cada algoritmo se cargan los datos de los pacientes y se ejecutan los algoritmos, para ver las estadísticas y evaluar más de cerca la efectividad.

Para el algoritmo de Regresión Logística, los resultados son los siguientes: (recordando que **1 es falso** para síndrome de burnout, es decir que no tiene burnout, y **0 es verdadero** para síndrome de burnout es decir que si tiene el síndrome).

Email	Identificador	Burnout_Antes	prediction	probability
uemusuari01@gmail.com 1	1	0.0	[NaN,NaN,NaN]	
uemusuari08@gmail.com 11	1	1.0	[8.813734910715095E-7, 0.999999118626509, 3.2136909342597626E-22]	
uemusuari07@gmail.com 13	1	0.0	[NaN,NaN,NaN]	
uemusuari04@gmail.com 21	0	0.0	[NaN,NaN,NaN]	
uemusuari05@gmail.com 22	0	0.0	[0.9999999999978235, 2.176573688219233E-12, 6.1877899152379645E-27]	
uemusuari06@gmail.com 25	0	0.0	[0.999999999543931, 4.5606899571922716E-11, 7.072745716079234E-32]	
mti.jgaytan@gmail.com 26	1	0.0	[NaN,NaN,NaN]	
uemusuari04@gmail.com 28	1	1.0	[1.057389553713662E-5, 0.999989426104463, 9.06849967057413E-21]	
uemusuari02@gmail.com 39	1	1.0	[6.245207204886128E-9, 0.999999937547928, 3.8775421369470856E-23]	
uemusuari03@gmail.com 40	0	0.0	[0.99999999985471, 1.4529030374329588E-11, 2.015362197122182E-27]	
uemusuari09@gmail.com 42	1	1.0	[3.960941713322857E-6, 0.9999960390582866, 9.031154902420332E-25]	
uemusuari013@gmail.com 43	1	1.0	[9.508864642809788E-6, 0.9999904911353571, 4.568902699224657E-24]	
uemusuari011@gmail.com 44	1	0.0	[NaN,NaN,NaN]	
uemusuari015@gmail.com 47	0	0.0	[0.999999999992306, 7.693844385882997E-13, 2.771268183669807E-26]	
sukuzhanay@gmail.com 58	0	0.0	[NaN,NaN,NaN]	

Ilustración 53: Predicciones Regresión Logística

Se observa que, para un total de 16 casos, el algoritmo de regresión logística ha acertado 12 casos por lo que su eficacia es del 75% de los casos.

5.3.4 Algoritmo Árbol de Decisión

El algoritmo decision tree o árboles de decisión, se utiliza ampliamente debido a su fácil interpretación; su facilidad de utilizar variables de todo tipo, tanto numéricas como categóricas, no requiere de escalas de características y es capaz de identificar datos de una manera no lineal.

Este conjunto de algoritmos que corresponde a los bosques aleatorios, son unos de los algoritmos con mejores resultados para generar modelos de clasificación o regresión.

Los árboles de decisión aprenden de los datos para intentar aproximarlos a una curva sinusoidal con un conjunto de reglas de decisión, que van creando un árbol, por lo cual, cuanto más profundo es el árbol, más complejas son estas reglas de decisión y más se ajusta al modelo.

El algoritmo crea una estructura de árbol desglosando un conjunto de datos en subconjuntos cada vez más pequeños, mientras que al mismo tiempo se desarrolla un árbol de decisión asociado. (<https://spark.apache.org/docs/1.5.2/ml-decision-tree.html>)

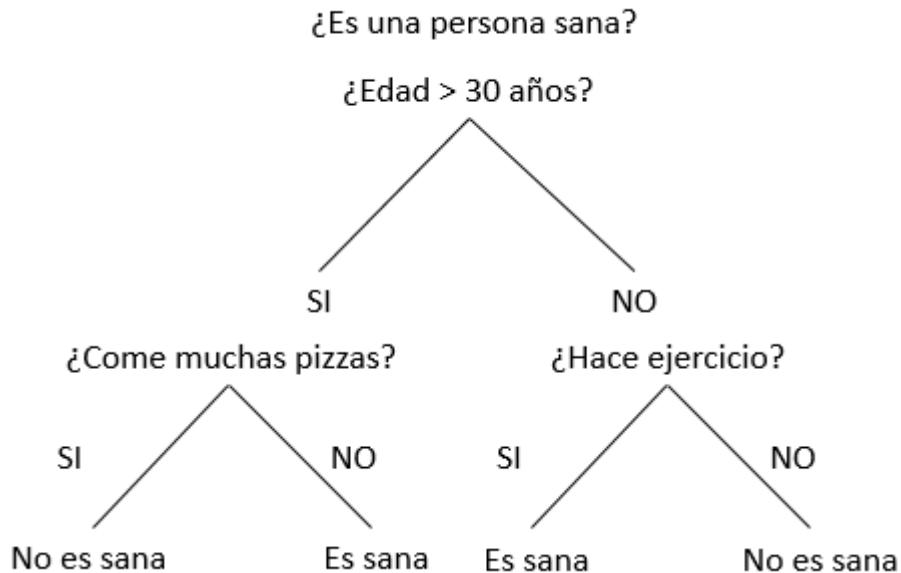


Ilustración 54: Árbol de decisión

La aplicación en Spark realizada pertenece al siguiente código:

```

def DecisionTree(train,test):
    from pyspark.ml.classification import DecisionTreeClassifier #importamos la libreria necesaria
    modelTree = DecisionTreeClassifier(featuresCol="features", labelCol='label',maxDepth=30,minInfoGain=0.4,maxBins=10) #establecemos los parametros para el entrenamiento
    TreeModel = modelTree.fit(train)#entrenamos
    predictions = TreeModel.transform(test)
    predictions.select('Burnout_Antes', 'label', 'rawPrediction', 'prediction', 'probability').show(10) #mostramos algunas predicciones
    from pyspark.ml.evaluation import BinaryClassificationEvaluator#importamos el evaluador binario para evaluar nuestro modelo
    evaluator = BinaryClassificationEvaluator()
    print('Test Area Under ROC', evaluator.evaluate(predictions))

    from pyspark.ml.tuning import ParamGridBuilder, CrossValidator #importamos el crossvalidator
    paramGrid = (ParamGridBuilder()
        .addGrid(modelTree.maxDepth, [5, 15, 30]) #seleccionamos otro conjunto de caracteristicas
        .addGrid(modelTree.minInfoGain, [0,0.4,0.8])
        .addGrid(modelTree.maxBins, [18, 20,10,5,2])
        .build())
    cv = CrossValidator(estimator=modelTree, estimatorParamMaps=paramGrid, evaluator=evaluator, numFolds=10) #ejecutamos el crossvalidation donde se van a entrenar varios modelos
    #de random forest y el se quedara con el que mejor probabilidades tenga
    cvModel = cv.fit(train)
    path = 'modelo_DecisionTree' #guardamos el modelo
    os.makedirs(path)
    cvModel.save(os.path.join(path, 'modelDecisionTree'))
    predictions = cvModel.transform(test)
    print('Test Area Under ROC', evaluator.evaluate(predictions)) #probamos nuevamente el ROC
  
```

Ilustración 55: Código del árbol de decisión

En el código se siguen una serie de pasos para conseguir entrenar al modelo, lo primero es establecer los parámetros de entrada:

- featuresCol: es donde se le debe indicar al algoritmo cuál es el nombre de la columna donde se encuentran las features, recordando que al algoritmo se le pasan las variables “train” y “test”, que han sido previamente tratadas como describimos en los pasos anteriores.
- labelCol: Se indica la columna de la variable que se quiere predecir.
- maxDepth: este parámetro ayuda a especificar el máximo de profundidad que se quiere que alcance el modelo, el valor por defecto es 5.

- minInfoGain: este parámetro define la cantidad de información que puede ser usada para para una nueva rama del árbol, un valor alto puede prevenir que el modelo tenga sobreajuste.
- maxBins: este parámetro define la cantidad de características que se van a utilizar en cada rama del árbol. El valor por defecto es 32.

Luego, se ejecuta la validación cruzada, con $k = 10$ y esta técnica va a probar que modelo tiene mejores posibilidades con cada uno de los siguientes parámetros:

- maxDepth: va a ejecutar pruebas con este parámetro en los siguientes valores: [5, 15, 30].
- minInfoGain: va a ejecutar pruebas con este parámetro en los siguientes valores: [0,0.4,0.8]
- maxBins: va a ejecutar pruebas con este parámetro en los siguientes valores: [18, 20,10,5,2]

Luego de que se hagan todas esas combinaciones, se debe guardar el modelo que la validación cruzada consiga mejor.

