Tecnológico de Monterrey
Escuela de Ingeniería y Ciencias

NEUROFELLOWS

Reto: Modelo predictivo de satisfacción en proyecto solidario

TC2004B: Análisis de Ciencia de Datos

Descripción del Problema
En la Dirección de Servicio Social del Campus Monterrey, se busca constantemente mejorar las experiencias ofrecidas a través del Proyecto Solidario. Al finalizar cada ciclo de proyectos solidarios con diferentes socio formadores, como parte del programa de calidad, se envía una encuesta a los estudiantes involucrados. El propósito de esta encuesta es averiguar qué tan satisfecho está con su experiencia y relación con el coordinador de asistentes de maestros del programa. Esta encuesta consta de preguntas numéricas y también brinda la posibilidad de ingresar texto. Actualmente, los datos recopilados desde 2021 hasta el último período de 2023 se visualizan manualmente en los paneles de Dash Studio. Sin embargo, aún se está buscando una forma de predecir si los mensajes son buenos o malos, al mismo tiempo que se ahorra tiempo la organización socio formadora y se facilita su trabajo.

Miembros del Equipo:

Victor Hugo Garza Mantecón **(CDO y Científico de datos)**

Gian Marco Innocenti Castellanos **(BI y Científico de datos)**

Valeria Maria Serna Salazar **(Ingeniera y Científica de datos)**

Adalía Fernanda Aneiros Gutiérrez **(PM y Científica de datos)**

Diego Garza González **(Diseñador UX/UI y Científico de datos)**

Fecha de Entrega:

Miércoles 14 de junio de 2023
Tec de Mty, Campus Monterrey

Docentes:

Felipe Castillo Rendón
Angelina Alarcón Romero

Índice

Resumen (Abstract)

Utilizando 5 bases de datos que corresponden a las respuestas recabadas por una encuesta aplicada por la Dirección de Servicio Social del Tecnológico de Monterrey, se realiza una red neuronal encargada de predecir los valores que tomaría un análisis de sentimientos, de acuerdo a las otras respuestas de opción múltiple que da el usuario. El análisis de sentimiento se decide hacer por la justificación de diversos artículos académicos que indican su funcionamiento y mencionan la razón de su uso, se mencionan más a detalle en la parte 1.6. Al final, el entregable es un dashboard que visualiza las respuestas a la encuesta, incluyendo las predicciones del modelo y filtros que permiten ver la manera en que cada OSF se comporta. Se considera una propuesta de valor, pues significa un ahorro considerable de tiempo para los directivos y usuarios del proyecto realizado, pues se evitarán el tiempo y costo de hacer el análisis de sentimientos y, al mismo tiempo, brindamos una visualización simple y sencilla de entender, con filtros que hacen que el uso sea comprensible.

Marco Teórico

Según Brisebois, Abran, Nadembega y N'Techobo (2017), un análisis de sentimiento o satisfacción tiene el objetivo de determinar la actitud de una persona respecto a algo. La manera más común de aplicar este análisis es mediante aprendizaje automático, utilizando alguna de las técnicas existentes, como *detección de palabras clave*, donde se relacionan a éstas con un sentimiento específico; *afinidad léxica*, donde se le asigna a ciertas palabras una afinidad probabilística, o *métodos estadísticos*, que utiliza un conjunto de datos para entrenar un modelo y usarlo para clasificación. Independientemente de la técnica utilizada, el análisis de sentimiento sigue un proceso específico, que empieza por la adquisición de datos y termina con su visualización, pasando por el preprocesamiento, minería de datos y obtención y resumen de resultados. Después de obtener los datos y llegar a su preprocesamiento, en este tipo de análisis se efectúan una serie de pasos para ajustarlos y eliminar aquello innecesario. Entre ellos está la identificación de idioma, segmentación, eliminación de 'stopwords', tokenización, identificación de signos de puntuación y derivación de palabras.

Es importante destacar que el Machine Learning es una estrategia prometedora para categorizar reseñas y opiniones y se ha demostrado eficacia en varios estudios. En una revisión de Shukla, Bhosale y Boob (2017), se discutieron diferentes técnicas de análisis de sentimientos, incluidos los enfoques basados en el aprendizaje automático. Los

resultados de este estudio respaldan la idea de usar modelos de análisis de sentimientos para asignar calificaciones numéricas a las reseñas en las calificaciones de reseñas.

Para conseguir un mejor rendimiento para los modelos de Machine Learning se debe poder procesar una cantidad un poco grande de información. En el caso de esta situación problema el socio formador nos brindó con una base de datos que cuenta con más columnas categóricas que numéricas, ya que estas cuentan con muchas columnas llenas de comentarios escritos por un usuario. La información de estos comentarios debe ser transformada numéricamente para poder ser interpretados por un modelo de Machine Learning. Una herramienta utilizada para esto puede ser Análisis de Sentimiento. Según Rosenbrock el Análisis de Sentimiento "estudia la extracción de información a partir de datos subjetivos" (Rosenbrock, 2021). Estos modelos permiten otorgarle una calificación a un bloque de texto donde se le puede otorgar una calificación de positividad,negatividad, o neutralidad. Esta calificación numérica puede ser interpretada por el modelo que se defina, y es una gran manera para poder interpretar un mayor flujo de información no numérica.

Además, la conversión de variables categóricas mediante funciones como pd.get_dummies() se ha estudiado y utilizado ampliamente en la comunidad de Machine Learning. En un estudio de Liu y Zhang (2017), se exploró el uso de redes neuronales para analizar datos categóricos. Los resultados de este estudio muestran la eficiencia de la red neuronal en la representación y clasificación de variables categóricas, apoyando así el enfoque propuesto en la hipótesis de transformación de columnas categóricas mediante el uso de pd.get_dummies(). La literatura científica también apoya el uso de algoritmos de clasificación de texto en el contexto de la clasificación de revistas. En un estudio de Sebastiani (2002), se presentaron y compararon varios algoritmos de clasificación de texto. Este estudio ha destacado la efectividad de los enfoques basados en el aprendizaje automático en la clasificación de textos, apoyando la idea de utilizar modelos de clasificación en la hipótesis propuesta.

El artículo titulado 'Using Sentiment Analysis to Detect Customer Attitudes in Social Media Comments', publicado por la Universidad de Estambul, menciona como se hace un análisis de sentimiento para detectar las actitudes de los clientes en los comentarios de las redes sociales. Aunque no es exactamente igual a lo que se plantea resolver durante el reto, funciona de la misma manera al utilizar entradas de texto para analizar la manera de sentir de una persona. Durante el artículo, se menciona que con el aumento de las redes sociales, los clientes comparten sus opiniones sobre marcas, productos e individuos de manera continua en un entorno natural. Lo anterior, proporciona a los tomadores de decisiones información actualizada al instante y una oportunidad para recopilar datos de

muchas personas, lo que sería físicamente imposible mediante encuestas tradicionales. El análisis de sentimiento puede transformar datos de texto no estructurados en análisis cuantitativos estructurados, lo que permite a las empresas obtener información sobre las actitudes y preferencias del cliente. Al analizar el tono de las expresiones subjetivas, las empresas pueden identificar áreas de mejora en sus productos o servicios y tomar decisiones informadas sobre estrategias de marketing. De esta manera, el análisis de sentimiento es una herramienta importante para las empresas que buscan mantenerse competitivas en el paisaje digital rápido de hoy.

Finalmente, la aplicación del análisis de sentimientos para mejorar la calidad de las reseñas en línea ha sido objeto de varios estudios. En un estudio de Thelwall, Buckley y Paltoglou (2012), se demostró cómo el análisis de opiniones puede identificar y filtrar automáticamente las opiniones negativas o sesgadas, mejorando así la calidad de las opiniones. Este enfoque de análisis de sentimientos respalda la idea de usar modelos de similitud para clasificar las reseñas en el contexto planteado por la hipótesis.

1) Descripción Del Problema (Comprensión del Negocio)

1.1 Objetivos del negocio, beneficios esperados, metas y criterios de éxito

Los objetivos del Negocio creado es realizar un análisis de datos exhaustivo a una empresa que ofrece opciones de servicio social a estudiantes. También, se propone desarrollar un modelo predictivo de análisis de sentimiento para predecir los resultados de las preguntas de texto sin necesidad de realizar el análisis en tiempo real. Finalmente, se busca ahorrar tiempo y recursos computacionales a la Dirección de Servicio Social al evitar la necesidad de realizar el análisis de sentimientos en cada pregunta individual.

Para la Dirección de Servicio Social del Tecnológico de Monterrey se presentan tres beneficios principales. El primero es la eficiencia, pues al usar el modelo se reducirá significativamente el tiempo requerido para analizar las respuestas. El segundo es el ahorro de recursos ya que al evitar hacer el análisis de sentimientos para cada pregunta nueva se pueden ahorrar costos de infraestructura y energía.

La tercera es la mejora en la toma de decisiones, pues el dashboard final se encargará de proporcionar información valiosa que permitirá a la empresa comprender

mejor las necesidades y preocupaciones de los estudiantes, y así tomar decisiones estratégicas y más informadas en relación al servicio ofrecido.
Las metas a las que se busca llegar son:

1. Desarrollar un modelo predictivo preciso que pueda predecir los resultados de análisis de sentimientos en preguntas de texto con alta confiabilidad.
2. Integrar el modelo predictivo en el flujo de trabajo existente de la empresa, de manera que se pueda utilizar de manera eficiente y sin interrupciones en las operaciones diarias.
3. Reducir el tiempo dedicado al análisis de sentimientos en un porcentaje significativo, en comparación con el enfoque tradicional de análisis en tiempo real.
4. Crear un dashboard simple y sencillo de entender que facilite el trabajo de directivos y personas a cargo del servicio social.

Finalmente, algunos de los criterios de éxito que se tomarán en cuenta:

1. **Exactitud del modelo y claridad del Dashboard:** El modelo predictivo debe tener una alta precisión en la predicción de los resultados de análisis de sentimientos en preguntas de texto. También, el dashboard debe ser simple y tener filtros.
2. **Eficiencia en el tiempo:** Se debe lograr una reducción sustancial en el tiempo requerido para el análisis.
3. **Integración exitosa:** El modelo predictivo debe integrarse sin problemas en el flujo de trabajo existente de la empresa y ser fácilmente accesible para su uso por parte de los responsables de la toma de decisiones.
4. **Retroalimentación positiva:** Se deben recopilar comentarios y opiniones de los usuarios y responsables de la empresa sobre la utilidad y efectividad del modelo predictivo, y se espera que sean en su mayoría positivos.

1.2 Evaluación de situación actual

Actualmente, la organización cuenta con los resultados de las encuestas, pero es posible observar que no han trabajado mucho en ellas. Algunos de los factores que influyen y que hacen el análisis más complicado es que las preguntas entre bases de datos no son las mismas, tienen valores nulos, vacíos, algunas de las respuestas se contradicen, etc.

Entonces, el modelo propuesto si significa una mejora significativa en la manera en que la Dirección de Servicio Social se maneja, representando que la propuesta es de valor y

tendrá un impacto positivo en el trabajo que hacen todos los días. Al final, el objetivo es colaborar para facilitar el servicio social y, de tal manera, se promueve el desarrollo personal y profesional de los estudiantes al involucrarse en actividades que les interesan y que benefician a la comunidad.

1.3 Solución Propuesta

La solución propuesta consiste en utilizar técnicas de aprendizaje automático (machine learning) para abordar el problema que se ha presentado. Se plantea desarrollar un modelo predictivo basado en análisis de sentimientos que pueda predecir los valores numéricos de las respuestas sin la necesidad de realizar el análisis de sentimientos manualmente. Además, se propone implementar un modelo de clustering para identificar agrupaciones de elementos con similitudes, para tener una mayor comprensión de los datos. Ambos enfoques utilizarán algoritmos de aprendizaje automático para procesar los datos y extraer patrones relevantes. Para visualizar los resultados de manera clara y accesible, se creará un dashboard que mostrará los análisis de sentimientos y los clusters identificados, proporcionando una propuesta de valor al brindar información procesada de manera efectiva.

Cabe aclarar que el aprendizaje automático es esencial para la solución propuesta, ya que permite desarrollar modelos predictivos y algoritmos de clustering capaces de procesar grandes volúmenes de datos, obtener patrones relevantes y visualizar los resultados de manera efectiva. Esta capacidad de aprendizaje y automatización mejora la eficiencia, la precisión y la utilidad de la solución, brindando una propuesta de valor sólida y potenciando la toma de decisiones informada y estratégica en el ámbito del servicio social estudiantil.

1.4 Hipótesis

Si se aplican técnicas de ML como modelos de clustering y redes neuronales a las respuestas de opción múltiple proporcionadas por los usuarios en una encuesta, donde estos representan valores numéricos del 1 al 5 que indican el grado de satisfacción, entonces se podrá predecir el grado de satisfacción de una organización socio formadora y el trabajo que ha realizado.

1.5 Objetivo del Proyecto

1. Crear un dashboard sencillo de entender por los usuarios y clientes del departamento de Servicio Social, para así mejorar el servicio que ofrece.
2. Ahorrar la necesidad de tener que hacer un análisis de sentimiento a cada respuesta a las preguntas abiertas, al utilizar un modelo que ya está entrenado con los datos que han recibido.
3. Crear una interfaz clara, con diferentes colores y poder filtrar los datos generados para, así, facilitar el entendimiento por el usuario, y permitirles tomar decisiones que generen valor para la organización del socio formador.

1.6 Justificación

El análisis de la satisfacción de los alumnos es sumamente importante para poder brindar experiencias mucho más llamativas y enriquecedoras. La calidad de la educación y el bienestar de los estudiantes son aspectos fundamentales en el desarrollo de una sociedad sostenible y equitativa. Con el fin de asegurar una educación inclusiva, equitativa y de calidad para todos, es crucial conocer las necesidades y expectativas de los alumnos. En ese sentido, se han llevado a cabo encuestas durante cinco períodos consecutivos para recabar información sobre la satisfacción de los alumnos. Estas encuestas han brindado valiosos datos que nos permiten entender mejor sus percepciones, inquietudes y áreas de mejora. La información obtenida a través de este proceso se ha revelado sumamente útil para cumplir el objetivo de brindar experiencias educativas más satisfactorias y enriquecedoras.

El propósito de mejorar la experiencia de los alumnos no solo se limita a su satisfacción personal, sino que también tiene un impacto directo en la participación de los mismos en causas sociales y en la contribución a los Objetivos de Desarrollo Sostenible (ODS). Al incrementar la cantidad de alumnos que se involucran activamente en proyectos y actividades relacionadas con los ODS, se fomenta la conciencia sobre los desafíos globales y se promueve el compromiso con la construcción de un futuro sostenible. Al impulsar la participación de los alumnos en acciones que respalden los ODS, se fortalece su sentido de responsabilidad y se cultivan valores como la solidaridad, la equidad y la sostenibilidad. Además, al trabajar de manera colaborativa en proyectos relacionados con los ODS, los alumnos adquieren habilidades y competencias clave para su desarrollo personal y profesional, como el pensamiento crítico, la resolución de problemas y el trabajo en equipo.

1.7 Mercado Potencial

Considerando que el análisis se realiza con el objetivo de tener una clara visualización sobre el desempeño de los socio formadores, el mercado potencial son los directivos y coordinadores del departamento de servicio social del Tecnológico de Monterrey. Así, lograrán tomar decisiones informadas al conocer la experiencia general de los alumnos y su sentir en las encuestas.

1.8 Identificación de clientes, consumidor y usuarios

El cliente es el departamento de Servicio Social del Tecnológico de Monterrey, pues para ellos es el análisis a realizar. Por otro lado, los usuarios son los coordinadores, directivos, y otras personas encargadas de tomar decisiones que indican la dirección a donde va dirigido su proyecto, basado en los datos que observan y que brinda nuestro programa.

1.9 Plan de actividades del Proyecto

Línea del Tiempo utilizando la metodología CRISP	
Fase 1: Entendimiento de los Datos	
22 de mayo	Comprender a fondo el conjunto de datos, su estructura y sus variables. Identificar también datos faltantes e incompletos.
Fase 2: Limpieza y tratamiento de los datos	
23 a 28 de mayo	Limpieza del conjunto de datos manejando los valores faltantes, los valores atípicos y los datos inconsistentes. Se realizarán, también, las transformaciones de datos necesarias. Realizar técnicas avanzadas para la minería de datos, al investigar el agrupamiento o reglas de asociación, aplicarlo y, de tal manera, lograr obtener conocimientos más profundos sobre el conjunto de los datos.
Fase 3: Creación del modelo de agrupamiento	

29 de mayo a 2 de junio	Selección del mejor modelo según los resultados que se obtienen, así como el entrenamiento y optimización de los modelos usando algoritmos.
3 de junio	Se toma una muestra del conjunto de datos para analizar aspectos específicos y asegurarnos que todo está en orden.
Fase 4: Evaluación del modelo y ajustes	
5 de junio	Evaluación del rendimiento del modelo, comparación entre varios y selección del mejor.
8 de junio	Preparación del modelo seleccionado para su implementación. También, se pasa a un formato presentable con comentarios claros.
Fase 5: Modificaciones finales	
9 de junio	Análisis del rendimiento del modelo, así como los resultados. Decidir si se requieren pasos o ajustes adicionales.
11 de junio	Se analiza la posibilidad de incluir un escalamiento o normalización de los datos, si es necesario, se realiza.
13 de junio	Se realiza una presentación para entregar los resultados de una manera más clara.
Fase 6: Entregable del modelo	
15 de junio	Se entrega el programa final.