Se puede conseguir más información sobre la ejecución en el apartado 3.4.4 del Capítulo 3.

Para la evaluación de cada algoritmo, se cargan los datos de los pacientes y se ejecutan en cada uno de los algoritmos para ver las estadísticas y poder evaluar más de cerca la efectividad.

Para el algoritmo de Árbol de Decisión, los resultados son los siguientes: (recordando que **1 es falso** para síndrome de burnout, es decir que no tiene burnout, y **0 es verdadero** para síndrome de burnout es decir que si tiene el síndrome).

Email	Identificador	Burnout_Antes	prediction	probability
uemusuario1@gmail.com	1	1	0.0	[[1.0,0.0,0.0]]
uemusuario8@gmail.com	11	1	1.0	[[0.0,1.0,0.0]]
uemusuario7@gmail.com	13	1	1.0	[[0.0,1.0,0.0]]
uemusuario14@gmail.com	21	0	0.0	[[1.0,0.0,0.0]]
uemusuario5@gmail.com	22	0	0.0	[[1.0,0.0,0.0]]
uemusuario6@gmail.com	25	0	0.0	[[1.0,0.0,0.0]]
lmti.jgaytan@gmail.com	26	1	0.0	[[1.0,0.0,0.0]]
uemusuario4@gmail.com	28	1	1.0	[[0.0,1.0,0.0]]
uemusuario12@gmail.com	39	1	1.0	[[0.0,1.0,0.0]]
uemusuario3@gmail.com	40	0	0.0	[[1.0,0.0,0.0]]
uemusuario9@gmail.com	42	1	1.0	[[0.0,1.0,0.0]]
uemusuario13@gmail.com	43	1	1.0	[[0.0,1.0,0.0]]
uemusuario11@gmail.com	44	1	0.0	[[1.0,0.0,0.0]]
uemusuario15@gmail.com	47	0	0.0	[[1.0,0.0,0.0]]
sukuzhanay@gmail.com	58	0	0.0	[[1.0,0.0,0.0]]

Ilustración 56: Predicciones Árbol de Decisión

Se puede observar que, para un total de 16 casos, el algoritmo de decision tree ha acertado 13 casos por lo que su eficacia es del 81.25% de los casos.

5.3.5 Algoritmo Bosques Aleatorios

El algoritmo Bosques Aleatorios o bosque aleatorio, consiste en una gran cantidad de árboles de decisión individuales que operan en grupo, donde cada árbol de decisión devuelve una clase y la clase que más se haya repetido, se convierte en la predicción del modelo.

El concepto fundamental detrás de este algoritmo es conocido como “la sabiduría de las multitudes”. La razón por la que este modelo funciona tan bien se debe a que “un gran número de modelos (árboles) relativamente no correlacionados que operan como comité superará a cualquiera de los modelos constituyentes individuales”. (<https://towardsdatascience.com/understanding-random-forest-58381e0602d2>).

La aplicación en Spark realizada pertenece al siguiente código:

```
def RandomForest(train,test):
    from pyspark.ml.classification import RandomForestClassifier #importamos la libreria necesaria
    rf = RandomForestClassifier(featuresCol='features', labelCol='label',numTrees=500,featureSubsetStrategy="all") #establecemos los parametros para el entrenamiento
    rfModel = rf.fit(train) #entrenamos
    predictions = rfModel.transform(test)
    predictions.select('Burnout_Antes', 'label', 'rawPrediction', 'prediction', 'probability').show(10) #mostramos algunas predicciones

    from pyspark.ml.evaluation import BinaryClassificationEvaluator #importamos el evaluador binario para evaluar nuestro modelo
    evaluator = BinaryClassificationEvaluator()
    print('Test Area Under ROC:', evaluator.evaluate(predictions))
    from pyspark.ml.tuning import ParamGridBuilder, CrossValidator #importamos el Crossvalidator
    paramGrid = (ParamGridBuilder()
        .addGrid(rf.numTrees, [1000,2500,6000,10000]) #seleccionamos otro conjunto de caracteristicas
        .addGrid(rf.featureSubsetStrategy, ["all","auto", "sqrt","log2"])
        .build())
    cv = CrossValidator(estimator=rf, estimatorParamMaps=paramGrid, evaluator=evaluator, numFolds=10) #ejecutamos el crossvalidation donde se van a entrenar varios modelos
    #de random forest y el se quedara con el que mejor probabilidades tenga
    cvModel = cv.fit(train)
    path = 'modelo_RandomForest' #guardamos el modelo
    os.mkdir(path)
    cvModel.save(os.path.join(path, 'modelRandomForest'))
    predictions = cvModel.transform(test)
    print('Test Area Under ROC:', evaluator.evaluate(predictions)) #probamos nuevamente el ROC
```

Ilustración 57: Código Bosques Aleatorios

En el código se llevan a cabo una serie de pasos para conseguir entrenar el modelo, lo primero es establecer los parámetros de entrada:

- featuresCol: es donde se le debe indicar al algoritmo cuál es el nombre de la columna donde se encuentran las features, recordando que al algoritmo se le pasan las variables “train” y “test”, que han sido previamente tratadas como se ha descrito en los pasos anteriores.
- labelCol: Se indica la columna de la variable que se quiere predecir.
- numTrees: este parámetro indica la cantidad de árboles que se van a entrenar.
- featureSubsetStrategy: este parámetro determina cuantas características deben ser consideradas para hacer la división entre ramas de los árboles. Esto puede oscilar entre los siguientes valores: “auto”, “all”, “sqrt”, “log2” o un número n que se determine.
 - Auto: va a determinar este valor automáticamente
 - All: va a tomar todas las características
 - Sqrt: va a tomar el resultado de la raíz cuadrada del total de características disponibles

- Log2: va a tomar el logaritmo neperiano del total de características disponibles.

Posterior a esto, se ejecuta la validación cruzada, con $k = 10$ y esta técnica va a probar que modelo tiene mejores posibilidades con cada uno de los siguientes parámetros:

- numTrees: va a ejecutar pruebas con este parámetro en los siguientes valores: [1000,2500,6000,10000]
- featureSubsetStrategy: va a ejecutar pruebas con este parámetro en los siguientes valores: ["all", "auto", "sqrt", "log2"]

Luego de que se hagan todas esas combinaciones, se guarda el modelo que la validación cruzada consiga mejor.

Se puede conseguir más información sobre la ejecución en el apartado 3.4.4 del Capítulo 3.

Para la evaluación de cada algoritmo, se cargan los datos de los pacientes y se ejecutan en cada uno de los algoritmos, para ver las estadísticas y poder evaluar más de cerca la efectividad.

Para el algoritmo de Bosques Aleatorios, los resultados son los siguientes: (recordando que **1 es falso** para síndrome de burnout, es decir que no tiene burnout, y **0 es verdadero** para síndrome de burnout es decir que si tiene el síndrome).

Email	Identificador	Burnout_Antes	prediction	probability
uemusuario1@gmail.com 1	1	1.0	[0.26643161467567716, 0.7335683853243229, 0.0]	
uemusuario8@gmail.com 11	1	1.0	[0.028900683019028647, 0.9710993169809714, 0.0]	
uemusuario7@gmail.com 13	1	1.0	[0.06123973899248952, 0.9387602610075105, 0.0]	
uemusuario14@gmail.com 21	0	0.0	[0.9235310381261382, 0.07646896187386185, 0.0]	
uemusuario5@gmail.com 22	0	0.0	[0.9992260859212684, 7.73914078731596E-4, 0.0]	
uemusuario6@gmail.com 25	0	0.0	[0.9810975971272763, 0.01890240287272368, 0.0]	
mti.jgaytan@gmail.com 26	1	1.0	[0.2966537231921454, 0.7033462768078547, 0.0]	
uemusuario4@gmail.com 28	1	1.0	[0.08256093046652167, 0.9174390695334784, 0.0]	
uemusuario12@gmail.com 39	1	1.0	[0.016730916514988863, 0.9832690834850111, 0.0]	
uemusuario3@gmail.com 40	0	0.0	[0.9853014911781172, 0.014698508821882896, 0.0]	
uemusuario9@gmail.com 42	1	1.0	[0.03655740131302415, 0.9634425986869759, 0.0]	
uemusuario13@gmail.com 43	1	1.0	[0.043553061831566035, 0.956446938168434, 0.0]	
uemusuario11@gmail.com 44	1	1.0	[0.32214098919162854, 0.6778590108083715, 0.0]	
uemusuario15@gmail.com 47	0	0.0	[0.9996073455168841, 3.9265448311591233E-4, 0.0]	
sukuzhanay@gmail.com 58	0	0.0	[0.8236770852948058, 0.1763229147051942, 0.0]	

Ilustración 58: Predicciones Bosques Aleatorios

Se puede visualizar que, para un total de 16 casos, el algoritmo de Bosques Aleatorios ha acertado 16 casos por lo que su eficacia es del 100% de los casos.