2) Comprensión de los datos

2.1 Descripción del set de datos

Variable	Descripción	Tipo de	Tipo de	Dataset en

		dato	variable	el que se encuentra
StartDate	Fecha y hora en que se empezó la encuesta.	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
EndDate	Fecha y hora en que se finalizó la encuesta.	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
Status	Tipo de respuesta por parte del usuario	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
IP Address	Dirección IP del usuario	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
Progresss	Progreso respecto al llenado de la encuesta	int	Cuantitativa	INV_23 FJ23 FJ22 FD21 AD22
Duration (in seconds)	Segundos que le tomó a cada usuario contestar la encuesta	int	Cuantitativa	INV_23 FJ23 FJ22 FD21 AD22
Finished	Indica si se finalizó o no la encuesta	bool	Cualitativa	INV_23 FJ23 FJ22

				FD21 AD22
RecordedDate	Fecha y hora en que la encuesta quedó registrada	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
ResponseID	ID asociado a cada respuesta	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
RecipientEmail	Correo electrónico del destinatario	string	Cualitativa	INV_23 FJ23 FD21 AD22
LocationLatitud	Latitud de la ubicación de cada respuesta	float	Cuantitativa	INV_23 FJ23 FJ22 FD21 AD22
LocationLongitud	Longitud de la ubicación de cada respuesta	float	Cuantitativa	INV_23 FJ23 FJ22 FD21 AD22
UserLanguage	Idioma utilizado al contestar la encuesta	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
DistributionChannel	Canal de distribución de la encuesta	string	Cualitativa	FJ22 FD21 AD22

Al concluir este Bloque con Sentido Humano	Evaluación de satisfacción respecto a la conclusión del proyecto solidario	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
Vivir la experiencia de aprendizaje relacionada con un Objetivo de Desarrollo Sustentable	Evaluación de satisfacción respecto al aprendizaje relacionado con un ODS	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
Nivel de sensibilización que aportó esta unidad de formación para la atención de necesidades sociales	Evaluación de satisfacción respecto al nivel de valor aportado a la OSF a través de los entregables.	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
Seguimiento y liderazgo que dio la organización socio formadora en el desarrollo de tu experiencia	Evaluación de satisfacción respecto a los momentos de interacción con los beneficiarios del proyecto	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
Herramientas que aplicaste como las actividades, reportes, “quizzes”, dentro de la plataforma de CANVAS en el desarrollo de la experiencia de Servicio Social	Evaluación de satisfacción respecto a las herramientas aplicadas para el desarrollo de la experiencia	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22

Q2_6	Evaluación de satisfacción respecto a la colaboración con la OSF	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
Q2_7	Evaluación de satisfacción respecto al seguimiento y liderazgo de la OSF	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
Q2_8	Evaluación de satisfacción respecto al servicio proporcionado por el área de Servicio Social del campus	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
Q1	Indica el nombre de la OSF y del proyecto	string	Cualitativa	INV_23
Q2	ID correspondiente a cada registro de respuesta	string	Cualitativa	FJ23
Q3	Opinión acerca de la experiencia con el servicio social	string	Cualitativa	INV_23 FJ23 AD22 FJ22 FD21
Q5	Indica si la OSF ofreció retroalimentación sobre el desarrollo del proyecto	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
Q6	Comentario a compartir con la OSF	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22

Q7	Categoría a la que pertenece el comentario (variable Q3). En FJ23, éste indica el comentario a compartir con la OSF.	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
Q9	Se indica si se considera interesante la causa social de la OSF	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22 AD22
Q10	Categoría a la que pertenece el comentario (variable Q7).	string	Cualitativa	FJ23
Q11	Indica el nombre de la OSF y del proyecto	string	Cualitativa	AD22
Al concluir este Bloque con Sentido Humano	Opinión respecto a la oportunidad de generar un sentimiento de sensibilidad gracias a la experiencia	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
Vivir la experiencia de aprendizaje relacionada con un Objetivo de Desarrollo Sustentable	Opinión respecto a la oportunidad de actuar con responsabilidad	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
Nivel de sensibilización que aportó esta unidad de formación para la atención de necesidades sociales	Opinión respecto a la oportunidad de actuar con respeto	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22

Seguimiento y liderazgo que dio la organización socio formadora en el desarrollo de tu experiencia	Opinión respecto a la oportunidad de promover soluciones en relación a las problemáticas	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
OSF	Nombre de la Organización Socio Formadora	string	Cualitativa	INV_23 FJ23 FJ22 FD21 AD22
CRN	Número identificador de la Organización Socio Formadora	int	Cuantitativa	INV_23 FJ23 FJ22 FD21 AD22
Nombre de Experiencia	Nombre del proyecto	string	Cualitativa	INV_23 FJ23 FJ22 AD22
Periodo	Número identificador del periodo donde se llevó a cabo la experiencia	int	Cuantitativa	INV_23 FJ23 FJ22 AD22
Tipo de formato	Formato en el que se desarrolló la experiencia	string	Cualitativa	INV_23 FJ23 FJ22 AD22
Semana	Número de semana o semanas en la que se desarrolló la experiencia	string	Cualitativa	FJ22

Cabe recalcar que las siguientes variables se encuentran en algunos datasets pero no guardan ningún valor:

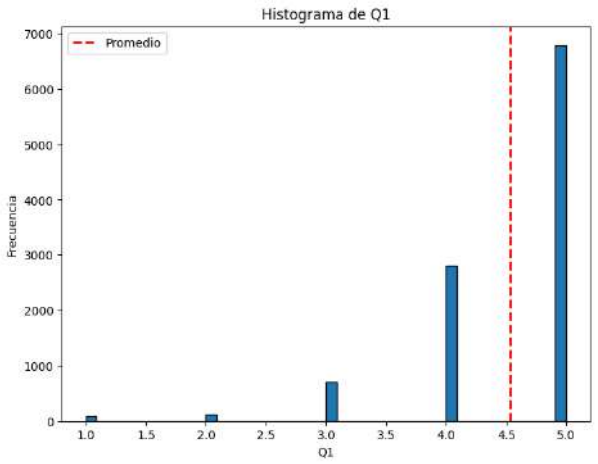
1. RecipientFirstName
2. ExternalReference

Además, la variable ‘LastName’ no cuenta con datos en ningún dataset exceptuando el de AD22, sin embargo, se esperaría que indicara el apellido del destinatario, mientras que en realidad muestra algún tipo de identificador de cada respuesta, del tipo @00001, por ejemplo.

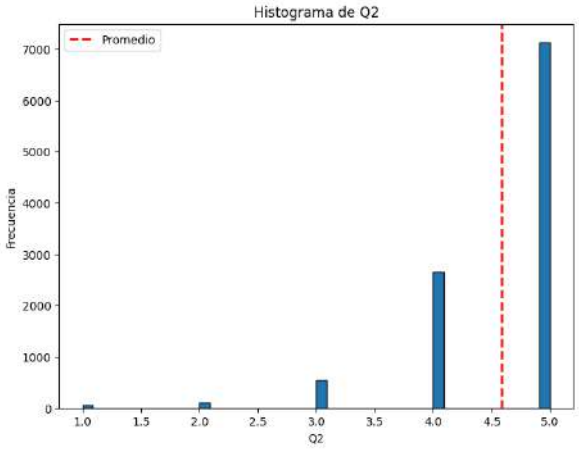
2.2 Exploración de datos

En el análisis exploratorio de los datos se imprimen diferentes gráficas para entender las variables categóricas, estas son:

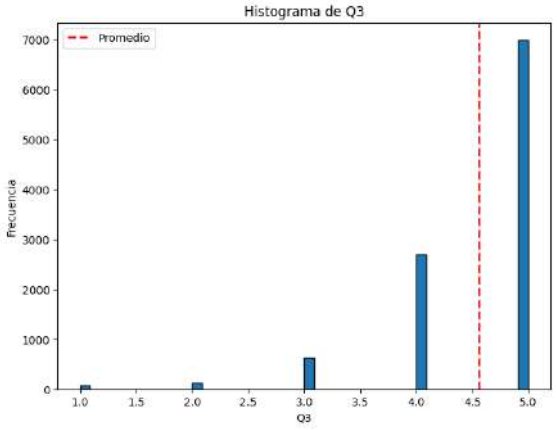
La primer variable que en el data frame aparece como ‘Q1’, pero en realidad es ‘1. Evalúa tu nivel satisfacción en los siguientes aspectos: - a) Al concluir este Proyecto Solidario’, se puede responder con valores numéricos del 1-5. La gráfica de barras muestra que el promedio es 4.5, pero la mayoría de los valores son positivos.



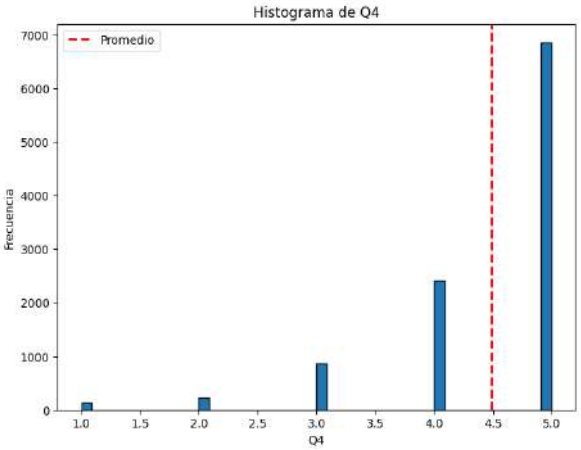
La siguiente variable aparece en el data frame aparece como ‘Q2’, pero en realidad es ‘Vivir la experiencia de aprendizaje relacionada con un Objetivo de Desarrollo Sostenible’, se puede responder con valores numéricos del 1-5. La gráfica de barras muestra que el promedio es un poco mayor a 4.5, y la mayoría son positivos también.



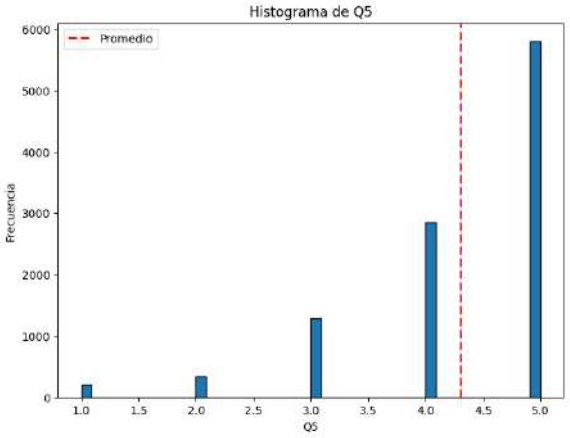
La siguiente variable aparece en el data frame aparece como ‘Q3’, pero en realidad es ‘Nivel de sensibilización que aportó esta unidad de formación para la atención de necesidades sociales’, se puede responder con valores numéricos del 1-5. La gráfica de barras muestra que el promedio es 4.5, y la mayoría son positivos.



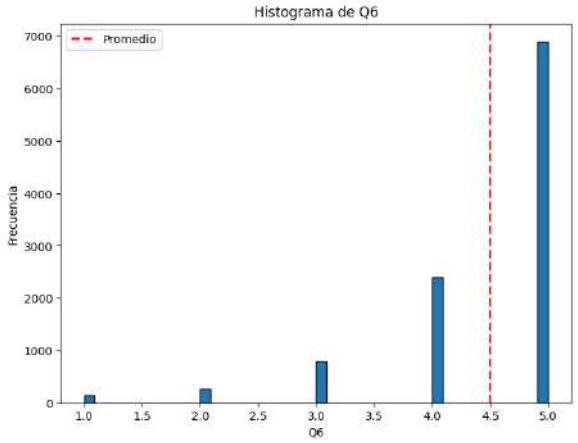
La siguiente variable aparece en el data frame aparece como ‘Q4’, pero en realidad es ‘1. Evalúa tu nivel satisfacción en los siguientes aspectos: - e) Momentos de interacción y escucha con los beneficiarios/destinatarios del proyecto’, se puede responder con valores numéricos del 1-5. La gráfica de barras muestra que el promedio es 4.5, y la mayoría son positivos.



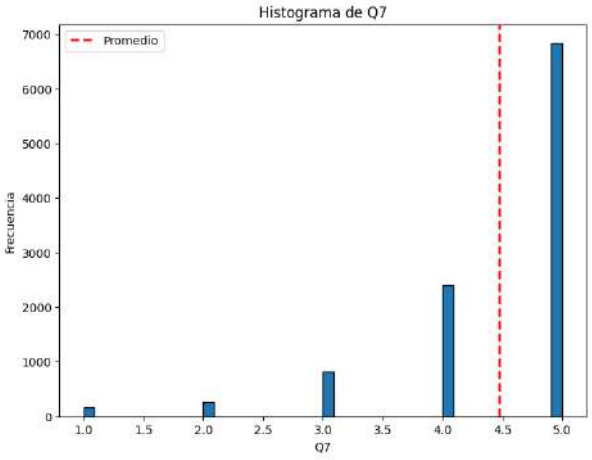
La siguiente variable aparece en el data frame aparece como ‘Q5’, pero en realidad es ‘1. Evalúa tu nivel satisfacción en los siguientes aspectos: - f) Herramientas que aplicaste como las actividades, reportes, "quizzes", dentro de la plataforma de CANVAS en el desarrollo de la experiencia de Servicio Social.’, se puede responder con valores numéricos del 1-5. La gráfica de barras muestra que el promedio es un poco mayor a 4.0, y, aunque en la mayoría son positivos, están mejor distribuidos.



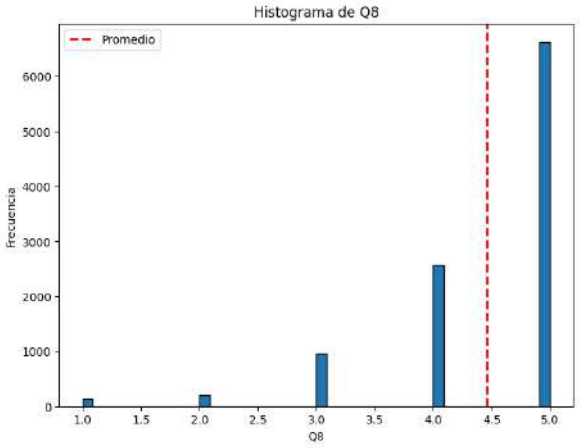
La siguiente variable aparece en el data frame aparece como ‘Q6’, pero en realidad es ‘1. Evalúa tu nivel satisfacción en los siguientes aspectos: - g) Experiencia de colaboración con la organización socio formadora’, se puede responder con valores numéricos del 1-5. La gráfica de barras muestra que el promedio es 4.5 y la mayoría son positivos.



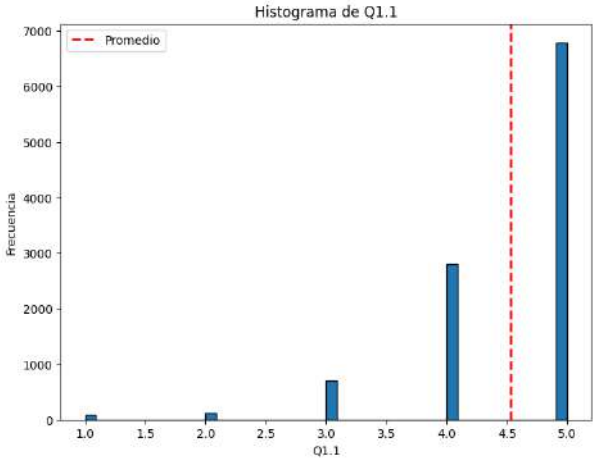
La siguiente variable aparece en el data frame aparece como ‘Q7’, pero en realidad es ‘1. Evalúa tu nivel satisfacción en los siguientes aspectos: - h) Seguimiento y liderazgo de la organización socio formadora’, se puede responder con valores numéricos del 1-5. La gráfica de barras muestra que el promedio es un poco menor a 4.5 y la mayoría son positivos.