5.3.6 Conclusión

Luego de observar el comportamiento de los diferentes algoritmos que se han entrenado, para construir un modelo capaz de predecir el síndrome del burnout, mediante la relación de las variables fisiológicas y de sueño, con otras variables como (datos personales, datos laborales, hábitos sociales y de tiempo de ocio).

Se puede afirmar que se ha logrado el objetivo y que el algoritmo con mayor precisión probado es el Bosques Aleatorios. Por este motivo, es que, para el formulario de detección del síndrome de burnout, se va a desplegar el algoritmo Bosques Aleatorios para predecir nuevos casos.

Capítulo 6. Desarrollo de una Herramienta Propia de Visualización

En el presente capítulo se detallará la herramienta de visualización para el sistema de burnout, el cual es un sistema web que se encuentra disponible en el siguiente enlace <http://burnoutweb.ddns.net/>; donde se puede visualizar una página principal con información básica sobre el síndrome de burnout, así como una clasificación principal de los datos fisiológicos, en función de las variables “verdadero” o “falso” del síndrome.

6.1.1 Importancia de la Herramienta

La herramienta de visualización es primordial para lograr el estudio y la comprensión de los datos. El objetivo principal es darles a los expertos sanitarios la capacidad y el control total sobre los datos que se manejan en este estudio, lo que resulta de suma importancia, debido a que les ayuda desde la comodidad de cualquier dispositivo electrónico, visualizar, descargar y probar el sistema de detección construido con los datos que se visualizan.

La **visualización de los datos** es de suma importancia en la actualidad, ya que los mismos crecen a ritmos agigantados. El emplear un modo visual que presente la información incluso cuando el volumen de datos es bastante considerable, permite **ahorrar tiempo y costes**, debido a que la información se puede visualizar de forma **clara, rápida y sencilla**, permitiendo la toma de buenas decisiones. (**Abraham Requena Mesa, 2017**).

Se pretende entonces con este gran hito, lograr un sistema completo desde el proceso ETL (extracción, transformación y carga), darles sentido a los datos y visualizarlos.

Una herramienta de visualización es información, por lo cual se prefiere construir esta herramienta para dar la posibilidad a un experto de analizar y sacar conclusiones avanzadas, antes de aventurarse profundamente en sacar conclusiones de áreas fuera de la especialidad.

6.1.2 Componentes utilizados

Esta herramienta está siendo ejecutada en la dirección comentada anteriormente, fue desarrollada utilizando Python 3.7 (Anaconda) y la librería Python Dash 1.7.0.

6.1.3 Datos alojados en Big Query

Se pueden observar el conjunto de datos utilizados para la herramienta de visualización, los cuales son iguales a los empleados para hacer el sistema de detección y su obtención se destaca en el Capítulo 4.

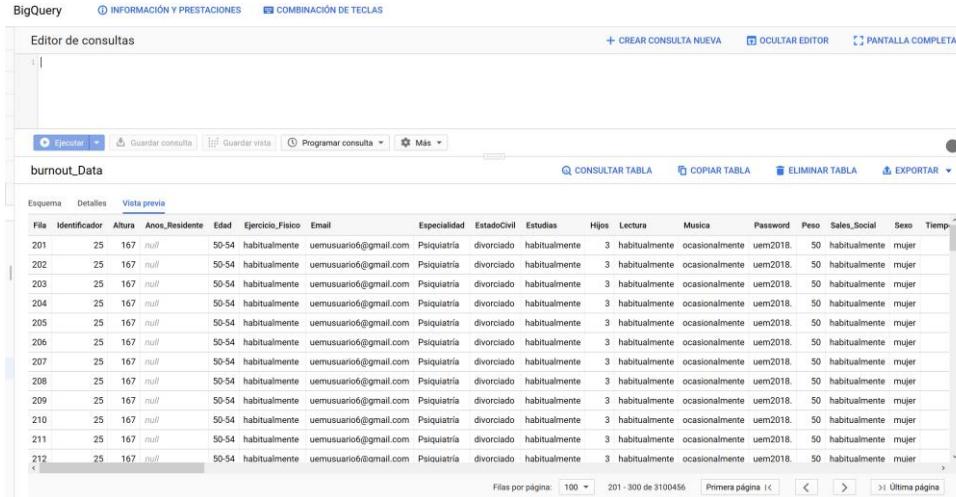


Ilustración 59: Datos alojados en Big Query

6.1.4 Código de la herramienta de visualización

En este apartado se mostrará el código desarrollado para el correcto funcionamiento de la herramienta.

Se visualizan algunas de las querys que se deben ejecutar para obtener los datos ordenados correspondientes a cada una de las subcategorías y por subsecuentes las gráficas. Es por esta razón que, se ha decidido utilizar la función AVG() de SQL, que corresponde a la media aritmética.

```
queryDatosFisiologicos = ('SELECT Burnout_Antes as Burnout, avg(Calorias) as Calorias, avg(Frecuencia_Cardiacas_Minuto) as Frecuencia_Cardiacas_Minuto ,'
    ',avg(Resting_HeartRate) as Resting_HeartRate,avg(Eficiencia_Sueno) as Eficiencia_Sueno, avg(Peso) as Peso,avg(Hijos) '
    ',as Hijos,avg(Tiempo_PlazaActual) as Tiempo_Plaza,avg( Tiempo_Vida_Laboral) as Tiempo_Vida_Laboral,avg(Max_HeartRate)as '
    'Max_HeartRate,avg( Min_HeartRate) as Min_HeartRate,avg( Duracion_Sueno ) as Duracion_Sueno,avg( Minutos_Rem ) as Minutos_Rem '
    ',avg( Cantidad_Sueno_Profundos) as SuenoProfundo,avg( Minutos_Sueno_Profundos) as Min_SuenoProfundo,avg( Minutos_Sueno_Ligero ) '
    ',as Min_SuenoLigero, avg( Minutos_Sueno_Wake) as Minutos_SuenoDespierto, avg( Minutos_Dormido) as Min_Dormido, '
    ', avg( Minutos_Despierto_enCama) Min_Despierto_enCama, avg( Tiempo_enCama) as Tiempo_enCama '
    'FROM `burnout-258011.burnout.burnout_Data` group by Burnout_Antes')
df_DatosFisiologicos = pd.read_gbq(queryDatosFisiologicos, project_id=project, dialect='standard')
df_DatosFisiologicos[["Burnout"]].replace({True: "VERDADERO", False: "FALSO"}, inplace=True)

queryPCA = ('SELECT Caracteristicas, (Importancia*100)as Importancia FROM `burnout-258011.burnout.Caracteristicas` order by Importancia DESC')
df_PCA= pd.read_gbq(queryPCA, project_id=project, dialect='standard')

queryDatosFisiologicosIndividuales = ('SELECT Email,avg(Calorias) as Calorias, '
    ',avg(Frecuencia_Cardiacas_Minuto) as Frecuencia_Cardiacas_Minuto ,'
    ',avg(Resting_HeartRate) as Resting_HeartRate,avg(Eficiencia_Sueno) as '
    'Eficiencia_Sueno, avg(Peso) as Peso,avg(Hijos),avg(Tiempo_PlazaActual) as Tiempo_Plaza,'
    ',avg( Tiempo_Vida_Laboral) as Tiempo_Vida_Laboral,avg(Max_HeartRate)as Max_HeartRate,avg( Min_HeartRate) '
    ',as Min_HeartRate,avg( Minutos_Rem ) as Minutos_Rem ,avg( Cantidad_Sueno_Profundos) as SuenoProfundo, '
    ',avg( Minutos_Sueno_Profundos) as Min_SuenoProfundo,avg( Minutos_Sueno_Ligero ) as Min_SuenoLigero, avg( Minutos_Sueno_Wake ) '
    ',as Minutos_SuenoDespierto, avg( Minutos_Dormido) as Min_Dormido, avg( Minutos_Despierto_enCama) Min_Despierto_enCama, '
    ',avg( Tiempo_enCama) as Tiempo_enCama, avg( Cansancio_Emocional) as Cansancio_Emocional,avg( Despersonalizacion ) '
    ',as Despersonalizacion, avg( Realizacion_Personal) as Realizacion_Personal, avg( Altura) as Altura,avg( Tiempo_PlazaActual) '
    ',as Tiempo_PlazaActual, avg(Ultima_Encuesta_Cansancio_Emocional) as Ultima_Encuesta_Cansancio_Emocional, '
    ',avg( Ultima_Encuesta_Despersonalizacion) as Ultima_Encuesta_Despersonalizacion, avg( Ultima_Encuesta_Realizacion_Personal) '
    ',as Ultima_Encuesta_Realizacion_Personal, avg( Horas_Cuidados) as Horas_Cuidados,avg( Horas_Activ_Fisica) as '
    ',Horas_Activ_Fisica, avg( Hora_Gratificante) as Hora_Gratificante, avg( Hora_Social) as Hora_Social FROM '
    '`burnout-258011.burnout.burnout_Data` group by Email')
df_DatosFisiologicosIndividuales = pd.read_gbq(queryDatosFisiologicosIndividuales, project_id=project, dialect='standard')

querySubescalasBurnout = ('SELECT avg(Cansancio_Emocional) as Cansancio_Emocional , avg(Depersonalizacion) as Despersonalizacion , '
    ',avg(Realizacion_Personal) as Realizacion_Personal, avg(Ultima_Encuesta_Realizacion_Personal) as '
    'Ultima_Encuesta_Realizacion_Personal, avg(Ultima_Encuesta_Desperpersonalizacion) as Ultima_Encuesta_Desperpersonalizacion , '
    ',avg(Ultima_Encuesta_Cansancio_Emocional) as Ultima_Encuesta_Cansancio_Emocional,Burnout_Antes,Burnout_Despues FROM '
    '`burnout-258011.burnout.burnout_Data` group by Burnout_Antes, Burnout_Despues')
df_SubescalasBurnout = pd.read_gbq(querySubescalasBurnout, project_id=project, dialect='standard')
df_SubescalasBurnout[["Burnout_Antes"]].replace({True: "VERDADERO", False: "FALSO"}, inplace=True)
df_SubescalasBurnout[["Burnout_Despues"]].replace({True: "VERDADERO", False: "FALSO"}, inplace=True)
```

Ilustración 60: Querys ejecutadas para las gráficas

En la siguiente ilustración es posible observar el procedimiento estándar para construir cada uno de los gráficos que se mostrarán en la herramienta.

```
158 dff = query_df_subescalaburnout_if_rows_is_None else pd.DataFrame(rows)
159
160 colors = ['#FF00FF' if i in derived_virtual_selected_rows else '#0074C0'
161     for i in range(len(dff))]
```

return [html.Br(),
164 html.H3("Es un gráfico dinámico!"),
165 html.P("Síntete libre de hacer zoom, seleccionar variables, filtrar en la tabla superior."),
166 html.Br(),
167 html.H3("Primera Encuesta de Burnout"),
168 html.P("En esta primera gráfica observamos los datos recogidos en la primera encuesta"),
169
170 dcc.Graph(
171 id='ii',
172 figure={
173 "data": [
174 {
175 "x": dff["Burnout_Antes"],
176 "y": dff["Cansancio_Emocional"],
177 "type": "bar",
178 "marker": {"color": colors},
179 "name": "Cansancio_Emocional",
180 },
181 {
182 "x": dff["Burnout_Antes"],
183 "y": dff["Despersonalizacion"],
184 "type": "bar",
185 "marker": {"color": color},
186 "name": "Despersonalizacion",
187 },
188 {
189 "x": dff["Burnout_Antes"],
190 "y": dff["Realizacion_Personal"],
191 "type": "bar",
192 "marker": {"color": color},
193 "name": "Realizacion_Personal",
194 }
195]
196 }
197)
198]

Se obtienen los datos alojados en big query con una búsqueda para relacionar el burnout con cada una de las variables categóricas.

Con el siguiente procedimiento se grafica el eje de las X y de las Y. En este caso, se muestra el código para graficar Burnout según su sub categorías, las cuales se dividen en Cansancio Emocional, Despersonalización y Realización Personal.

Ilustración 61: Código de la herramienta de visualización

Con el siguiente código se obtiene el panel de navegación, el cual es general para todas las páginas de la herramienta.

```
navbar = dbc.NavbarSimple(
    children=[
        dbc.NavItem(dbc.NavLink("Sistema de Detección", href="/deteccion")),
        dbc.NavItem(dbc.NavLink("Componentes Principales", href="/PCA")),
        dbc DropdownMenu(
            nav=True,
            in_navbar=True,
            label="Datos Categorizados",
            children=[
                dbc DropdownMenuItem("Burnout & Datos Fisiológicos por Paciente", href="/paciente"),
                dbc DropdownMenuItem(divider=True),
                dbc DropdownMenuItem("Burnout por Subescalas", href="/subescalas"),
                dbc DropdownMenuItem("Burnout por Subescalas Individuales", href="/subescalasindividual"),
                dbc DropdownMenuItem(divider=True),
                dbc DropdownMenuItem("Burnout por Especialidad", href="/especialidad"),
                dbc DropdownMenuItem("Burnout por Área de Trabajo", href="/trabajo"),
                dbc DropdownMenuItem(divider=True),
                dbc DropdownMenuItem("Burnout por Sexo", href="/sexo"),
                dbc DropdownMenuItem(divider=True),
                dbc DropdownMenuItem("Burnout por Estado Civil", href="/civil"),
                dbc DropdownMenuItem(divider=True),
                dbc DropdownMenuItem("Burnout por Tipo de Contrato", href="/contrato"),
                dbc DropdownMenuItem(divider=True),
                dbc DropdownMenuItem("Burnout por Tipo Número de Hijos", href="/hijos"),
                dbc DropdownMenuItem(divider=True),
                dbc DropdownMenuItem("Burnout por Edad", href="/edad"),
            ],
        ),
        dbc.NavItem(dbc.NavLink("Descargar Dataset", href="https://storage.googleapis.com/burnout/Burnout_Data.csv")),
    ],
    brand="Burnout Study",
    brand_external_link='https://storage.cloud.google.com/burnout/Burnout.png',
    #src="data:image/png;base64,{}".format(encoded_image),
    #brand_href="https://burnoutweb.dhns.net:80",
    brand_href="http://burnoutweb.dhns.net:80",
    #brand_href="http://localhost:80/",
    sticky="top",
)
```

Ilustración 62: Código del panel de navegación

A partir del siguiente código se obtienen las diversas tablas dinámicas, con las que se puede interactuar libremente, ofreciendo un dinamismo con el gráfico.

```

BurnoutxEstadoCivil = dbc.Container(
    [
        dbc.Row([
            dbc.Col([
                html.H4("Burnout por Estado Civil"),
                html.P(
                    "En este apartado de datos categorizados, mostramos los datos filtrando por una de las variables importantes EstadoCivil"),
                html.P(
                    "Son datos detallados por lo que están pensados para personas expertas en el tema. Para que les sirva en posteriores estudios."),
                dash_table.DataTable(
                    id="datatable-interactivity15",
                    columns=[
                        {"name": i, "id": i, "deletable": True, "selectable": True} for i in query.df_EstadoCivil.columns
                    ],
                    style_table={'overflowX': 'scroll'},
                    style_cell={
                        "# all three widths are needed"
                        'minWidth': '180px', 'width': '180px', 'maxWidth': '180px',
                        'overflow': 'hidden',
                        'textOverflow': 'ellipsis',
                    },
                    data=query.df_EstadoCivil.to_dict('records'),
                    filter_action="native",
                    sort_action="native",
                    sort_mode="multi",
                    row_selectable="multi",
                    row_deletable=True,
                    selected_rows=[],
                    page_actions=['native'],
                    page_current=0,
                    page_size=4,
                ),
                html.Div(id="datatable-interactivity-container15"),
            ],
            md=12,
        ),
    ],
    className="mt-4",
)

```

Ilustración 63: Código de tablas dinámicas

A continuación, se presentan algunas imágenes sobre la herramienta de desarrollo propio para visualizar y estudiar a profundidad el síndrome de estar quemado (burnout).