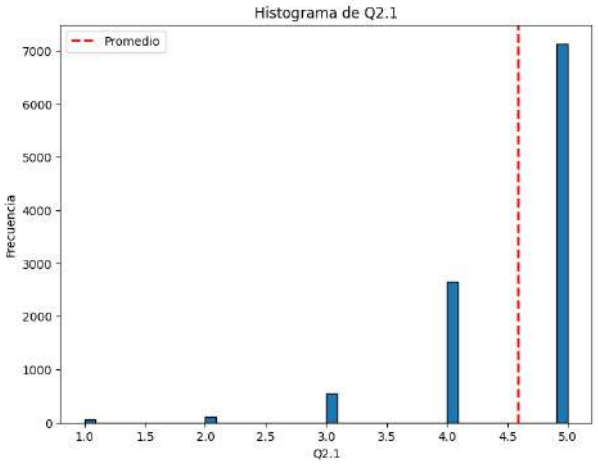
La siguiente variable aparece en el data frame aparece como ‘Q8’, pero en realidad es ‘1. Evalúa tu nivel satisfacción en los siguientes aspectos: - i) Atención y servicio del área que administra el Servicio Social en el campus (asesoría y orientación, información puntual, atención de dudas, seguimiento de incidentes)’, se puede responder con valores numéricos del 1-5. La gráfica de barras muestra que el promedio es un poco menor a 4.5 y la mayoría son positivos.



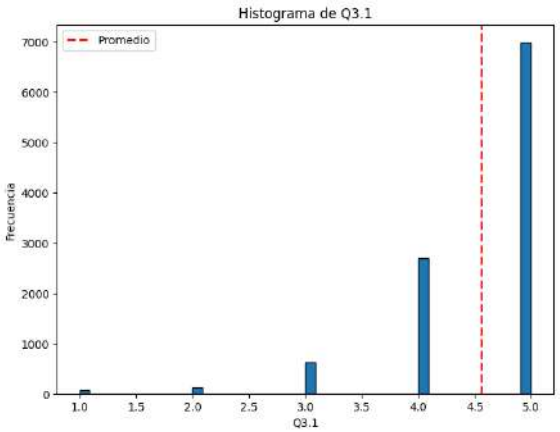
La siguiente variable aparece en el data frame aparece como ‘Q1.1’, pero en realidad es ‘5. Expresa tu opinión en los siguientes enunciados, en esta experiencia de servicio social tuve la oportunidad de: - a) Ser sensible ante la vulnerabilidad, el dolor y el sufrimiento del otro y actuar con el fin de eliminarlo, aliviarlo o evitarlo, a través de acciones justas alejadas de la pasión egoísta y/o de sentimientos de superioridad’, se puede responder con valores numéricos del 1-5. La gráfica de barras muestra que el promedio es un poco mayor a 4.5 y la mayoría son positivos.



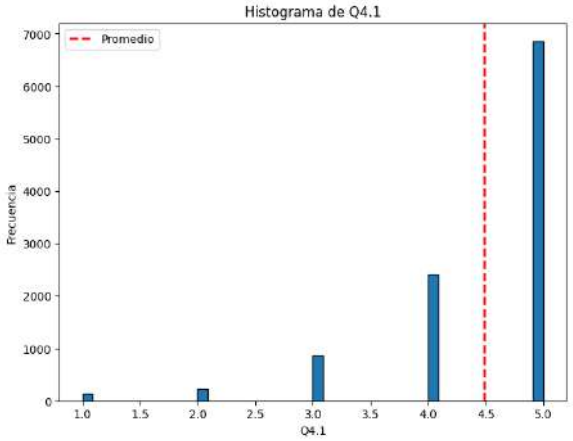
La siguiente variable aparece en el data frame aparece como ‘Q2.1’, pero en realidad es ‘5. Expresa tu opinión en los siguientes enunciados, en esta experiencia de servicio social tuve la oportunidad de: - b) Actuar con responsabilidad, con el fin de asegurar el bienestar de la colectividad, a través de acciones que garantizan el acceso a los derechos humanos, el empoderamiento de los ciudadanos y de las comunidades, así como el cuidado, mantenimiento y uso sostenible de los recursos y bienes comunes’, se puede responder con valores numéricos del 1-5. La gráfica de barras muestra que el promedio es un poco mayor a 4.5 y la mayoría son positivos.



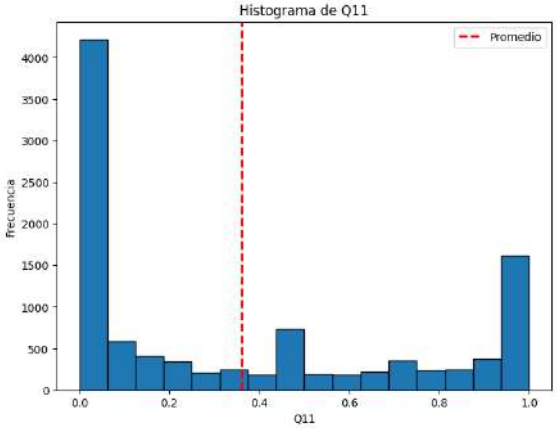
La siguiente variable aparece en el data frame aparece como ‘Q3.1’, pero en realidad es ‘5. Expresa tu opinión en los siguientes enunciados, en esta experiencia de servicio social tuve la oportunidad de: - c) Actuar con respeto ante la diversidad de género, sexual, étnica, cultural, de capacidades, generacional, religiosa y socioeconómica mostrando una cordial aceptación de las diferencias y la capacidad para gestionar de manera razonable los conflictos’, se puede responder con valores numéricos del 1-5. La gráfica de barras muestra que el promedio es un poco mayor a 4.5 y la mayoría son positivos.



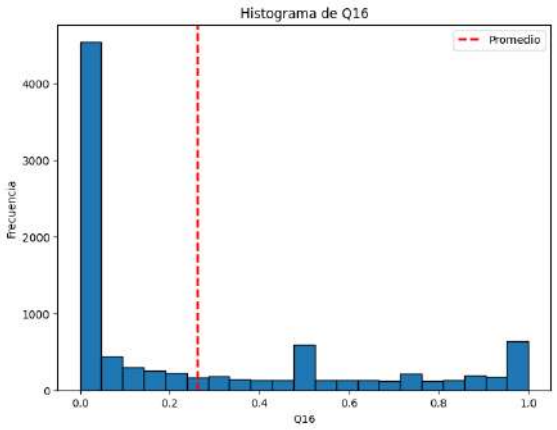
La siguiente variable aparece en el data frame aparece como ‘Q4.1’, pero en realidad es ‘5. Expresa tu opinión en los siguientes enunciados, en esta experiencia de servicio social tuve la oportunidad de: - e) Promover soluciones cooperativas en problemas o coordinar acciones colectivas con el fin de mejorar la calidad de vida de la sociedad, fomentando la cultura de la legalidad, los derechos humanos y/o el fortalecimiento de la democracia’, se puede responder con valores numéricos del 1-5. La gráfica de barras muestra que el promedio es 4.5 y la mayoría son positivos.



La siguiente variable aparece en el data frame aparece como ‘Q11’, pero en realidad es ‘4. Escribe algún comentario que te interese compartir con la organización socio formadora:’, aunque es respondido con input de texto, se ha aplicado un análisis de sentimientos, que será explicado más adelante, y cuyo resultado varía de 0 a 1. El resultado muestra que la mayoría de resultados son 0, significando que son mayormente negativos; pero hay suficientes con 1, significando que son positivos.



La última variable aparece en el data frame aparece como ‘Q16’, pero en realidad es ‘6. Comparte algo en particular sobre la experiencia de servicio social que acabas de vivir. Nos interesa conocer tu opinión:’, aunque es respondido con input de texto, se ha aplicado un análisis de sentimientos, que será explicado más adelante, y cuyo resultado varía de 0 a 1. El resultado muestra que la mayoría de resultados son 0, significando que son mayormente negativos; pero hay algunos con 1, significando que son positivos.



1. **Estadísticos**
    a. Se crean los estadísticos correspondientes con la función y se descargan en un csv llamado ‘tabla de estadísticos’.

```python
# Se crea la matriz de estadisitcos y se gurada en un csv
estadistico_column=calculate_statistics(df)
estadistico_column.to_csv('tabladeestadisticos.csv')
```

Python

    b. El documento que se descarga tiene los siguientes resultados:

	mean	median	max	min	var	std
1. Evalúa tu nivel satisfacción en los siguientes aspectos: - a) Al co	4.4841461	5	5	1	0.5881662	0.7669199
1. Evalúa tu nivel satisfacción en los siguientes aspectos: - b) Vivir	4.5882633	5	5	1	0.4834674	0.6953182
1. Evalúa tu nivel satisfacción en los siguientes aspectos: - c) Nivel	4.5611127	5	5	1	0.5187735	0.7202593
1. Evalúa tu nivel satisfacción en los siguientes aspectos: - e) Mom	4.4887111	5	5	1	0.6978765	0.8353901
1. Evalúa tu nivel satisfacción en los siguientes aspectos: - f) Herr	4.3046585	5	5	1	0.8947887	0.9459327
1. Evalúa tu nivel satisfacción en los siguientes aspectos: - g) Expe	4.4921406	5	5	1	0.7084224	0.8416783
1. Evalúa tu nivel satisfacción en los siguientes aspectos: - h) Segu	4.4737544	5	5	1	0.7523837	0.8674005
1. Evalúa tu nivel satisfacción en los siguientes aspectos: - i) Atenc	4.4607983	5	5	1	0.7063756	0.8404616
3. ¿Consideras interesante la causa social del socio formador?	0.3883587	0.3969751	0.999956	1.02E-27	0.0064241	0.0801506
4. Escribe algún comentario que te interese compartir con la orga	0.3602776	0.2148432	1	2.17E-48	0.1415096	0.3761776
5. Expresa tu opinión en los siguientes enunciados, en esta experi	3.7573718	4	4	2	0.2363996	0.4862095
5. Expresa tu opinión en los siguientes enunciados, en esta experi	3.851626	4	4	2	0.1573174	0.3966326
5. Expresa tu opinión en los siguientes enunciados, en esta experi	3.8138733	4	4	2	0.1920661	0.4382534
5. Expresa tu opinión en los siguientes enunciados, en esta experi	3.8144759	4	4	2	0.1919502	0.4381212
6. Comparte algo en particular sobre la experiencia de servicio so	0.2608374	0.045606	1	5.40E-74	0.114474	0.3383401
6.1 Tu comentario sobre la experiencia es:	0.4954237	0.4978923	0.9999749	7.21E-33	0.0064895	0.0805577

    c. En el archivo de excel se muestra el promedio de valores que toma la pregunta correspondiente, en donde los enunciados toman un menor valor a diferencia de los aspectos que se pide identificar. También, es posible observar que la varianza varía dependiendo de la pregunta, factor que puede ser esencial en el análisis de los datos. Además, factores como la desviación estándar también son interesantes, pues varían considerablemente; en algunas preguntas toma un valor de 0.94 y en otras 0.08.

2. **Tabla de Frecuencias**
    a. Se utiliza la función realizada con anterioridad para obtener la tabla de frecuencias.

```python
# Se crea la tabla de frecuencias para cada pregunta y se almacena en un csv
columnas_categoricas,tabla_frecuencia=tabla_de_frecuencia(df)
tabla_frecuencia.to_csv('tabladefrecuencia.csv')
```

Python

    b. El archivo que se descarga en formato csv es el siguiente; muestra la frecuencia de las respuestas binarias.

	2. La Organización Socio Formadora ¿ofreció retroalimentación sobre el desarrollo del Proyecto Solidario y tu desempeño?	Frequency	2. La Organización Socio Formadora ¿ofreció retroalimentación sobre el desarrollo del Proyecto Solidario y tu desempeño? (Mode)
0	Si	8874	Si
1	No	1823	Si

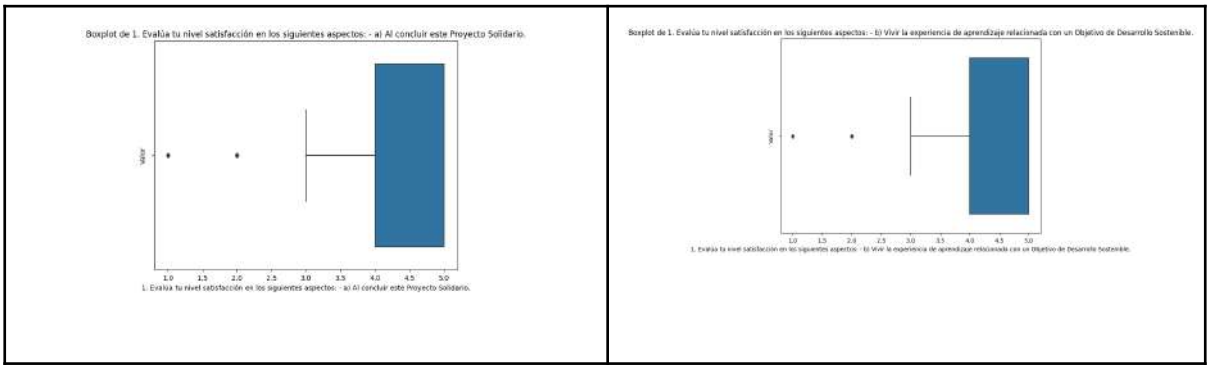
    c. Esta tabla muestra la frecuencia con la que el usuario, que responde a la encuesta, selecciona cada opción. Es posible observar que 8874 mencionaron que sí habían recibido retroalimentación, mientras que 1823 dijeron que no la habían recibido. Por lo tanto, aunque la moda es que sí la recibieron, las OSF's que no la recibieron podrían ser afectadas en la evaluación final.

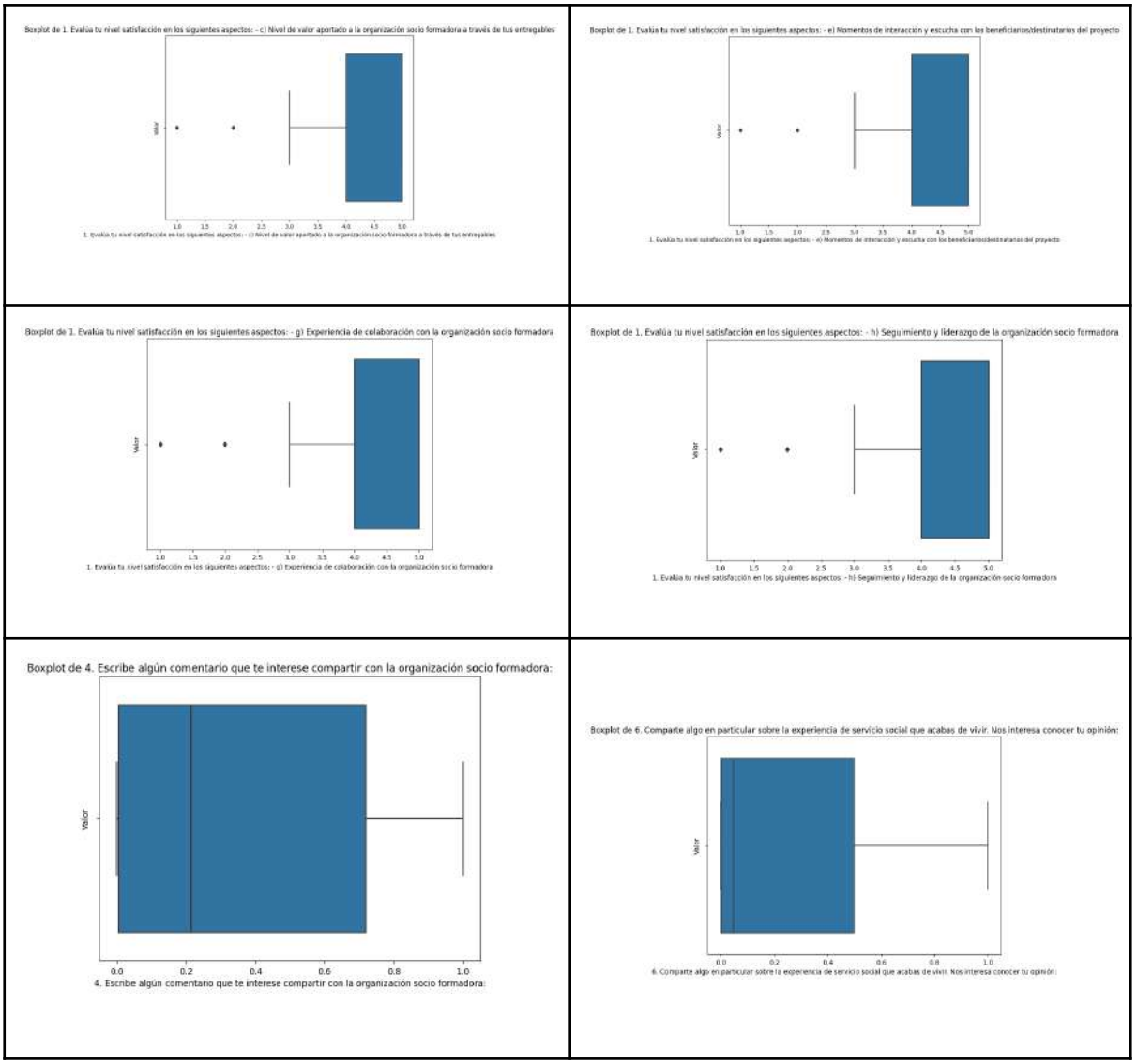
3. **Boxplots**
    a. Los diagramas de caja y bigotes son una forma gráfica de representar la distribución de un conjunto de datos numéricos. Estos gráficos son útiles para visualizar la mediana, los cuartiles, el rango intercuartílico y los valores atípicos de un conjunto de datos.

```python
boxplot_graf(df)
```

Python

    b. Algunos de las gráficas que sale al utilizar la función son:



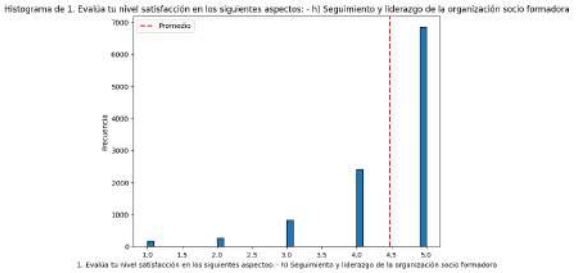


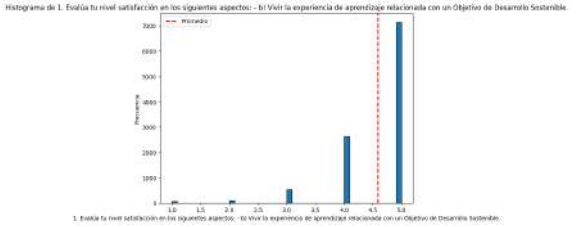
En las primeras 6 gráficas se muestran algunos valores atípicos, mientras que las últimas 2 son herramientas para comprender la manera en que están divididos los datos que resultan del análisis de sentimientos. El primero, que corresponde a la pregunta 4, tiene un promedio de 0.2 mientras que el segundo, que corresponde a la pregunta 6.1, tiene un promedio de 0.1.