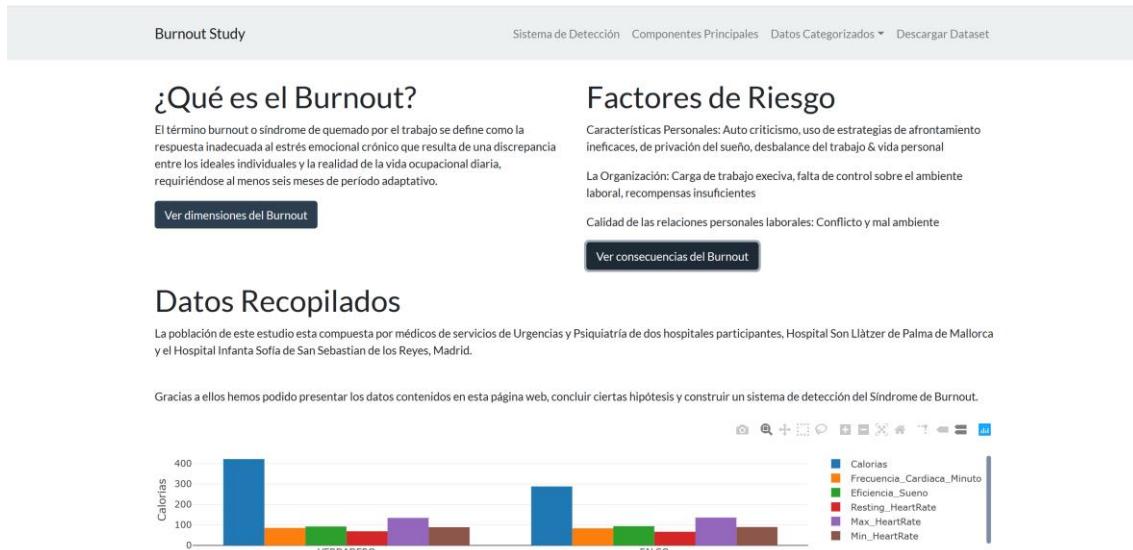
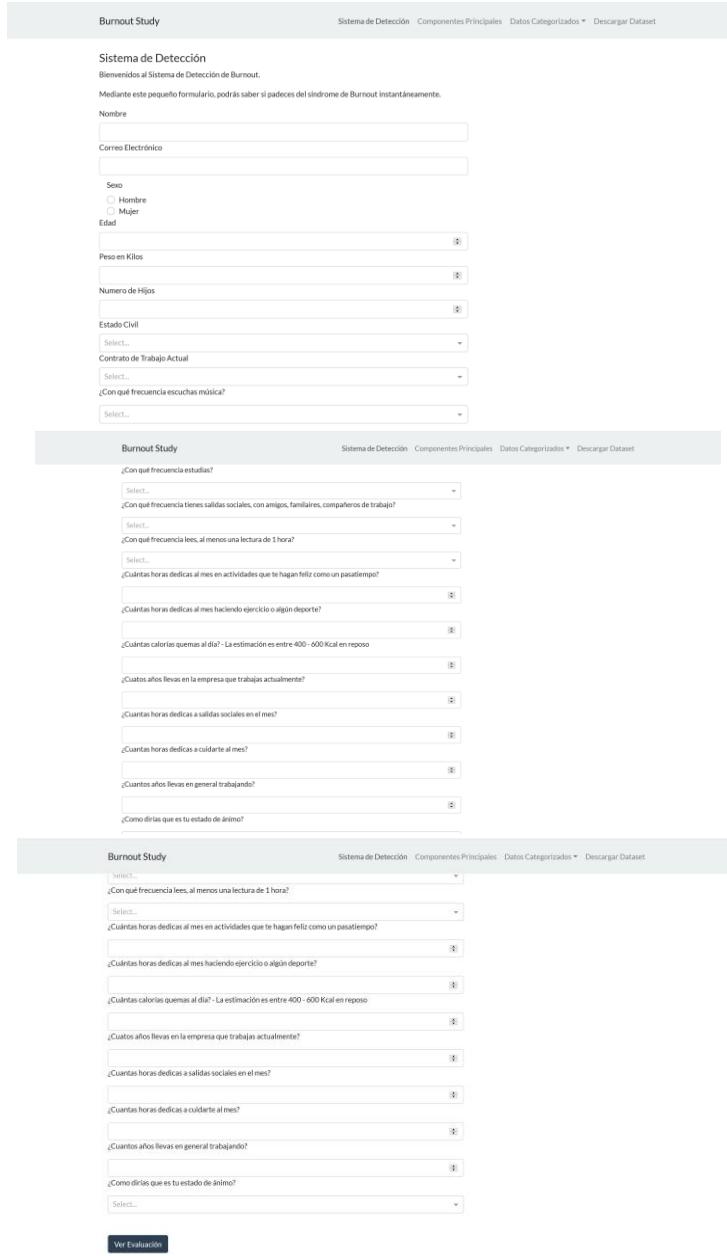


Ilustración 64: Página principal del Sistema Burnout Study

Se observa que, se cuenta con varias opciones que se irán detallando a lo largo de este Capítulo. En la sección de “Sistema de Detección” se puede acceder al cuestionario de predicciones del síndrome.



The three screenshots show the 'Burnout Study' detection system interface. The first screenshot shows basic personal information: Name, Email, Gender (Male or Female), Age, Weight in Kilograms, Number of Children, Civil Status, Work Contract, and Music Frequency. The second screenshot shows leisure and exercise habits: Exercise Frequency, Social Activities, Reading Frequency, Leisure Time, Calories Burned, Work Experience, Self-care, General Work, and Mood. The third screenshot shows additional details: Exercise Frequency, Leisure Time, Calories Burned, Work Experience, Self-care, General Work, and Mood. At the bottom of each page is a 'Ver Evaluación' button.

Ilustración 65: Formulario del sistema de detección.

Al presionar “Ver Evaluación” y haber cumplimentado el formulario, el sistema arrojaría la predicción con su consecuente recomendación.

The screenshot shows a web-based survey titled "Burnout Study". The user has completed several questions, including:

- "¿Cuántas horas dedicas al mes en actividades que te relajan?" (Never) - Response: 1
- "Click fuera de este recuadro para salir"
- "¿Cuántas horas dedicas al mes haciendo ejercicio o algún deporte?" - Response: 1
- "¿Cuántas calorías quemas al día? - La estimación es entre 400 - 600 Kcal en reposo" - Response: 1500
- "¿Cuatos años llevas en la empresa que trabajas actualmente?" - Response: 15
- "¿Cuantas horas dedicas a salidas sociales en el mes?" - Response: 1
- "¿Cuantas horas dedicas a cuidarte al mes?" - Response: 1
- "¿Cuantos años llevas en general trabajando?" - Response: 15
- "¿Como dirías que es tu estado de ánimo?" - Response: Triste

A modal window titled "Tu Evaluación" displays the prediction: "Enhorabuena!! test Tu evaluación es: FALSO !! Sigue así!!". Below the form, a green bar says "Tú evaluación está en curso, por favor espera, ¡No actualice la página!"

Ilustración 66: Predicción del Sistema Burnout Study

Otro menú interesante que ofrece una perspectiva sobre la razón por la cual se tomaron esas variables para el estudio, es el correspondiente a “Componentes Principales”, en el que se detalla la importancia de cada una de las características.

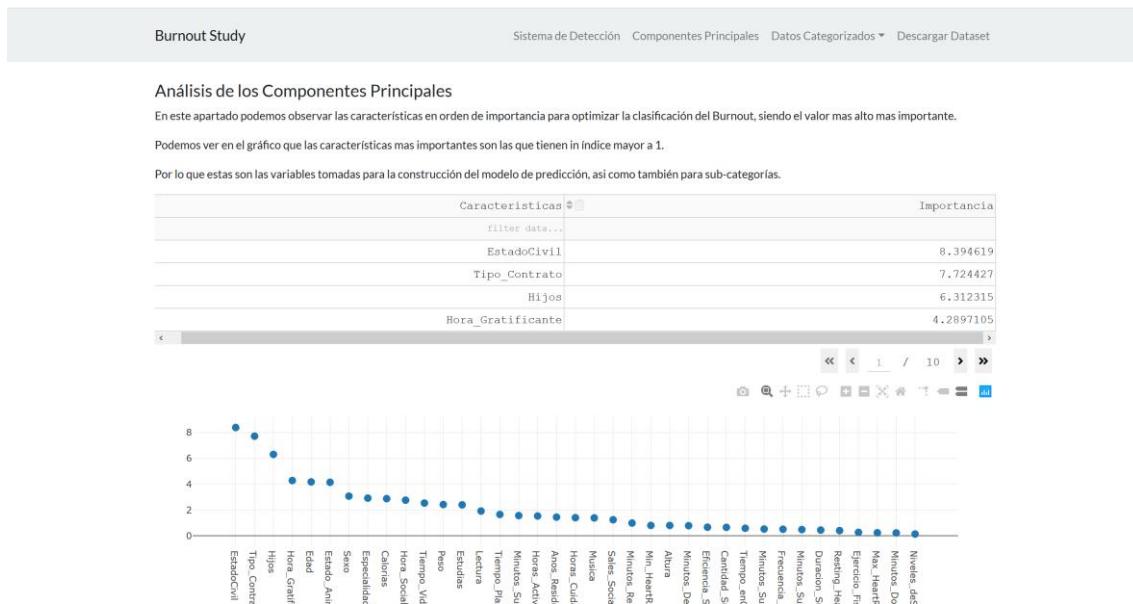


Ilustración 67: Importancia de las características

Luego, se encuentra el apartado de “Datos Categorizados”, en el cual se puede observar el tipo de afectación del burnout, en función de una serie de variables.