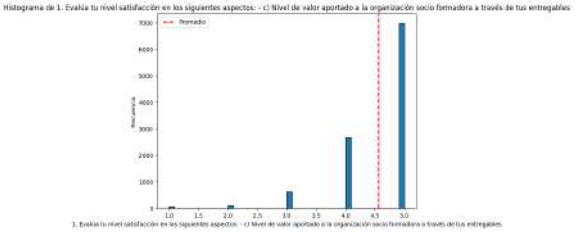
4. **Histogramas**
    a. Los histogramas se encargan de mostrar la frecuencia con la que ocurren los diferentes valores o rangos de valores en el conjunto de datos. Son útiles para analizar la forma de la distribución de los datos y para identificar patrones, tendencias y características importantes.

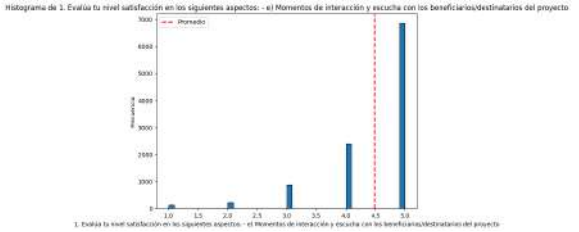
```python
histograma_graf(df)
```

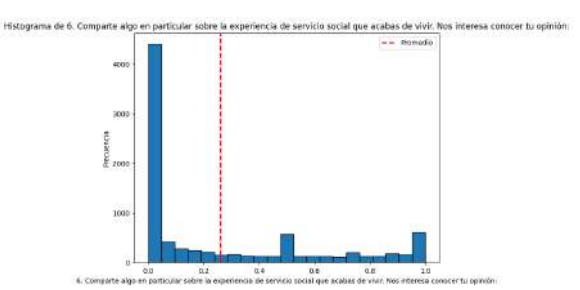
b. Algunos de los histogramas que se generan son:

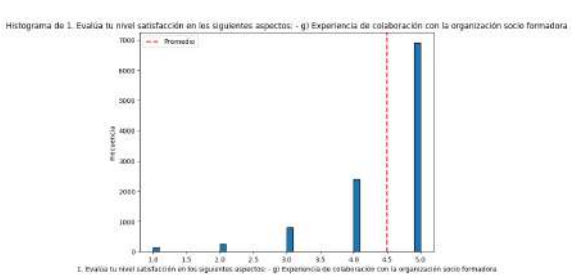


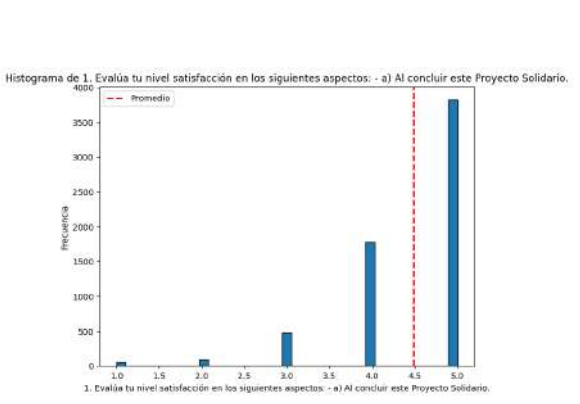


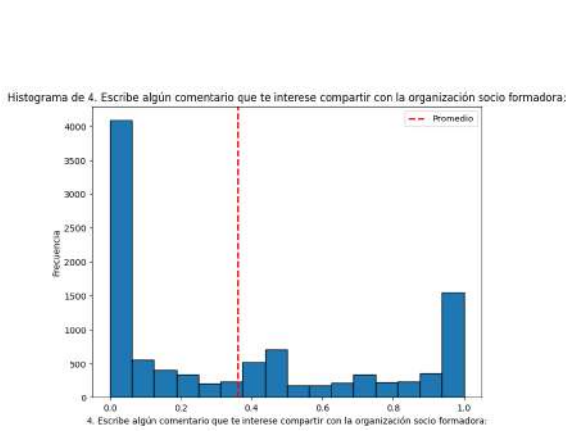










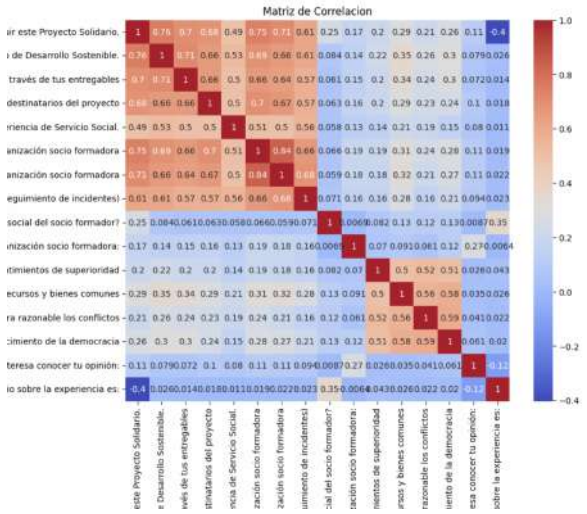


Como fue explicado anteriormente, el análisis de estas variables son importantes para comprender cómo se comportan los datos que tiene el dataset, así como ver como es necesario tratarlos en el futuro.

5. **Matriz de Correlación**
    a. La matriz de correlación se utiliza para analizar y visualizar las relaciones entre pares de variables en un conjunto de datos, esta permite identificar la fuerza y la dirección de las asociaciones entre las variables, lo que ayuda a comprender cómo se relacionan entre sí.

```
Analisis_de_correlacion(df)
```
Python

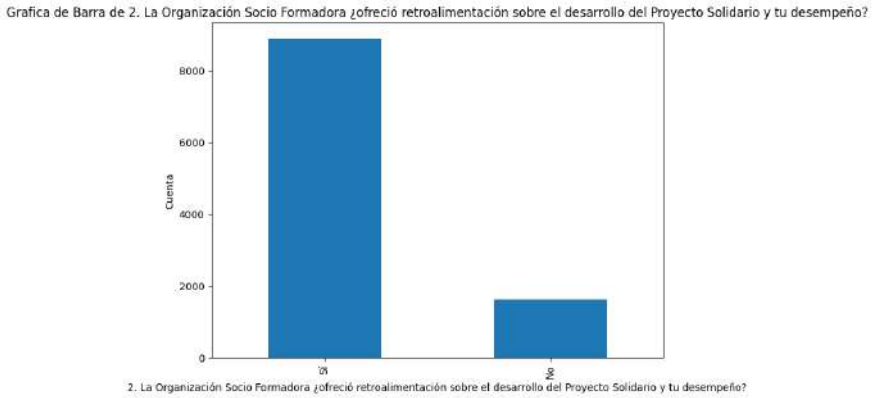
    b. El resultado del pedazo de código es la siguiente imagen, aunque los títulos de cada pregunta fueron recortados para mejorar la calidad. Cabe resaltar que los colores significan la fuerza de correlación entre las variables, las rojas tienen mucha y las blancas son las que tienen menos.



6. Gráficas de barras
    a. Por último, las gráficas de barras sirven para representar y comparar diferentes categorías o valores discretos en un conjunto de datos.

```
Grafica_barra(df)
```
Python

    b. La gráfica se hace con la única variable que responde a sí o no, y el resultado es:



c. Esto, muestra lo mismo que la tabla de frecuencias, la mayoría de las personas dijeron que si habían recibido retroalimentación y, las OSF's que no la dieron, podrían ser afectadas en su rendimiento y calificación.

2.3 Calidad de datos

En un inicio, los sets de datos proporcionados contaban con características que no hacían posible o recomendable su análisis, pues obtendríamos resultados posiblemente no válidos. En general, la calidad de los datos no era la óptima, pues se contaba con muchos valores faltantes, datos inconsistentes, y en general, tipos de datos que no permitían aplicar un modelo. Sin embargo, y como se puede observar en las gráficas mostradas y explicadas previamente, se cuenta con una serie de datos suficiente y de valor para, después de procesarlos, obtener buenos resultados y predicciones. Esto se puede comprobar mediante los estadísticos y gráficas obtenidas, y los resultados de la matriz de correlación.

3) Preparación de los datos

3.1 Selección de datos

Primero, es necesario determinar que el objetivo es crear un modelo capaz de predecir los valores que tomarán las entradas de texto, así como agrupar los comentarios y opiniones de los usuarios respecto a su experiencia formando parte de un proyecto de Servicio Social, y, para esto, es necesario convertir las variables antes mencionadas a variables cuantitativas para su correcto análisis, exceptuando las variables de *OSF* y *Fecha*. Se cambiaron los valores de ciertas columnas, incluyendo aquellas que contienen comentarios, mediante un modelo de análisis de sentimiento. Este modelo es un campo del

procesamiento del lenguaje que busca entender opiniones o cualquier tipo de comentario. Aplicado a este caso en específico, permite tener una percepción clara de las opiniones de los usuarios que contestaron las encuestas, otorgando un cierto valor numérico o puntuación que lo relaciona con un comentario positivo, negativo o neutro, es decir, determina su polaridad.

Con el objetivo de preparar los datos del análisis preliminar, se inició con un análisis comparativo de las columnas de los 5 dataframes. Al realizarlo, fue posible observar que todos las bases de datos tenían columnas diferentes, aunque parecidas, pues algunas cuentan con el mismo tipo de dato pero dentro de variables con nombres diferentes.. Tomando esto en cuenta, se decidió trabajar sobre ciertas variables específicas que estuviesen presentes en todos los datasets, para así concatenarlas y crear un nuevo data frame, con el que se trabajará desde ahora.

Específicamente, este nuevo set de datos cuenta con las las siguientes columnas, en este orden:

- OSF
- Fecha
- Evalúa tu nivel satisfacción en los siguientes aspectos: - a) Al concluir este Proyecto Solidario.
- Evalúa tu nivel satisfacción en los siguientes aspectos: - b) Vivir la experiencia de aprendizaje relacionada con un Objetivo de Desarrollo Sostenible.
- Evalúa tu nivel satisfacción en los siguientes aspectos: - c) Nivel de valor aportado a la organización socio formadora a través de tus entregables.
- Evalúa tu nivel satisfacción en los siguientes aspectos: - e) Momentos de interacción y escucha con los beneficiarios/destinatarios del proyecto.
- Evalúa tu nivel satisfacción en los siguientes aspectos: - f) Herramientas que aplicaste como las actividades, reportes, "quizzes", dentro de la plataforma de CANVAS en el desarrollo de la experiencia de Servicio Social.
- Evalúa tu nivel satisfacción en los siguientes aspectos: - g) Experiencia de colaboración con la organización socio formadora.
- Evalúa tu nivel satisfacción en los siguientes aspectos: - h) Seguimiento y liderazgo de la organización socio formadora.
- Evalúa tu nivel satisfacción en los siguientes aspectos: - i) Atención y servicio del área que administra el Servicio Social en el campus (asesoría y

orientación, información puntual, atención de dudas, seguimiento de incidentes).

- ¿Consideras interesante la causa social del socio formador?.
- Escribe algún comentario que te interese compartir con la organización socio formadora.
- Expresa tu opinión en los siguientes enunciados, en esta experiencia de servicio social tuve la oportunidad de: - a) Ser sensible ante la vulnerabilidad, el dolor y el sufrimiento del otro y actuar con el fin de eliminarlo, aliviarlo o evitarlo, a través de acciones justas alejadas de la pasión egoísta y/o de sentimientos de superioridad.
- Expresa tu opinión en los siguientes enunciados, en esta experiencia de servicio social tuve la oportunidad de: - b) Actuar con responsabilidad, con el fin de asegurar el bienestar de la colectividad, a través de acciones que garantizan el acceso a los derechos humanos, el empoderamiento de los ciudadanos y de las comunidades, así como el cuidado, mantenimiento y uso sostenible de los recursos y bienes comunes.
- Expresa tu opinión en los siguientes enunciados, en esta experiencia de servicio social tuve la oportunidad de: - c) Actuar con respeto ante la diversidad de género, sexual, étnica, cultural, de capacidades, generacional, religiosa y socioeconómica mostrando una cordial aceptación de las diferencias y la capacidad para gestionar de manera razonable los conflictos.
- Expresa tu opinión en los siguientes enunciados, en esta experiencia de servicio social tuve la oportunidad de: - e) Promover soluciones cooperativas en problemas o coordinar acciones colectivas con el fin de mejorar la calidad de vida de la sociedad, fomentando la cultura de la legalidad, los derechos humanos y/o el fortalecimiento de la democracia.
- Comparte algo en particular sobre la experiencia de servicio social que acabas de vivir. Nos interesa conocer tu opinión.
- Tu comentario sobre la experiencia es.

Como se podrá observar, los nombres de las variables de este nuevo dataframe no son exactamente las mismas que las presentes en los datasets originales, pues decidimos extraer las preguntas para que pasaran a ser los nuevos nombres de las columnas. Todo esto debido a la falta de información que ciertos nombres de columnas proporcionaban. Además, fue necesario renombrar la columna de *OSF,* pues no en todos los datasets estos datos se encontraban en un columna con dicho nombre.

Además, se creó una función que se encarga de reconocer las columnas que representan preguntas de opción múltiple. Lo anterior, porque las respuestas están en forma de texto y deben de ser cambiadas a puntuaciones numéricas del 1 al 5, al encontrar las celdas en donde aparecía '\n5 Muy Satisfecho' cambiarlas. Es importante mencionar que el nombre de estas columnas era guardado en una lista y, después, se ejecutó un loop con esos valores para hacer su cambio.

Más adelante, con el objetivo de representar las columnas de preguntas con respuestas binarias, se creó una función en donde se ingresan sus nombres mediante una lista y, al usar el método de pandas get_dummies(), se encargó de transformar cada columna a una binaria.

Para llenar los valores vacíos en cada celda, se decidió reemplazarlos utilizando el promedio de las encuestas respectivas a cada OSF, esto para poder contar con un número significativo de datos.