En primer lugar, se ha decidido volcar todos los datos fisiológicos por cada uno de los pacientes involucrados en el estudio, de manera que un experto en medicina – psiquiatría pueda contar con una herramienta de visualización de estos datos, pudiendo además sacar sus propias conclusiones.

Lo interesante de esta herramienta, es que le da la capacidad al investigador de interactuar con el gráfico y con la tabla, ya que estos son completamente dinámicos.

Es importante resaltar, que la mayoría de estos gráficos fueron solicitados por uno de los investigadores finales, la Doctora **María José Martín Vázquez**.

Datos Fisiológicos por Paciente:

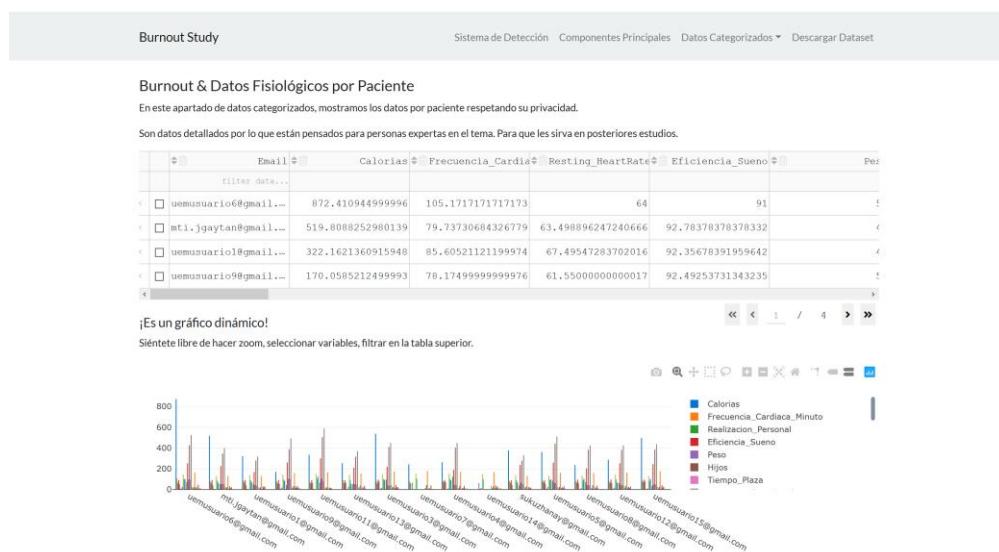


Ilustración 68: Datos fisiológicos por paciente

Burnout por Subescalas:

Burnout Study Sistema de Detección Componentes Principales Datos Categorizados ▾ Descargar Dataset

Burnout por Subescalas

En este apartado de datos categorizados, mostramos las subescalas del Burnout.

Son datos detallados por lo que están pensados para personas expertas en el tema. Para que les sirva en posteriores estudios.

Cansancio_Emocional	Despersonalización	Realización_Personal	Última_Encuesta_F	Última_Encuesta_I	Última_Encuesta
filter data...					
27.862476417436593	13	28.282721250549734	27.55880392073347	10.807509539413305	23.7689263372914
22.501275188475873	12.048251637192108	24.42698417147043	30.839460154641777	18.125379520379763	30.93218613080704
18.44196339075097	10.874165343038356	42.052606905487025	38.1691643406798	10.779394574922163	23.81151942264061
28.230715073572664	13.115328201638233	25.97119875149453	26.956777592640417	20.0143800864863	42.028818849592

¡Es un gráfico dinámico!

Siéntete libre de hacer zoom, seleccionar variables, filtrar en la tabla superior.

Primera Encuesta de Burnout

En esta primera gráfica observamos los datos recogidos en la primera encuesta

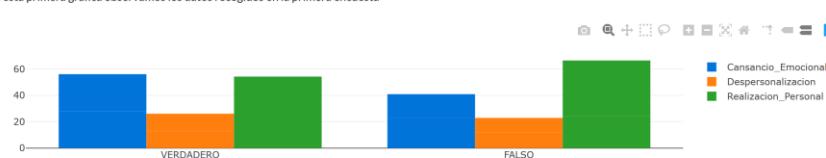


Ilustración 69: Burnout por Subescalas

Burnout por Subescalas Individuales:

Burnout Study Sistema de Detección Componentes Principales Datos Categorizados ▾ Descargar Dataset

Burnout por Subescalas Individuales

En este apartado de datos categorizados, mostramos las subescalas del Burnout por paciente.

Son datos detallados por lo que están pensados para personas expertas en el tema. Para que les sirva en posteriores estudios.

Cansancio_Emocional	Despersonalización	Realización_Personal	Última_Encuesta_F	Última_Encuesta_I	Última_Encuesta
filter data...					
27	13	26	48	5	2
26	15	25	23	15	2
21	11	23	32	19	2
21	17	34	30	16	2

¡Es un gráfico dinámico!

Siéntete libre de hacer zoom, seleccionar variables, filtrar en la tabla superior.

Primera Encuesta de Burnout

En esta primera gráfica observamos los datos recogidos en la primera encuesta por paciente

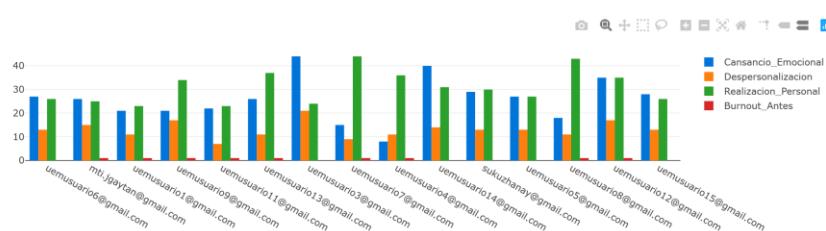


Ilustración 70: Burnout por Subescalas Individuales

Burnout por Especialidad:

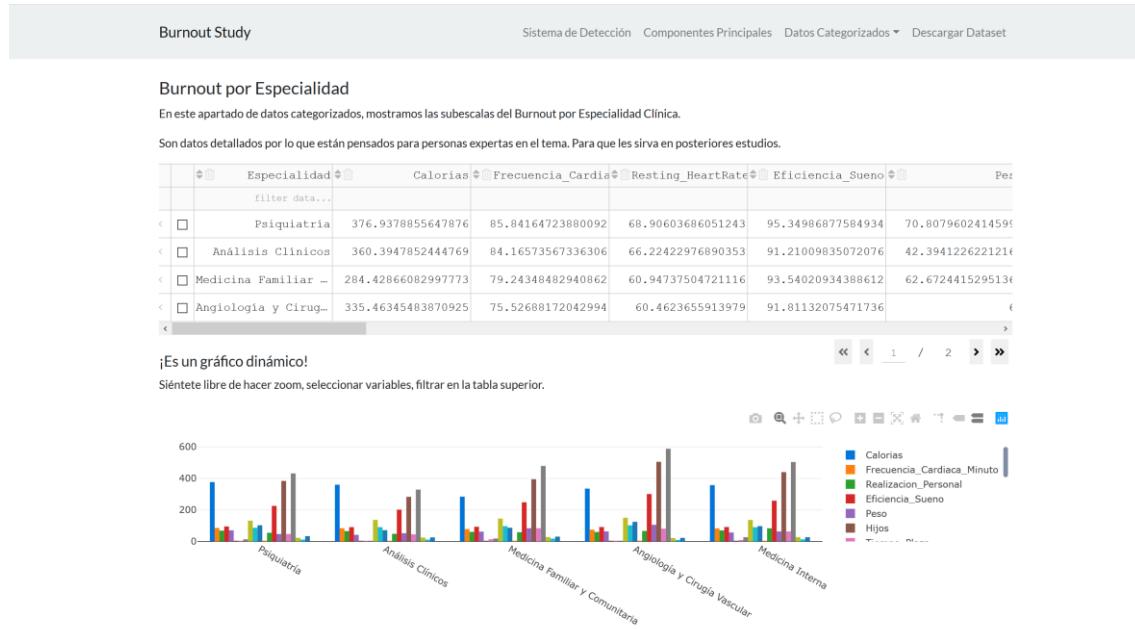


Ilustración 71: Burnout por Especialidad

Burnout por Área de Trabajo:

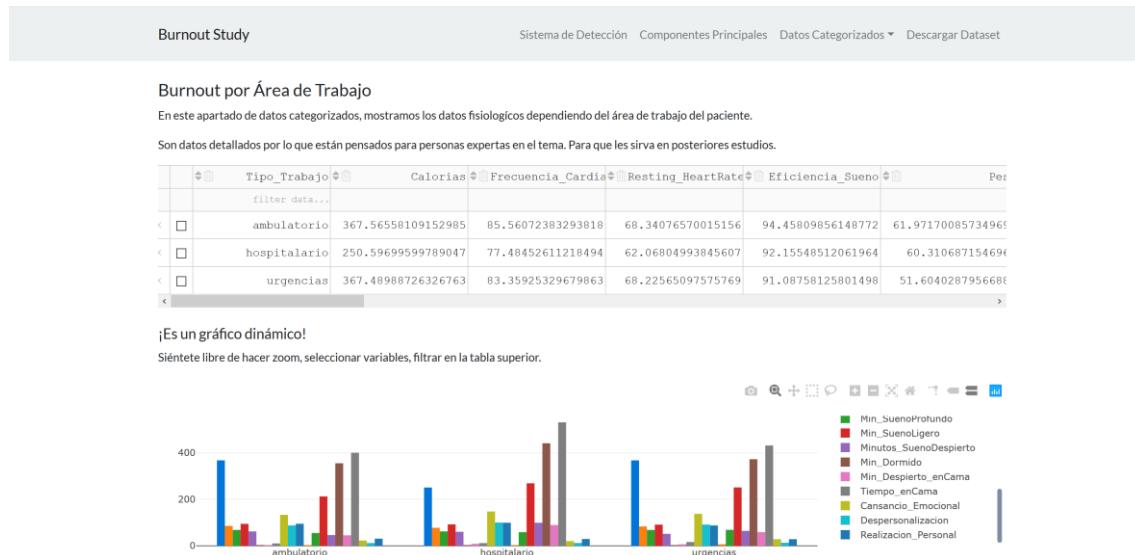


Ilustración 72: Burnout por Área de trabajo

Burnout por Sexo:

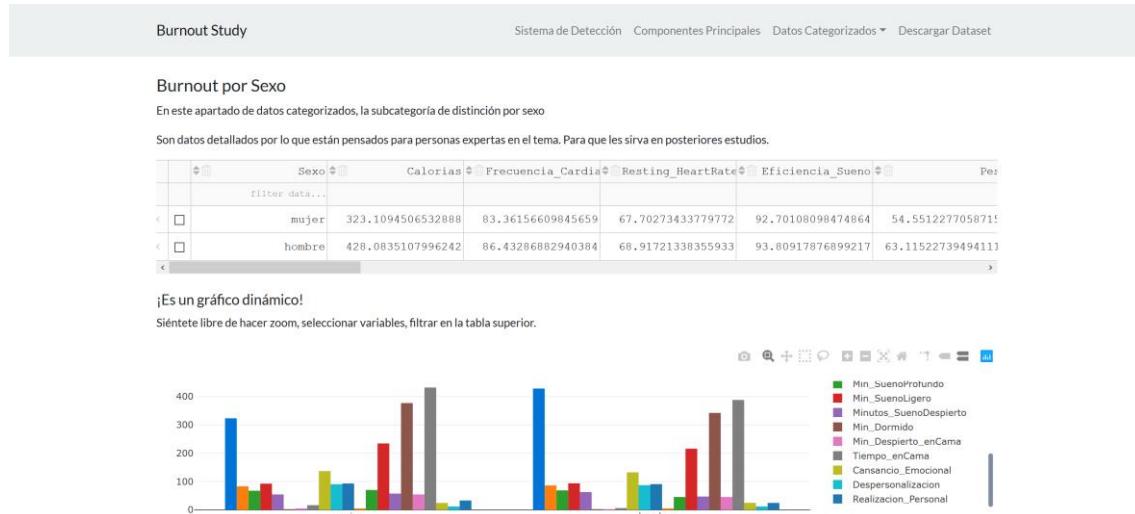


Ilustración 73: Burnout por Sexo

Burnout por Estado Civil:

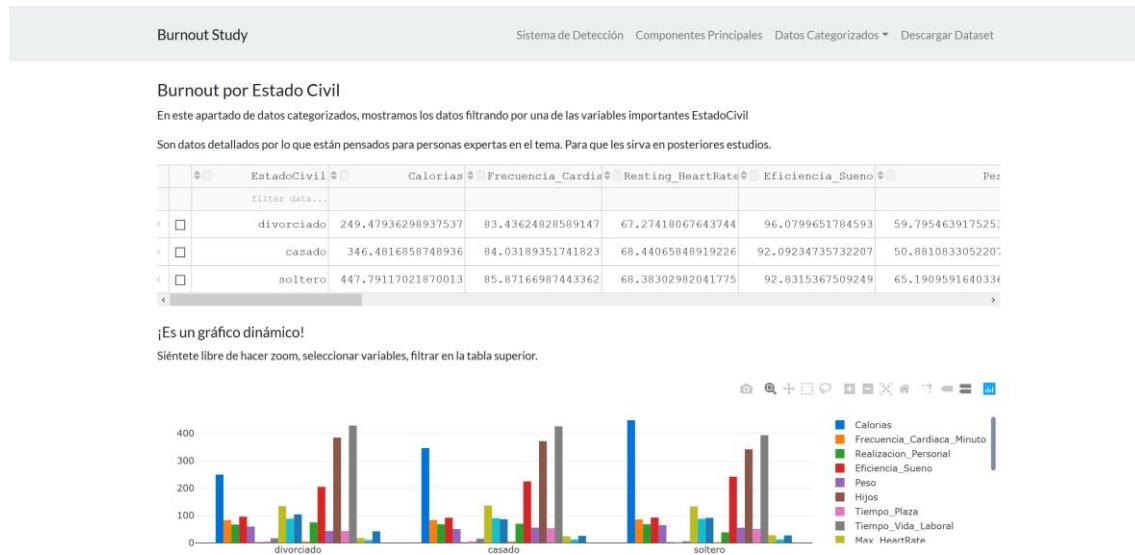


Ilustración 74: Burnout por Estado Civil

Burnout por tipo de Contrato:

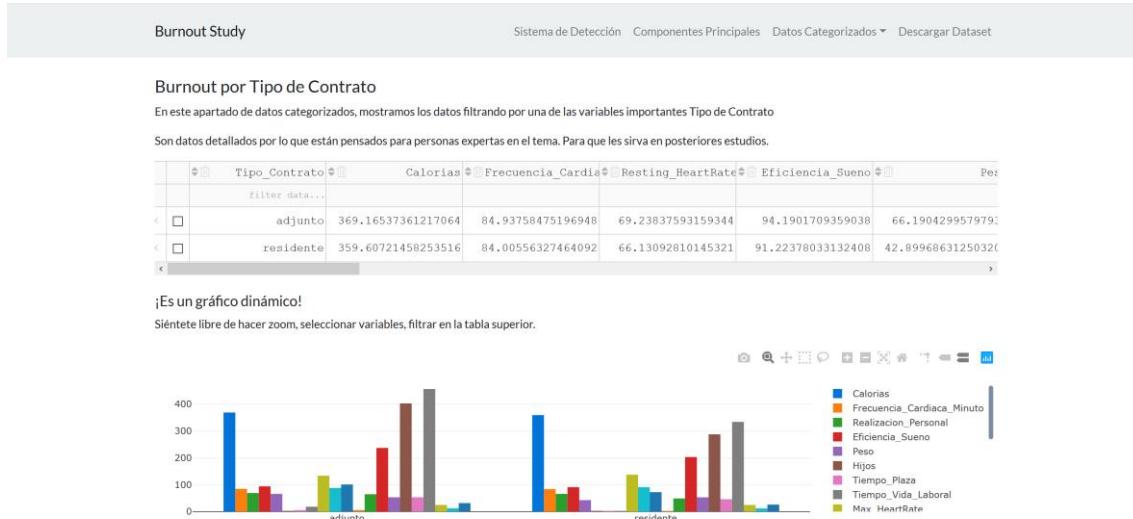


Ilustración 75: Burnout por Tipo de Contrato

Burnout por Número de Hijos:



Ilustración 76: Burnout por Número de Hijos

Burnout por Edad:

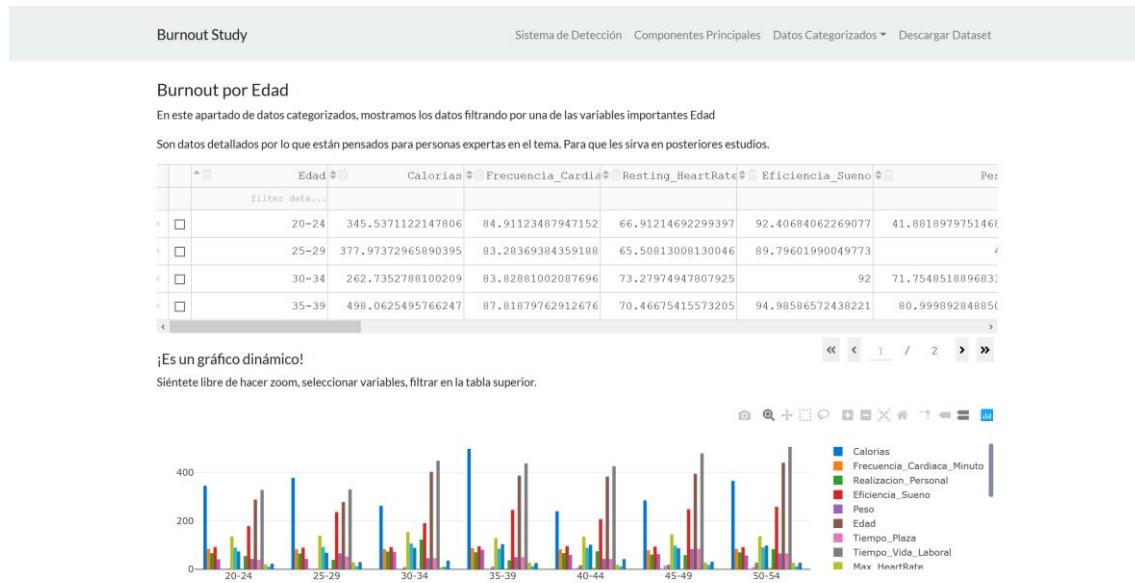


Ilustración 77: Burnout por Edad

Capítulo 7. Conclusiones

En este proyecto de fin de grado, se han involucrado una serie de tecnologías nuevas que han permitido su desarrollo exitoso, ya que, gracias a la inversión de Google Cloud, se ha podido tener a disposición los recursos necesarios para el procesamiento de todos los datos.

La experiencia con Google Cloud fue satisfactoria y de vital importancia, debido al gran volumen de datos que manejó el proyecto, además del amplio poder de cómputo que ofrece y las características de versatilidad, siendo de gran utilidad y apoyo para llevar a cabo todos los hitos.

En este sentido, a partir de los datos obtenidos, se puede observar que sí existe una relación entre el síndrome de estar quemado (burnout) y los datos fisiológicos. Estas conclusiones analíticas se obtienen, debido a que hay una relación entre la variable categórica de “estar casado” y el burnout, o “Hijos” y el burnout, tal y como se demuestra en la herramienta de visualización. Igualmente, es necesario destacar que existe una diferencia entre las personas que tienen el síndrome y las que no lo tienen, en cuanto a variables como las calorías, frecuencia cardiaca o calidad del sueño.

Sin embargo, se considera oportuno que un experto analice estos datos para poder concluir con mayor exactitud dicha relación, ya que este proyecto aporta la herramienta para realizar dicha investigación.

Por otro lado, el análisis de características principales ha sido clave para llevar a cabo el estudio, siendo posible desarrollar un sistema de detecciones del síndrome con un algoritmo con excelentes resultados.

En líneas generales, el proyecto ha logrado los objetivos planteados al inicio de este, por lo cual resulta conveniente detallar a continuación, algunas líneas futuras de trabajo posibles a partir de este gran hito.

Capítulo 8. Futuras Líneas de Trabajo

Este trabajo finaliza la etapa de análisis de IoT sobre el síndrome de burnout. Sin embargo, a partir de este surgen posibles nuevas líneas de investigación:

1. Resultaría interesante, volver a recopilar datos en diferentes hospitales y hacer pruebas exhaustivas sobre el algoritmo elegido para detectar el síndrome.
2. Un futuro trabajo sería que expertos en el síndrome analicen la herramienta creada para ellos y saquen conclusiones sobre el tema.
3. Sería interesante desplegar esta herramienta en diferentes hospitales para predecir las probabilidades que pueda tener el personal de sufrir el síndrome.
4. Sería posible expandir esta investigación y realizar un despliegue del modelo con pulseras conectadas en tiempo real, de manera que se puedan detectar anomalías con antelación al síndrome.

Por último, es posible mejorar la interfaz gráfica y la interacción del sistema con los usuarios.

PRESUPUESTO

El presupuesto de este sistema.

Descripción	Valor
Créditos utilizados en Google Cloud Platform	1034.89€
Tiempo invertido en horas de estudio del problema, reuniones, documentación, capacitación. Aprox 100 horas x 15 euros la hora	1500€
Tiempo invertido en el análisis, programación, data science. Aprox. 325 horas x 30 euros la hora	9750€
TOTAL	10784.89€

Ilustración 78: Presupuesto

BIBLIOGRAFÍA

Referencias usadas en esta investigación:

Susan Li. (2018). Multi-Class Text Classification with PySpark. 01/20/2020, de Towards Data Science Sitio web: <https://towardsdatascience.com/multi-class-text-classification-with-pyspark-7d78d022ed35>

Will Koehrsen. (2018). Data Visualization with Bokeh in Python. 02/12/2020, de Towards Data Science Sitio web: <https://towardsdatascience.com/data-visualization-with-bokeh-in-python-part-one-getting-started-a11655a467d4>

Plotly Dash. (2020). Dash User Guide. 02/12/2020, de Plotly Sitio web: <https://dash.plotly.com/>

Susan Li. (2018). Machine Learning with PySpark and MLlib — Solving a Binary Classification Problem. 01/15/2020, de Towards Data Science Sitio web: <https://towardsdatascience.com/machine-learning-with-pyspark-and-mllib-solving-a-binary-classification-problem-96396065d2aa>

Ali Masri. (2019). Feature Transformation. 01/17/2020, de Towards Data Science Sitio web: <https://towardsdatascience.com/apache-spark-mllib-tutorial-7aba8a1dce6e>

Dmitriy Kavyazin. (2019). Principal Component Analysis and k-means Clustering to Visualize a High Dimensional Dataset. 02/22/2020, de Medium Sitio web: <https://medium.com/@dmitriy.kavyazin/principal-component-analysis-and-k-means-clustering-to-visualize-a-high-dimensional-dataset-577b2a7a5fe2>

Aurea Grané. (2020). Análisis de Componentes Principales. 04/10/2020, de Universidad Carlos III de Madrid Sitio web: http://halweb.uc3m.es/esp/Personal/personas/agrane/ficheros_docencia/MULTIVARIANT/slides_comp_reducido.pdf

Prashant Gupta. (2017). Cross-Validation in Machine Learning. 01/24/2020, de Towards Data Science Sitio web: <https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f>

Tony Yiu. (2019). Understanding Bosques Aleatorios. 03/10/2020, de Towards Data Science Sitio web: <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>

Spark. (2020). Machine Learning Library (MLlib) Guide. 01/17/2020, de Spark Sitio web: <https://spark.apache.org/docs/latest/ml-guide.html>

Bill Chambers, Matei Zaharia. (2018). Spark: The Definitive Guide: Big Data Processing Made Simple . 01/17/2020, de Spark Sitio web: <https://ru.b-ok2.org/book/3505368/f04c83>