De una manera más específica, los pasos realizados fueron:

1. **Se importan las librerías a utilizar**

```python
import pandas as pd
from sentiment_analysis_spanish import sentiment_analysis
import seaborn as sns
import matplotlib.pyplot as plt
```

Python

2. **Se leen los archivos de Excel**
    a. A cada archivo se le otorga el nombre del periodo que le corresponde, y la lectura se hace a través de la función *pd.read_excel* de la librería de Pandas.

```python
#Leemos los archivos de excel

FJ22 = pd.read_excel('Estudiantes+PS+FJ22+TEC21_17+de+mayo+de+2023_12.17.xlsx')
FJ23 = pd.read_excel('Estudiantes+PS+FJ23_17+de+mayo+de+2023_12.28.xlsx')
INV23 = pd.read_excel('Estudiantes+PS_INV23+TEC21_17+de+mayo+de+2023_12.22.xlsx')
PS17 = pd.read_excel("Encuesta+a+Estudiantes+PS_17+de+mayo+de+2023_12.05.xlsx")
AD22 = pd.read_excel('Estudiantes+PS+AD22+TEC21_17+de+mayo+de+2023_12.07.xlsx')
```

Python

3. **Función para encontrar columnas diferentes**
    a. Se reciben dos datasets, se guardan las columnas de cada uno en dos variables diferentes y, con la función symetric_difference() se encuentran las columnas diferentes entre ellas. Al finalizar se imprime una lista. Se utilizan

en la limpieza para comprobar que, al concatenar los dataframes, no hayan columnas que difieran.

```python
def get_different_columns(df1, df2):
    columns1 = set(df1.columns)
    columns2 = set(df2.columns)
    different_columns = list(columns1.symmetric_difference(columns2))
    return different_columns
```

Python

4. **Función para eliminar valores específicos**
    a. Este código define una función llamada *remove_values_from_list* que toma dos argumentos: lst, que es una lista, y *values_to_remove*, que es otra lista que contiene los valores que se deben eliminar de lst. El propósito de esta función es eliminar los valores específicos de la lista lst y devolver la lista resultante.

```python
def remove_values_from_list(lst, values_to_remove):
    for value in values_to_remove:
        if value in lst:
            lst.remove(value)
    return lst
```

Python

5. **Eliminar la primera fila y concatenar**
    a. La función toma dos argumentos: *list_of_dfs*, que es una lista de pandas DataFrames, y *col_names*, que es una lista de nombres de columnas. El propósito de esta función es eliminar la primera fila de cada DataFrame en la lista y luego concatenar los DataFrames en uno solo. Además, se renombran las columnas del DataFrame resultante con los nombres especificados en la lista *col_names*.

```python
def drop_first_row_and_concatenate(list_of_dfs, col_names):
    concatenated_df = pd.DataFrame()  # Empty DataFrame for concatenation

    for df in list_of_dfs:
        df = df.drop(0).reset_index(drop=True)  # Drop first row and reset index
        concatenated_df = pd.concat([concatenated_df, df], ignore_index=True)  # Concatenate


    concatenated_df = concatenated_df.rename(columns=dict(zip(concatenated_df.columns, col_names)))

    return concatenated_df
```

Python

6. **Se encuentran las preguntas**
    a. Considerando que cada base de datos tiene las preguntas en posiciones diferentes, se hace una búsqueda manual y se guarda cada uno con el

nombre de su periodo respectivo. Así, se busca poder concatenar todos los datasets en uno grande.

```python
# Extraemos las preguntas de cada df para concaternarlas
FJ_22=FJ22.iloc[:, 15:32]
FJ_23=FJ23.iloc[:,19:36]
AD_22=AD22.iloc[:,19:36]
INV_23=INV23.iloc[:,19:36]
PS_17=PS17.iloc[:,16:33]
```

Python

7. **Se concatena el dataframe**
    a. Se declara una lista con los dataframes a concatenar, se obtienen los nombres de las columnas al extraer las preguntas y, finalmente, se utiliza la función mencionada anteriormente para concatenar los valores. Finalmente, se eliminan las últimas 3 columnas, pues eran vacías y no mostraban importancia para el análisis.

```python
listofdfs=[FJ_22,FJ_23,AD_22,INV_23,PS_17]
col_names = FJ_22.iloc[0, 0:17] #Extraemos las preguntas para que sean el nombre de las columnas

# Nuevo DF
df_preguntas = drop_first_row_and_concatenate(listofdfs, col_names) # Utilizamos la funcion para concaternar los df y crear un df
df_preguntas = df_preguntas.iloc[:, :-3] #Hacemos drop de las ultimas 3 preguintas
```

Python

8. **Se obtienen las columnas de la OSF**
    a. Como la posición de la columna de la Organización Socio Formadora es diferente para cada periodo, se obtiene el valor y se guarda en las variables que se presentan a continuación.

```python
FJ22_OSF =pd.DataFrame(FJ22['OSF'])
FJ23_OSF = pd.DataFrame(FJ23['Q3'])
AD22_OSF = pd.DataFrame( AD22['OSF'])
INV23_OSF =  pd.DataFrame(INV23['Q1'])
PS17_OSF =  pd.DataFrame(PS17['OSF'])
```

Python

9. **Se agrega la columna de OSF**
    a. Se agrega la columna de las Organizaciones Socio Formadoras correspondientes a los data frames y se concatenan, dejando una columna de solamente OSF.

```python
listofdfs=[FJ22_OSF,FJ23_OSF,AD22_OSF,INV23_OSF,PS17_OSF]
col_names = ["OSF"]

# Nuevo DF
OSF_row = drop_first_row_and_concatenate(listofdfs, col_names) # Concatenamos todos los nombres de las OSF
OSF_row = OSF_row.iloc[:, :-2] #Hacemos drop de las ultimas 3 preguintas
```

Python

10. Se definen las fechas

a. Se crean variables con el nombre del periodo que reciben la fecha en que el usuario terminó de contestar la encuesta.

```python
FJ22_fecha =pd.DataFrame(FJ22['EndDate'])
FJ23_fecha = pd.DataFrame(FJ23['EndDate'])
AD22_fecha = pd.DataFrame( AD22['EndDate'])
INV23_fecha =  pd.DataFrame(INV23['EndDate'])
PS17_fecha =  pd.DataFrame(PS17['EndDate'])
```

Python

11. **Se concatenan las fechas**

a. Utilizando el procedimiento anterior en donde se habían unificado, se concatenan las fechas en una columna titulada “Fecha”. Se deja solo esa columna.

```python
listofdfs=[FJ22_fecha,FJ23_fecha,AD22_fecha,INV23_fecha,PS17_fecha]
col_names = ["Fecha"]

# Nuevo DF
df_fechas = drop_first_row_and_concatenate(listofdfs, col_names) # Concatenamos todas las fechas
```

Python

12. El Data Frame queda listo

a. La base de datos nueva que se utilizará se guarda dentro de la variable *complete_df* y consiste de la OSF, la fecha y las preguntas que el encuestado realizó.

```python
complete_df = pd.concat([OSF_row,df_fechas, df_preguntas], axis=1)  # Concatenar las columnas nuevas
```

Python

Después, la transformación de los datos consistió en definir diferentes funciones, entre ellas se encuentran:

1. **Encontrar columnas de opción múltiple**

a. Consiste en identificar las columnas que tienen el texto “5 Totalmente de Acuerdo” o “5 Muy Satisfecho” para después hacer una lista. El objetivo es tener claras las columnas de opción múltiple, para tratarlas después.

```python
#Generar lista para cambiar columnas multiple choice
def encontrar_columnas_multiple_choice(df):
    target_texts = ['\n5 Totalmente de Acuerdo', '\n5 Muy Satisfecho']
    columns_with_text = []

    for column_name in df.columns:
        if df[column_name].astype(str).str.contains('|'.join(target_texts), na=False).any():
            columns_with_text.append(column_name)

    return columns_with_text
```
Python

2. **Convertir a numérico**
    a. Consiste en utilizar la función *pd.to_numeric* para convertir cada valor de las columnas a numérico.

```python
#funcion para convertir columna del tipo objeto a numerica
def convertir_numerico(df, column_list):
    for column in column_list:
        df[column] = pd.to_numeric(df[column], errors='coerce')
```
Python

3. Transformar valores de opción múltiple
    a. Con el objetivo de eliminar el texto y el formato de string de los valores, se utiliza la función map(). De tal manera, las de opción múltiple toman un valor numérico que se puede analizar.

```python
#cambiar los valores de las columnas multiple choice
def transformar_valores_multiple_choice(df, column_names):
    for column_name in column_names:
        df[column_name] = df[column_name].map({
            '\n5 Muy Satisfecho': 5,
            '4': 4,
            '3':3,
            '2':2,
            '\n1 Nada Satisfecho':1

        }).fillna(df[column_name])

    convertir_numerico(df, column_names)

    return df
```
Python

4. **Análisis de Sentimiento**
    a. Se utiliza una librería de Python para realizar el análisis de sentimiento en español.

```python
#convertir comentarios a numerico con sentiment analysis model
def analisis_de_sentimiento(df, column_names):
    # Load the sentiment analysis model
    model = sentiment_analysis.SentimentAnalysisSpanish()

    for column_name in column_names:
        for index, row in df.iterrows():
            text = row[column_name]
            if pd.notnull(text):  # Skip NaN values
                sentiment = model.sentiment(text)
                df.at[index, column_name] = sentiment

    convertir_numerico(df, column_names)

    return df
```

Python

5. **Calcular estadísticos**
    a. Se calcula la media, mediana, máximo, mínimo, varianza y desviación estándar. Al final, crea una tabla con los valores del Data Frame y con los estadísticos.

```python
# Funcion que calcula el promedio, la mediana, el valor maximo, el valor minimo, la varianza y la desviación estandar de del df
def calculate_statistics(df):
    numericas = df.select_dtypes(include='number').columns

    statistics = pd.DataFrame()

    for column in numericas:
        column_stats = df[column].agg(['mean', 'median', 'max', 'min', 'var', 'std'])
        statistics = pd.concat([statistics, column_stats], axis=1)

    return statistics.T
```

Python

6. **Tabla de Frecuencia**
    a. Se hace una función que hace una tabla de frecuencia para cada pregunta del data frame, esta incluye la cuenta de ocurrencias de cada valor en las columnas categóricas de la base de datos, así como la moda de cada columna.

```python
# Se crea la tabla de frecuencias para cada pregunta del df
def tabla_de_frecuencia(df):
    columnas_categoricas = df.select_dtypes(exclude='number').columns
    columnas_categoricas = [col for col in columnas_categoricas if col not in ['OSF', 'Fecha']]

    tabla_de_frecuencia_moda = pd.DataFrame()

    for column in columnas_categoricas:
        if column != 'OSF':
            frecuencia_columna = df[column].value_counts().reset_index()
            frecuencia_columna.columns = [column, 'Frequency']
            moda_columna = df[column].mode().values.tolist()

            tabla_de_frecuencia_moda = pd.concat([tabla_de_frecuencia_moda, frecuencia_columna], axis=1)
            tabla_de_frecuencia_moda[column + ' (Mode)'] = moda_columna[0] if moda_columna else None

    return columnas_categoricas,tabla_de_frecuencia_moda
```
Python

7. **Box Plots**
    a. Esta función consiste en graficar un boxplot para cada pregunta en la base de datos.

```python
# Crea gráficas de bigote para cada pregunta con valores numericos
def boxplot_graf(df):
    numericas = df.select_dtypes(include='number').columns

    for column in numericas:
        plt.figure(figsize=(8, 6))
        sns.boxplot(x=df[column])
        plt.title(f"Boxplot de {column}")
        plt.xlabel(column)
        plt.ylabel('Valor')
        plt.show()
```
Python

8. **Histogramas**
    a. Se crea un histograma para cada pregunta en la base de datos, se grafica también el promedio que se representa con una línea punteada roja.

```python
# Crea histograma para cada pregunta con valores numericos
def histograma_graf(df):
    numericas = df.select_dtypes(include='number').columns

    for column in numericas:
        plt.figure(figsize=(8, 6))
        plt.hist(df[column], bins='auto', edgecolor='black')
        plt.axvline(df[column].mean(), color='red', linestyle='dashed', linewidth=2)
        plt.title(f"Histograma de {column}")
        plt.xlabel(column)
        plt.ylabel("Frecuencia")
        plt.legend(["Promedio"])
        plt.show()
```
Python

9. **Análisis de Correlación**
    a. Se crea una matriz de correlación para las variables numéricas. Este paso es esencial para comprender y analizar las relaciones lineales entre variables en un conjunto de datos.

```python
# Genera matriz de correlacion para las variables numericas
def Analisis_de_correlacion(df):
    numericas = df.select_dtypes(include='number').columns
    matriz_de_correlacion = df[numericas].corr()
    #print(matriz_de_correlacion)
    plt.figure(figsize=(10, 8))
    sns.heatmap(matriz_de_correlacion, annot=True, cmap='coolwarm')
    plt.title("Matriz de Correlacion")
    plt.show()
```

Python

10. Gráfica de Barras

a. Se crean gráficas de barras para las preguntas con valores categóricos.

```python
# Crea gráficas de barras para cada pregunta con valores categóricas
def Grafica_barra(df):
    categoricas = df.select_dtypes(include='object').columns
    categoricas = [col for col in categoricas if col not in ['OSF', 'Fecha']]
    for column in categoricas:
        plt.figure(figsize=(8, 6))
        df[column].value_counts().plot(kind='bar')
        plt.title(f"Grafica de Barra de {column}")
        plt.xlabel(column)
        plt.ylabel("Cuenta")
        plt.show()
```

Python

11. Gráfica Circular

a. Se crean gráficas pie para los valores categóricos

```python
# Crea gráficas de pay para cada pregunta con valores categóricas
def grafica_pie(df):
    categoricas = df.select_dtypes(include='object').columns
    categoricas = [col for col in categoricas if col not in ['OSF', 'Fecha']]

    for column in categoricas:
        plt.figure(figsize=(8, 6))
        counts = df[column].value_counts()
        plt.pie(counts, labels=counts.index, autopct='%1.1f%%', startangle=90)
        plt.title(f"Grafica Pie  {column}")
        plt.show()
```

Python

12. One hot encoding

a. Se realiza el one hot encoding, una técnica utilizada en el procesamiento de datos para representar variables categóricas o de tipo nominal en una forma numérica. Se encarga de convertir cada categoría en una nueva columna binaria y asigna un valor de 1 o 0 para indicar si la observación pertenece o no a esa categoría. Aplica para que las columnas categóricas pasen a binario.

```python
# Convierte las columnas categorcas en formato binario
def one_hot_encode(df):
    columnas_categoricas = df.select_dtypes(include='object').columns
    columnas_categoricas = [col for col in columnas_categoricas if col not in ['OSF', 'Fecha']]
    print(columnas_categoricas)
    df = pd.get_dummies(df, columns=columnas_categoricas)

    df = df.replace({True: 1, False: 0})

    return df
```
Python

13. Reemplazo de NAN's

a. Se cambian los valores nulos por el promedio de la columna.

```python
# Remplaza los valores nulos con el promedio de la encuesta por cada OSF
def replace_nan_with_average(df, columnas, OSF):
    for column in columnas:
        average_by_osf = df.groupby(OSF)[column].transform('mean')
        df[column].fillna(average_by_osf, inplace=True)

    return df
```
Python

14. Columnas NAN

a. La función se encarga de tomar a un data frame como entrada y da, como salida, una lista de las columnas que contienen valores nulos.

```python
# Verificar cuantos valores nan hay en las columnas
def columnas_nan(df):
    nan_columnas = []

    for column in df.columns:
        nan_count = df[column].isna().sum()
        if nan_count > 0:
            nan_columnas.append(column)

    return nan_columnas
```

15. Imprimir los valores nulos

a. La función toma un Data Frame como entrada e imprime las columnas que contienen valores NaN, así como la cantidad de valores NaN en cada una de esas columnas.

```python
# Imprime las columnas que tiene valores nulos y cuanta cuantos tiene
def print_nan_values(df):
    nan_columnas = columnas_nan(df)
    print('Las columnas con NaN values son las siguientes:')
    for column in nan_columnas:
        nan_count = df[column].isna().sum()
        print(f"La columna '{column}': {nan_count} NaN values")
        print('_____')

    return nan_columnas
```
Python

Después, se aplican las funciones de la siguiente manera:

1. **Análisis de sentimientos**
    a. Llama a la función de opción múltiple y cambia los datos no numéricos a numéricos, después, aplica el análisis de sentimiento a las preguntas 4 y 6, que son las encargadas de recibir input del usuario.

```python
# Encuentra las columnsas multiple choise y cmabia algunos datos que no eran numericos a numericos
lista_multiple_choice=encontrar_columnas_multiple_choice(complete_df)
df=transformar_valores_multiple_choice(complete_df, lista_multiple_choice)

# Análisis de sentimientos Para la pregunta 4 y pregunta 6 las cuales tienen texto escrito por los alumnos
list_columnas_analisis_sentimiento=['4. Escribe algún comentario que te interese compartir con la organización socio formadora:','
df=analisis_de_sentimiento(df, list_columnas_analisis_sentimiento)
```

Python

3.2 Limpieza de datos

1. **One hot encode**
    a) Se utiliza la función one_hot_encode() definida previamente para representar las variables categóricas del dataset en formato binario. Con esto, se tendrán entradas de 0 y 1. Para este caso, la única columna clasificada como categórica y la que puede sufrir esta modificación es: ‘La Organización Socio Formadora ¿ofreció retroalimentación sobre el desarrollo del Proyecto Solidario y tu desempeño?’, pasando de tener entradas de Sí/No a 1/0.

```python
# representamos las variables categórcas en formato binario
df= one_hot_encode(df)
```

Python

2. **Valores nan y reemplazo**
    a) Se utilizan las función columnas_nan() para extraer el número de valores NaN que hay en las columnas. Esto permitirá reemplazar estos espacios con los promedios obtenidos para cada encuesta, lo que se lleva a cabo gracias a la función replace_nan_with_average().

```python
# Se buscan los valors nulos y se remplazan por promedios de las encuestas
columnas_con_nan_values=columnas_nan(df)
columnas_con_nan_values = columnas_con_nan_values[1:]
df = replace_nan_with_average(df, columnas_con_nan_values, 'OSF')
```

Python

3. **Eliminación de columnas sin *OSF***
    a) Si existen filas que no tienen registrado un nombre de OSF, se eliminan, pues el conocer este dato es esencial para el análisis. Mediante el método *df.dropna(subset=[“OSF”])* se eliminan los valores nulos existentes sólo en la columna *OSF.*

```python
# Se borran las encuestas que no tienen registrada la OSF en la encuesta
df = df.dropna(subset=['OSF'])
df=df.dropna()
```

4. **Print NaN values**
    a) Se aplica dicha función que toma como entrada el data frame modificado previamente, para comprobar si aún siguen existiendo o no valores nulos. En ese caso, se desplegarán aquellas columnas que tengan estos valores y la cantidad.

```python
# Se despliegan los valores nulos
nan=print_nan_values(df)
```

3.3 Transformación de datos

1. **Encontrar columnas de opción múltiple**
    a. Consiste en identificar las columnas que tienen el texto “5 Totalmente de Acuerdo” o “5 Muy Satisfecho” para después hacer una lista. El objetivo es tener claras las columnas de opción múltiple, para tratarlas después.

```python
#Generar lista para cambiar columnas multiple choice
def encontrar_columnas_multiple_choice(df):
    target_texts = ['\n5 Totalmente de Acuerdo', '\n5 Muy Satisfecho']
    columns_with_text = []

    for column_name in df.columns:
        if df[column_name].astype(str).str.contains('|'.join(target_texts), na=False).any():
            columns_with_text.append(column_name)

    return columns_with_text
```

2. **Convertir a numérico**
    a. Consiste en utilizar la función *pd.to_numeric* para convertir cada valor de las columnas a numérico.

```python
#funcion para convertir columna del tipo objeto a numerica
def convertir_numerico(df, column_list):
    for column in column_list:
        df[column] = pd.to_numeric(df[column], errors='coerce')
```

3. **Transformar valores de opción múltiple**
    a. Con el objetivo de eliminar el texto y el formato de string de los valores, se utiliza la función map(). De tal manera, las de opción múltiple toman un valor numérico que se puede analizar.

```python
#cambiar los valores de las columnas multiple choice
def transformar_valores_multiple_choice(df, column_names):
    for column_name in column_names:
        df[column_name] = df[column_name].map({
            '\n5 Muy Satisfecho': 5,
            '4': 4,
            '3':3,
            '2':2,
            '\n1 Nada Satisfecho':1

        }).fillna(df[column_name])

    convertir_numerico(df, column_names)

    return df
```

Python

4. **Análisis de Sentimiento**
    a. Se utiliza una librería de Python para realizar el análisis de sentimiento en español.

```python
#convertir comentarios a numerico con sentiment analysis model
def analisis_de_sentimiento(df, column_names):
    # Load the sentiment analysis model
    model = sentiment_analysis.SentimentAnalysisSpanish()

    for column_name in column_names:
        for index, row in df.iterrows():
            text = row[column_name]
            if pd.notnull(text):  # Skip NaN values
                sentiment = model.sentiment(text)
                df.at[index, column_name] = sentiment

    convertir_numerico(df, column_names)

    return df
```

Python

5. **Calcular estadísticos**
    a. Se calcula la media, mediana, máximo, mínimo, varianza y desviación estándar. Al final, crea una tabla con los valores del Data Frame y con los estadísticos.

```python
# Funcion que calcula el promedio, la mediana, el valor maximo, el valor minimo, la varianza y la desviación estandar de del df
def calculate_statistics(df):
    numericas = df.select_dtypes(include='number').columns

    statistics = pd.DataFrame()

    for column in numericas:
        column_stats = df[column].agg(['mean', 'median', 'max', 'min', 'var', 'std'])
        statistics = pd.concat([statistics, column_stats], axis=1)

    return statistics.T
```

Python

6. **Tabla de Frecuencia**
    a. Se hace una función que hace una tabla de frecuencia para cada pregunta del data frame, esta incluye la cuenta de ocurrencias de cada valor en las

columnas categóricas de la base de datos, así como la moda de cada columna.

```python
# Se crea la tabla de frecuencias para cada pregunta del df
def tabla_de_frecuencia(df):
    columnas_categoricas = df.select_dtypes(exclude='number').columns
    columnas_categoricas = [col for col in columnas_categoricas if col not in ['OSF', 'Fecha']]

    tabla_de_frecuencia_moda = pd.DataFrame()

    for column in columnas_categoricas:
        if column != 'OSF':
            frecuencia_columna = df[column].value_counts().reset_index()
            frecuencia_columna.columns = [column, 'Frequency']
            moda_columna = df[column].mode().values.tolist()

            tabla_de_frecuencia_moda = pd.concat([tabla_de_frecuencia_moda, frecuencia_columna], axis=1)
            tabla_de_frecuencia_moda[column + ' (Mode)'] = moda_columna[0] if moda_columna else None

    return columnas_categoricas,tabla_de_frecuencia_moda
```

Python

7. **Box Plots**
    a. Esta función consiste en graficar un boxplot para cada pregunta en la base de datos.

```python
# Crea gráficas de bigote para cada pregunta con valores numericos
def boxplot_graf(df):
    numericas = df.select_dtypes(include='number').columns

    for column in numericas:
        plt.figure(figsize=(8, 6))
        sns.boxplot(x=df[column])
        plt.title(f"Boxplot de {column}")
        plt.xlabel(column)
        plt.ylabel('Valor')
        plt.show()
```

Python

8. **Histogramas**
    a. Se crea un histograma para cada pregunta en la base de datos, se grafica también el promedio que se representa con una línea punteada roja.

```python
# Crea histograma para cada pregunta con valores numericos
def histograma_graf(df):
    numericas = df.select_dtypes(include='number').columns

    for column in numericas:
        plt.figure(figsize=(8, 6))
        plt.hist(df[column], bins='auto', edgecolor='black')
        plt.axvline(df[column].mean(), color='red', linestyle='dashed', linewidth=2)
        plt.title(f"Histograma de {column}")
        plt.xlabel(column)
        plt.ylabel("Frecuencia")
        plt.legend(["Promedio"])
        plt.show()
```

Python

9. **Análisis de Correlación**

a. Se crea una matriz de correlación para las variables numéricas. Este paso es esencial para comprender y analizar las relaciones lineales entre variables en un conjunto de datos.

```python
# Genera matriz de correlacion para las variables numericas
def Analisis_de_correlacion(df):
    numericas = df.select_dtypes(include='number').columns
    matriz_de_correlacion = df[numericas].corr()
    #print(matriz_de_correlacion)
    plt.figure(figsize=(10, 8))
    sns.heatmap(matriz_de_correlacion, annot=True, cmap='coolwarm')
    plt.title("Matriz de Correlacion")
    plt.show()
```

Python

10. Gráfica de Barras

a. Se crean gráficas de barras para las preguntas con valores categóricos.

```python
# Crea gráficas de barras para cada pregunta con valores categóricas
def Grafica_barra(df):
    categoricas = df.select_dtypes(include='object').columns
    categoricas = [col for col in categoricas if col not in ['OSF', 'Fecha']]
    for column in categoricas:
        plt.figure(figsize=(8, 6))
        df[column].value_counts().plot(kind='bar')
        plt.title(f"Grafica de Barra de {column}")
        plt.xlabel(column)
        plt.ylabel("Cuenta")
        plt.show()
```

Python

11. Gráfica Circular

a. Se crean gráficas pie para los valores categóricos

```python
# Crea gráficas de pay para cada pregunta con valores categóricas
def grafica_pie(df):
    categoricas = df.select_dtypes(include='object').columns
    categoricas = [col for col in categoricas if col not in ['OSF', 'Fecha']]

    for column in categoricas:
        plt.figure(figsize=(8, 6))
        counts = df[column].value_counts()
        plt.pie(counts, labels=counts.index, autopct='%1.1f%%', startangle=90)
        plt.title(f"Grafica Pie  {column}")
        plt.show()
```

Python

12. One hot encoding

a. Se realiza el one hot encoding, una técnica utilizada en el procesamiento de datos para representar variables categóricas o de tipo nominal en una forma numérica. Se encarga de convertir cada categoría en una nueva columna binaria y asigna un valor de 1 o 0 para indicar si la observación pertenece o no a esa categoría. Aplica para que las columnas categóricas pasen a binario.

```python
# Convierte las columnas categorcas en formato binario
def one_hot_encode(df):
    columnas_categoricas = df.select_dtypes(include='object').columns
    columnas_categoricas = [col for col in columnas_categoricas if col not in ['OSF', 'Fecha']]
    print(columnas_categoricas)
    df = pd.get_dummies(df, columns=columnas_categoricas)

    df = df.replace({True: 1, False: 0})

    return df
```
Python

13. Reemplazo de NAN's

a. Se cambian los valores nulos por el promedio de la columna.

```python
# Remplaza los valores nulos con el promedio de la encuesta por cada OSF
def replace_nan_with_average(df, columnas, OSF):
    for column in columnas:
        average_by_osf = df.groupby(OSF)[column].transform('mean')
        df[column].fillna(average_by_osf, inplace=True)

    return df
```
Python

14. Columnas NAN

a. La función se encarga de tomar a un data frame como entrada y da, como salida, una lista de las columnas que contienen valores nulos.

```python
# Verificar cuantos valores nan hay en las columnas
def columnas_nan(df):
    nan_columnas = []

    for column in df.columns:
        nan_count = df[column].isna().sum()
        if nan_count > 0:
            nan_columnas.append(column)

    return nan_columnas
```

15. Imprimir los valores nulos

a. La función toma un Data Frame como entrada e imprime las columnas que contienen valores NaN, así como la cantidad de valores NaN en cada una de esas columnas.

```python
# Imprime las columnas que tiene valores nulos y cuanta cuantos tiene
def print_nan_values(df):
    nan_columnas = columnas_nan(df)
    print('Las columnas con NaN values son las siguientes:')
    for column in nan_columnas:
        nan_count = df[column].isna().sum()
        print(f"La columna '{column}': {nan_count} NaN values")
        print('_____')

    return nan_columnas
```
Python

4) Aplicación de Técnicas de Modelación

4.1 Extracción de características

Durante el proceso de transformación de datos, se consideró que todas las columnas del dataset son relevantes para el informe y análisis de los datos, por lo cual se decidió transformar las características de las variables presentes en el dataset original. Esto se logró a través de múltiples funciones que se aplicaron a diferentes tipos de datos.

A continuación están las funciones que se utilizaron para transformar las características de los datos.

1. **Encontrar columnas de opción múltiple**
    a. Consiste en identificar las columnas que tienen el texto “5 Totalmente de Acuerdo” o “5 Muy Satisfecho” para después hacer una lista. El objetivo es tener claras las columnas de opción múltiple, para tratarlas después.

```python
#Generar lista para cambiar columnas multiple choice
def encontrar_columnas_multiple_choice(df):
    target_texts = ['\n5 Totalmente de Acuerdo', '\n5 Muy Satisfecho']
    columns_with_text = []

    for column_name in df.columns:
        if df[column_name].astype(str).str.contains('|'.join(target_texts), na=False).any():
            columns_with_text.append(column_name)

    return columns_with_text
```
Python

2. **Convertir a numérico**
    a. Consiste en utilizar la función *pd.to_numeric* para convertir cada valor de las columnas a numérico.

```python
#funcion para convertir columna del tipo objeto a numerica
def convertir_numerico(df, column_list):
    for column in column_list:
        df[column] = pd.to_numeric(df[column], errors='coerce')
```
Python

3. **Transformar valores de opción múltiple**
    a. Con el objetivo de eliminar el texto y el formato de string de los valores, se utiliza la función map(). De tal manera, las de opción múltiple toman un valor numérico que se puede analizar.

```python
#cambiar los valores de las columnas multiple choice
def transformar_valores_multiple_choice(df, column_names):
    for column_name in column_names:
        df[column_name] = df[column_name].map({
            '\n5 Muy Satisfecho': 5,
            '4': 4,
            '3':3,
            '2':2,
            '\n1 Nada Satisfecho':1

        }).fillna(df[column_name])

    convertir_numerico(df, column_names)

    return df
```
Python

4. **Análisis de Sentimiento**
    a. Se utiliza una librería de Python para realizar el análisis de sentimiento en español.

```python
#convertir comentarios a numerico con sentiment analysis model
def analisis_de_sentimiento(df, column_names):
    # Load the sentiment analysis model
    model = sentiment_analysis.SentimentAnalysisSpanish()

    for column_name in column_names:
        for index, row in df.iterrows():
            text = row[column_name]
            if pd.notnull(text):  # Skip NaN values
                sentiment = model.sentiment(text)
                df.at[index, column_name] = sentiment

    convertir_numerico(df, column_names)

    return df
```
Python

Gracias a estas transformaciones de características, la encuesta tiene una mayor precisión en cuanto a su valor y postura ante cada OSF, permitiendo que se pueda analizar más información a través de los distintos métodos de modelación predictiva que se realizan.

4.2 Explicación de diferentes modelos

1. **K-Means:** Es un algoritmo de clustering que, basado en su similitud, se encarga de agrupar un conjunto de datos. Funciona de una manera iterativa, se encarga de asignar puntos al cluster más cercano según la distancia, luego los centros de los clusters se actualizan calculando el promedio de las coordenadas de los puntos asignados. Este proceso se repite hasta que los centros convergen y se encuentra la mejor división de los datos.
   Modificar los parámetros de este modelo puede conllevar a obtener ligeras variaciones en los resultados. Específicamente, los parámetros modificados fueron:

a. **n_estimators:** Este parámetro indica el número de clusters que se desean crear en el modelo de grupos similares de datos.
b. **max_iter:** Determina el número máximo de iteraciones que el modelo hará el cálculo de centroides y reorganizará los datos, si no hay mejora en los resultados el modelo regresará a la iteración anterior.
c. **init:** Este parámetro determina el método que se utiliza para inicializar los centroides de los clusters. El valor default es 'kmeans++' pero también se puede escoger la versión de 'random'.
d. **tol:** Este parámetro da el valor de tolerancia para mejora entre más pequeño es este valor se pueden esperar resultados más precisos.
e. **random_state:** Este parámetro crea una semilla aleatoria para la reproducción de los datos.

2. **Agrupamiento Aglomerado:** El agrupamiento aglomerativo es un algoritmo de clustering que construye jerarquías de clústeres de manera incremental. Se inicia considerando a cada punto de dato como un cluster individual y luego fusiona los más similares en cada iteración, basándose en una medida de similitud. Este proceso se repite hasta obtener una jerarquía completa de clusters. Cabe destacar que esto se muestra esencial pues no requiere especificar el número de clusters al inicio, así como flexibilidad al permitir explorar diferentes niveles dentro de esta jerarquía.

   Modificar los parámetros de este modelo puede conllevar a obtener ligeras variaciones en los resultados. Específicamente, los parámetros modificados fueron:

   a. **n_estimators:** Este parámetro indica el número de clusters que se desean crear en el proceso de clustering jerárquico.

3. **Random Forest Regressor:** Este es un algoritmo de aprendizaje automático que combina múltiples árboles de decisión para realizar predicciones en problemas de regresión. Cada árbol se entrena de forma individual en una muestra aleatoria del conjunto de datos y, posteriormente, se promedian las predicciones de todos los árboles para obtener la predicción final. Así, se proporciona una mayor precisión y estabilidad en las predicciones. Puede lidiar con datos de alta dimensionalidad y evita el sobreajuste, por lo que es utilizado en diversos campos.

Modificar los parámetros de este modelo puede conllevar a obtener ligeras variaciones en los resultados. Específicamente, los parámetros modificados fueron:

a. **test_size:** indica la proporción de datos que será utilizado del total para formar el conjunto de prueba, es decir, para medir el rendimiento. El resto de los datos será utilizado para el entrenamiento del modelo. Normalmente se establece una relación 80% - 20%, o 70% - 30%, aunque esto dependerá de cada específico.
b. **random_state:** este parámetro establece una semilla que se utiliza en modelos de aprendizaje automático como número inicial para la generación de número aleatorios. Si no se establece un valor específico y constante, la generación de valores aleatorios durante la ejecución del modelo podría afectar los resultados. Aunque su valor por default es *None,* puede tomar cualquier valor entero.
c. **n_estimators:** es un hiperparámetro de la clase RandomForestRegressor() que implementa dicho algoritmo, y especifica la cantidad de árboles que se deben utilizar en el modelo. Su valor por default es 100, pero entre más grande sea, mejor, pues aumenta la capacidad del modelo, sin embargo, también se podrían notar repercusiones en el tiempo de ejecución.
d. **max_depth:** con valor por default *None*, establece la profundidad máxima permitida para los árboles, es decir, su cantidad máxima de niveles. Si este valor aumenta, más complejo será el modelo, lo que podría permitir obtener mejores resultados.
e. **min_sample_split:** aquí se establece el número mínimo requerido de muestras para poder dividir un nodo durante el desarrollo del algoritmo. Si el valor es alto, se tendrá un árbol con menos nodos hijos, sin embargo, si este es bajo, su complejidad aumentará. Su valor predefinido es 2, pero este se puede ajustar en base a la complejidad que se busque.
f. **max_features:** su valor default es 1, e indica el número límite de variables predictoras que se considerarán al dividir cada uno de los nodos. Esto ayuda a que durante el desarrollo del árbol se exploren diferentes combinaciones de variables.

4. **Support Vector Regresor:** Es un algoritmo de aprendizaje automático que se basa en la idea de los vectores de soporte y busca encontrar una función de regresión que se ajuste de una manera óptima a los datos. En lugar de minimizar el error en todos los puntos de los datos, como lo es en la regresión lineal tradicional, el SVR

establece un margen alrededor de la función y se enfoca en minimizar las violaciones de este margen. Se utiliza una función de kernel para mapear los datos en un espacio de mayor dimensión, donde se puede encontrar un hiperplano que se ajusta correctamente.

Modificar los parámetros de este modelo puede conllevar a obtener ligeras variaciones en los resultados. Específicamente, los parámetros modificados fueron:

a. **kernel:** Este parámetro se utiliza para definir la función kernel que se utilizara en el modelo, el propósito de esta función es transformar los datos a un plano de mayor dimensión donde se pueda encontrar una mejor separación lineal entre clases, las posibles opciones a escoger es: Linear, Polynomial,RBF, y Sigmoide.
b. **C:** Este parámetro se enfoca en la fuerza del tradeoff entre conseguir el error mínimo de entrenamiento y un margen mayor. Entre mas pequeño es este parámetro se entrega un mayor margen de error.
c. **epsilon:** Este parámetro define el margen de tolerancia para errores en la estimación del regresor.

5. **Red Neuronal:** Es un modelo de aprendizaje automático inspirado en el funcionamiento del cerebro humano, consiste en un conjunto de nodos llamados neuronas artificiales, que se encargan de procesar y transmitir la información. Cada neurona toma una combinación lineal de las entradas y las transforma mediante una función de activación, finalmente, las envía como salida a otras neuronas. Las conexiones entre sí tienen asociados pesos que determinan la importancia relativa de cada entrada. A través del entrenamiento, la red neuronal ajusta automáticamente los pesos, facilitando el reconocimiento de patrones.

   Modificar los parámetros de este modelo puede conllevar a obtener ligeras variaciones en los resultados. Específicamente, los parámetros modificados fueron:

   a. **Función de Activación:** Este parámetro es utilizado para permitir que la red neuronal aprende y representa relaciones no lineales entre las preguntas de la encuesta y la salida deseada. Las opciones utilizados es nuestro modelo fueron: Sigmoidal,Tangente Hiperbólica,Exponential Linear Unit,softmax,y Relu

4.3 Metodología para el Entrenamiento y Prueba

Ahora bien, ya que se tienen los datos estructurados en un dataframe se alistarán para ser ingresados a los modelos de ML. ***Para los modelos de clustering,*** la base de datos solamente tiene que estar limpia y sin los valores no numéricos e irrelevantes, en este caso en particular se quitan los valores como el *nombre de la OSF, la fecha de subida la encuesta, la duración de contestado y los valores de los comentarios*. Lo anterior se hace por medio de la función “drop”

En la otra mano, ***para los modelos de predicción***, se hace la limpieza de los datos de igual forma que con los modelos de clasificación, por consiguiente se alistan los datos separando los datos en 2 grupos. El *primer grupo* sería el grupo de datos de entrenamiento, los cuales representan el *80%* de los datos de la base de datos original, el *segundo grupo* representa la base de datos de testeo los cuales representan el *20%* restante de los datos. Estos grupos de datos se dividen en 2 tipos, en variables *dependientes e independientes*, esto con el propósito de hacer las predicciones con los modelos y comprobar la eficiencia del modelo. Esta separación por grupos se hace por medio del siguiente comando:

```
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

4.4 y 4.5 Generación de los modelos y ajuste de hiperparámetros

I. Modelos Creados por Adalía Aneiros

A. K-Means

1. Mediante ‘param_grid’ se definen los hiperparámetros que se podrían utilizar más adelante en la búsqueda. Se incluyen valores diferentes para número de clusters, número de iteraciones y métodos de inicialización de centroides.
2. Se crea ‘grid_search’, que utiliza los hiperparámetros antes definidos, para, por medio de validación cruzada, determinar las estimaciones más precisas, que en este caso sin ‘init’: ‘random’, ‘max_iter’: 100, y ‘n_clusters’: 5.
3. Tomando estos valores en cuenta, se crea la instancia de KMeans, se obtienen las etiquetas y los centroides.
4. Para medir su calidad, calcula el coeficiente de silueta, que es 0.45.

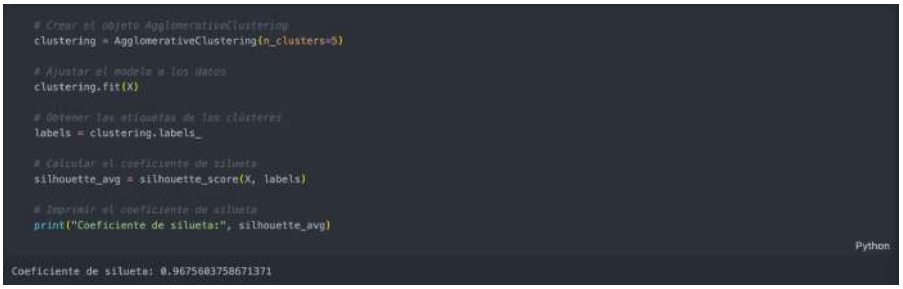
```python
param_grid = {
    'n_clusters': [2, 3, 4, 5],
    'init': ['k-means++', 'random'],
    'max_iter': [100, 300, 500]
}

# Perform grid search with cross-validation
grid_search = GridSearchCV(estimator=kmeans, param_grid=param_grid, cv=3)
grid_search.fit(scaled_data)

# Get the best parameters and best score
best_params = grid_search.best_params_


# Print the best parameters and best score
print("Best Parameters:", best_params)

kmeans = KMeans(n_clusters=5, max_iter=100)

# Fit the K-means model to the data
kmeans.fit(scaled_data)

# Get the cluster labels and centroids
labels = kmeans.labels_
centroids = kmeans.cluster_centers_

silhouette_avg = silhouette_score(scaled_data, labels)

print("Coeficiente de Silueta:", silhouette_avg)
```

Python

```
Best Parameters: {'init': 'random', 'max_iter': 100, 'n_clusters': 5}

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(

Coeficiente de Silueta: 0.455120772019799 8
```

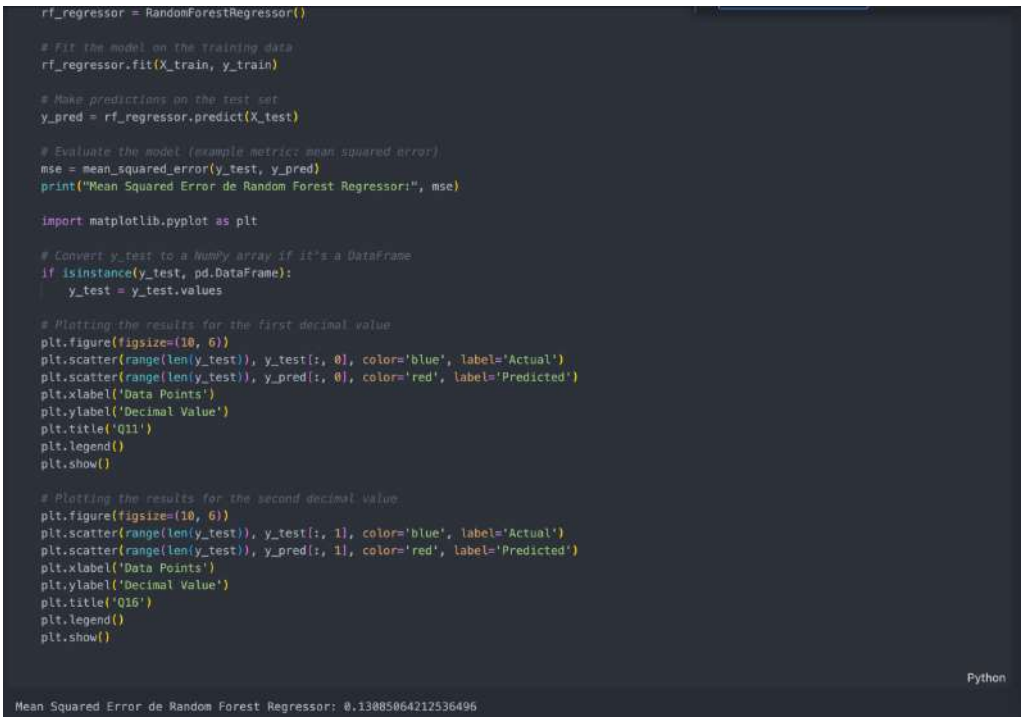
B. Agrupamiento Aglomerado

1. Se crea un objeto AgglomerativeClustering() con dos parámetros. n_clusters indica el número de clústeres deseados, en este caso es 5. El valor por default de linkage=ward especifica que se minimiza la varianza dentro de cada grupo, es decir, se minimiza la suma de las diferencias al cuadrado dentro de ellos.
2. El modelo se ajusta a los datos de entrada.
3. Se obtienen las etiquetas de clúster asignadas a cada muestra en función del modelo ajustado. Cada muestra se etiqueta con el número del clúster al que pertenece.
4. Se calcula el coeficiente de silueta promedio utilizando la función silhouette_score. X representa el conjunto de datos y labels son las etiquetas de los clústeres asignadas por el modelo.
5. Se muestra en pantalla el valor del coeficiente de silueta promedio obtenido anteriormente, proporcionando una medida de la calidad de la agrupación realizada por el modelo de agrupamiento aglomerativo.
6. Para este caso, el coeficiente de silueta es 0.96, un valor considerablemente alto, señal de que es un buen modelo.



C. Random Forest Regression

1. Se dividen los datos en conjuntos de entrenamiento y prueba. En este caso, el conjunto de prueba corresponde al 20% y el de entrenamiento al 80%. Se define una semilla para la generación de valores aleatorios (random_state) de 42, que es el valor normalmente usado.
2. Se utiliza la clase RandomForestRegressor() para entrenar el modelo ajustándolo al conjunto de datos antes definidos.
3. Se realizan las predicciones usando el set definido para prueba.
4. Para evaluar el modelo y su rendimiento, se calcula el Error Cuadrático Medio, que calcula la discrepancia entre los valores reales y los predichos. Resultó de 0.13, lo que indica que es un buen modelo, pues los datos difieren muy poco.
5. Finalmente, se grafican los resultados obtenidos, permitiendo comparar los datos reales con los predichos.



D. Support Machine Regressor

1. Se combinan las dos columnas de la variable objetivo y a solo una.
2. Se crea un objeto con el comando SVR() que representa el modelo de Support Vector Regression. Los parámetros de kernel, en este caso 'sigmoid', el parámetro C, que es 1, y el parámetro epsilon, en este caso 0.05.
3. Se ajusta el modelo de SVM a los datos X_train y y_train de entrenamiento.
4. Se hace la predicción de la variable objetivo Y según el dataset X de testeo.
5. Se separa la columna de la variable objetivo de prediccion a dos columnas.
6. Se calcula el error cuadrático medio (MSE) comparando las etiquetas reales del conjunto de prueba y_test con las predicciones del modelo y_pred.
7. Se imprime el resultado del MSE.

```python
# Concatenate the decimal values into a single target variable
y_train_combined = y_train.iloc[:, 0] + y_train.iloc[:, 1]

# Create an SVR model with desired parameters
svr = SVR(kernel='sigmoid', C=1.0, epsilon=0.05)

# Fit the SVR model on the training data
svr.fit(X_train, y_train_combined)

# Make predictions on the test set
y_pred_combined = svr.predict(X_test)

# Separate the predicted combined values into individual decimal values
y_pred = np.column_stack((y_pred_combined, y_pred_combined))

# Evaluate the model (example metric: mean squared error)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error de Support Vector Regressor:", mse)
```

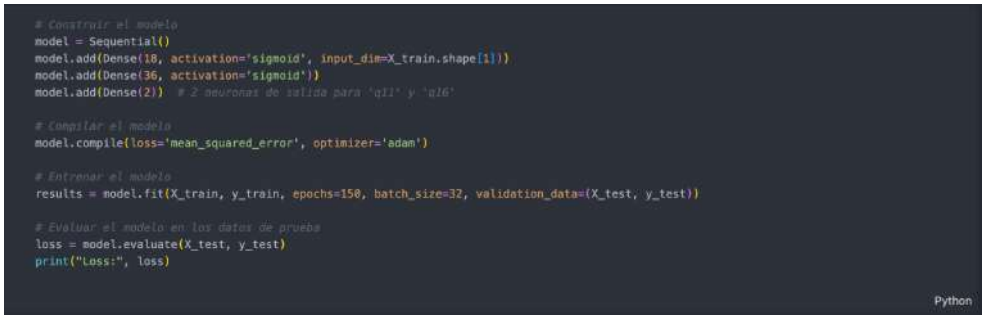
Python

```
Mean Squared Error de Support Vector Regressor: 2474.4665278546954
```

E. Red Neuronal

1. Se construye el modelo, definiendo tres capas, de entrada, de salida, y una oculta entre ellas. Se utiliza una función de activación *sigmoide,* característica por ser una curva en forma de "S", con valores de salida de 0 a 1.
2. Se compila el modelo con los parámetros definidos anteriormente, indicando que la función que se utilizará para obtener las variaciones entre datos es la del error cuadrático medio.
3. Se entrena el modelo con los conjuntos de datos predefinidos anteriormente y se evalúa mediante la función de pérdida. Para este

caso, es de 0.11, que, al ser un valor cercano a 0, nos indica que es un buen modelo.



```python
# Construir el modelo
model = Sequential()
model.add(Dense(18, activation='sigmoid', input_dim=X_train.shape[1]))
model.add(Dense(36, activation='sigmoid'))
model.add(Dense(2))  # 2 neuronas de salida para 'q11' y 'q16'

# Compilar el modelo
model.compile(loss='mean_squared_error', optimizer='adam')

# Entrenar el modelo
results = model.fit(X_train, y_train, epochs=150, batch_size=32, validation_data=(X_test, y_test))

# Evaluar el modelo en los datos de prueba
loss = model.evaluate(X_test, y_test)
print("Loss:", loss)
```

Python

```
Loss: 0.11173147708177567
```

Conclusión individual (Adalía): El haber podido probar diversos métodos de aprendizaje aplicados a este set de datos me hizo comprender que todos tienen objetivos diferentes y no son aplicables para todos los casos. Aunque en este reporte sólo presentamos cinco, estuvimos probando con más modelos, para finalmente optar por dos como los más certeros: Red Neuronal y Agrupamiento Aglomerativo, por la calidad de sus resultados. Adicionalmente, el modificar ciertos parámetros de cada uno de los modelos puede arrojar resultados diferentes, sin embargo, es imprescindible entender para qué sirve cada uno, para así no cometer errores y realizar los cambios que sean más eficientes en base a los objetivos que se busquen lograr.

II. Modelos Creados por Valeria Serna

A. K-Means

1. Primero, se crea un objeto KMeans con varios parámetros. N_Clusters indica el número de clusters deseados, en este caso son 4. Después, init='k-means++' se refiere a la estrategia de inicialización de los centroides. N_init es el número de veces que el algoritmo se ejecutará con diferentes semillas iniciales, max_iter es el número máximo de iteraciones permitidas en cada ejecución, tol es un umbral para la convergencia del algoritmo y random_state se utiliza para la reproducibilidad.
2. Se ajusta el modelo K-means a los datos escalados (scaled_data). El algoritmo se ejecuta para encontrar los centroides óptimos que definen los clusters.

3. Se obtienen las etiquetas de cluster asignadas a cada muestra en función del modelo ajustado. Cada muestra se etiqueta con el número del clúster al que pertenece.
4. Se obtienen los centroides finales de cada clúster, estos representan los puntos centrales de cada agrupación.
5. Se calcula el coeficiente de silueta promedio utilizando la función silhouette_score. Esta métrica evalúa la calidad de la agrupación asignando un valor entre -1 y 1, donde los valores más altos indican una mejor separación entre los clústeres.
6. Se muestra en pantalla el valor del coeficiente de silueta promedio obtenido anteriormente, proporcionando una medida de la calidad de la agrupación realizada por el modelo K-means.

7. El coeficiente de Silueta es de 0.43, lo que significa que el modelo no es el mejor.

B. Agrupamiento Aglomerado

1. Se crea un objeto AgglomerativeClustering() con dos parámetros. n_clusters indica el número de clústeres deseados, en este caso es 3. linkage='complete' especifica el método de enlace utilizado para calcular la distancia entre los clústeres durante el proceso de agrupamiento aglomerativo. En este caso, se utiliza el enlace completo.
2. El modelo se ajusta a los datos de entrada.
3. Se obtienen las etiquetas de clúster asignadas a cada muestra en función del modelo ajustado. Cada muestra se etiqueta con el número del clúster al que pertenece.
4. Se calcula el coeficiente de silueta promedio utilizando la función silhouette_score. X representa el conjunto de datos y labels son las etiquetas de los clústeres asignadas por el modelo.

5. Se muestra en pantalla el valor del coeficiente de silueta promedio obtenido anteriormente, proporcionando una medida de la calidad de la agrupación realizada por el modelo de agrupamiento aglomerativo.

```python
# Crear el objeto AgglomerativeClustering
clustering = AgglomerativeClustering(n_clusters=3,linkage='complete')

# Ajustar el modelo a los datos
clustering.fit(X)

# Obtener las etiquetas de los clústeres
labels = clustering.labels_

# Calcular el coeficiente de silueta
silhouette_avg = silhouette_score(X, labels)

# Imprimir el coeficiente de silueta
print("Coeficiente de silueta:", silhouette_avg)
```

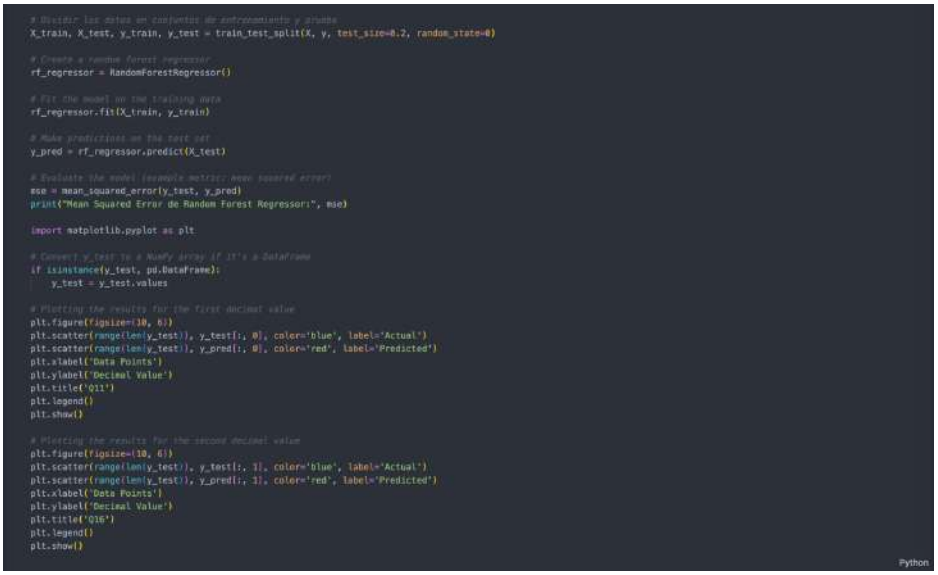
Python

```
Coeficiente de silueta: 0.9786570490712245
```

6. El resultado es 0.98, lo que significa un gran resultado.

C. Random Forest Regression

1. Los datos X y y se dividen en conjuntos de entrenamiento y prueba utilizando la función train_test_split de scikit-learn. El parámetro test_size=0.2 indica que el 20% de los datos se utilizará como conjunto de prueba, mientras que el 80% se utilizará como conjunto de entrenamiento.
2. Se crea un objeto RandomForestRegressor() que representa el modelo de regresión de bosque aleatorio.
3. El modelo de regresión de bosque aleatorio se ajusta a los datos de entrenamiento X_train y las etiquetas y_train.
4. Se realizan predicciones en el conjunto de prueba X_test utilizando el modelo entrenado.
5. Se calcula el error cuadrático medio (MSE) comparando las etiquetas reales del conjunto de prueba y_test con las predicciones del modelo y_pred.
6. Se imprime el resultado del MSE y se grafica el modelo.



```
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Create a random forest regressor
rf_regressor = RandomForestRegressor()

# Fit the model on the training data
rf_regressor.fit(X_train, y_train)

# Make predictions on the test set
y_pred = rf_regressor.predict(X_test)

# Evaluate the model (example metric: mean squared error)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error de Random Forest Regressor:", mse)

import matplotlib.pyplot as plt

# Convert y_test to a NumPy array if it's a DataFrame
if isinstance(y_test, pd.DataFrame):
    y_test = y_test.values

# Plotting the results for the first decimal value
plt.figure(figsize=(10, 6))
plt.scatter(range(len(y_test)), y_test[:, 0], color='blue', label='Actual')
plt.scatter(range(len(y_test)), y_pred[:, 0], color='red', label='Predicted')
plt.xlabel('Data Points')
plt.ylabel('Decimal Value')
plt.title('Q11')
plt.legend()
plt.show()

# Plotting the results for the second decimal value
plt.figure(figsize=(10, 6))
plt.scatter(range(len(y_test)), y_test[:, 1], color='blue', label='Actual')
plt.scatter(range(len(y_test)), y_pred[:, 1], color='red', label='Predicted')
plt.xlabel('Data Points')
plt.ylabel('Decimal Value')
plt.title('Q16')
plt.legend()
plt.show()
```

Python

```
Mean Squared Error de Random Forest Regressor: 0.1295060087960176
```

7. El resultado del MSE es 0.13 y, considerando que entre menor la cantidad mejor el modelo, significa que es un bueno y sus predicciones serán mayormente correctas.
8. Se imprime el resultado del MSE .

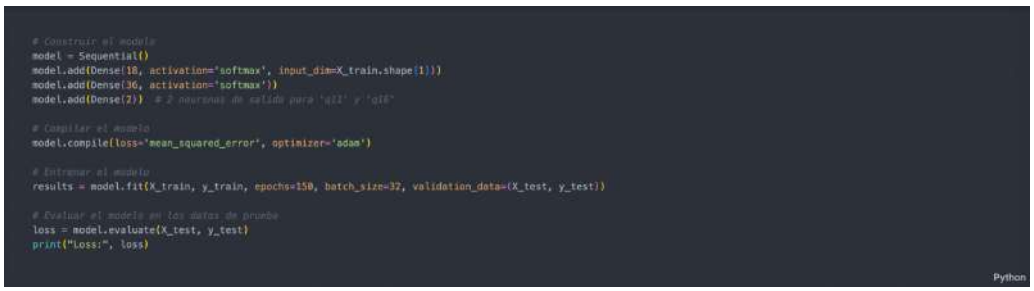
D. Support Machine Regressor

1. Se combinan las dos columnas de la variable objetivo y a solo una.
2. Se crea un objeto con el comando SVR() que representa el modelo de Support Vector Regression. con los parámetros de kernel en este caso ‘rbf’,el parámetro C entel caso 1 y el parámetro epsilon en este caso 0.1
3. Se ajusta el modelo de SVM a los datos X_train y y_train de entrenamiento.
4. Se hace la predicción de la variable objetivo Y según el dataset X de testeo.
5. Se separa la columna de la variable objetivo de prediccion a dos columnas.
6. Se calcula el error cuadrático medio (MSE) comparando las etiquetas reales del conjunto de prueba y_test con las predicciones del modelo y_pred.
7. Se imprime el resultado del MSE.

8. El resultado del MSE es de 0.18, y, aunque podría ser menor, es bueno.

E. Red Neuronal

1. Se crea un objeto Sequential() que representa un modelo de red neuronal secuencial. Este es adecuado para una arquitectura de red en la que las capas se apilan una encima de la otra.
2. Se agrega una capa densamente conectada (Dense) al modelo. La capa tiene 18 neuronas y utiliza la función de activación softmax. El parámetro input_dim se establece en el número de características en los datos de entrada X_train.
3. Se agrega otra capa densamente conectada al modelo, con 36 neuronas y la función de activación softmax.
4. Se agrega una capa densamente conectada final al modelo con 2 neuronas de salida. Esta capa se utiliza para predecir las variables 'Q11' y 'Q16', las cuales tienen valores continuos.
5. Se compila el modelo usando al error cuadrático medio (mean_squared_error) como función de pérdida. El optimizador "adam" se utiliza para ajustar los pesos y minimizar la pérdida durante el entrenamiento.
6. Se entrena el modelo utilizando los datos de entrenamiento X_train y las etiquetas y_train. El modelo se entrena durante 150 épocas y se utiliza un tamaño de lote de 32. Además, se proporcionan los datos de prueba X_test y las etiquetas y_test como datos de validación para evaluar el rendimiento del modelo durante el entrenamiento.
7. Se evalúa el modelo utilizando los datos de prueba y las etiquetas. Esto calcula la pérdida del modelo en los datos de prueba.



```python
# Construir el modelo
model = Sequential()
model.add(Dense(18, activation='softmax', input_dim=X_train.shape[1]))
model.add(Dense(36, activation='softmax'))
model.add(Dense(2))  # 2 neuronas de salida para 'q11' y 'q18'

# Compilar el modelo
model.compile(loss='mean_squared_error', optimizer='adam')

# Entrenar el modelo
results = model.fit(X_train, y_train, epochs=150, batch_size=32, validation_data=(X_test, y_test))

# Evaluar el modelo en los datos de prueba
loss = model.evaluate(X_test, y_test)
print("Loss:", loss)
```

Python

```
Loss: 0.11134801059961319
```

8. El resultado de pérdida es 0.11, lo que demuestra la precisión del modelo.

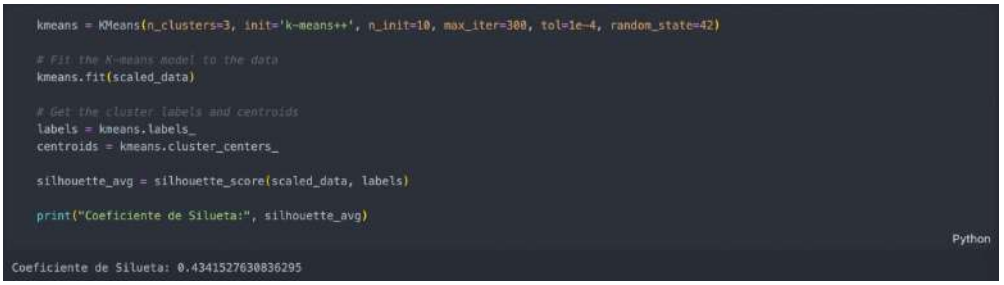
Conclusión Individual (Valeria): Al hacer los diferentes modelos y ver las métricas que salen, es posible observar la manera en que cada parámetro tiene un peso diferente en el resultado. Es necesario recalcar que también se mostró claro el objetivo y el funcionamiento que tiene cada uno, así como el aprendizaje de poder aplicar, de cero, estos modelos.

III. Modelos creados por Diego Garza González

A. K-Means

1. Se importan las bibliotecas necesarias: KMeans, StandardScaler, PCA y silhouette_score de scikit-learn, matplotlib.pyplot.
2. Se normalizan los datos utilizando StandardScaler. Asegura que todas las variables tengan la misma escala y evitar que una variable domine sobre las demás en el proceso.
3. Se crea un objeto de KMeans con parámetros específicos: 3 clusters, inicialización k-means++, un máximo de 300 iteraciones, criterio de tolerancia de 1e-4, 10 repeticiones del algoritmo con diferentes centroides iniciales, y se utiliza una semilla aleatoria para asegurar la reproducibilidad de los resultados.
4. Se entrena el modelo utilizando el método fit(). Encuentra los centroides óptimos y asigna etiquetas de cluster a los datos.
5. Se muestra el coeficiente de silhouette_score(). Mide la unión y la separación de los clusters, dando una medida de calidad de la

agrupación (Un valor de 1 indica una buena separación y unión de clusters).



```python
kmeans = KMeans(n_clusters=3, init='k-means++', n_init=10, max_iter=300, tol=1e-4, random_state=42)

# Fit the K-means model to the data
kmeans.fit(scaled_data)

# Get the cluster labels and centroids
labels = kmeans.labels_
centroids = kmeans.cluster_centers_

silhouette_avg = silhouette_score(scaled_data, labels)

print("Coeficiente de Silueta:", silhouette_avg)
```

Python

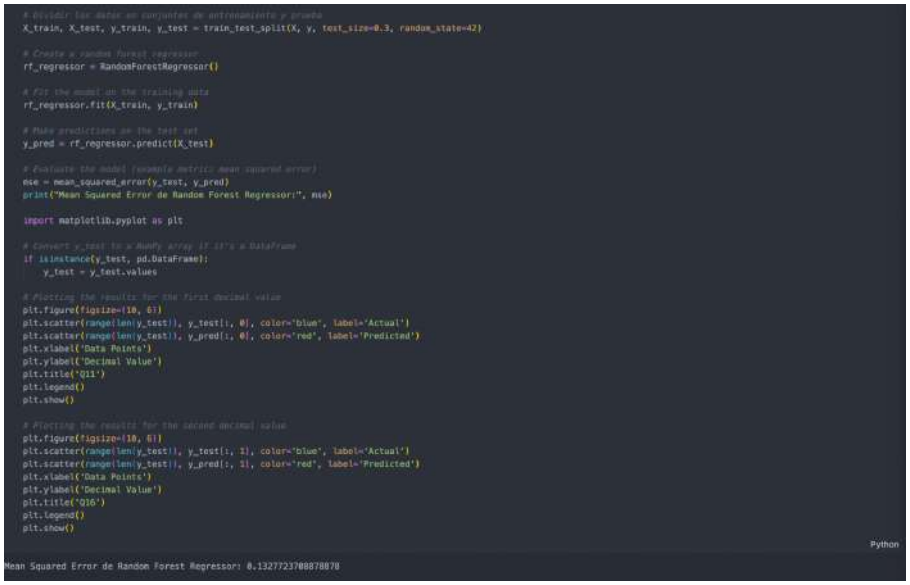
```
Coeficiente de Silueta: 0.4341527630836295
```

B. **Agrupamiento Aglomerado**

1. Se importan las librerías necesarias: AgglomerativeClustering y silhouette_score de scikit-learn, matplotlib.pyplot, numpy.
2. Se crea un objeto de AgglomerativeClustering con parámetros específicos: 5 clusters.
3. Se entrena el modelo utilizando el método fit(). Esto realiza el proceso de agrupamiento jerárquico aglomerativo y asigna etiquetas de cluster a los datos.
4. Se obtienen las etiquetas de cluster generadas por el modelo utilizando el atributo labels_.
5. Se muestra el coeficiente de silhouette_score(). Mide la unión y la separación de los clusters, dando una medida de calidad de la agrupación (Un valor de 1 indica una buena separación y unión de clusters).
6. Se convierten las etiquetas de prueba en un arreglo y se muestran los resultados de ambas salidas con gráficas de dispersión.

C. Random Forest Regression

1. Se importan las librerías necesarias: RandomForestRegressor y mean_squared_error de scikit-learn, matplotlib.pyplot, numpy.
2. Los datos se dividen en conjuntos de entrenamiento y prueba. Se reserva el 30% de los datos para las pruebas y se utiliza una semilla aleatoria para asegurar la reproducibilidad de los resultados.
3. Se crea un objeto de RandomForestRegressor sin especificar ningún parámetro (usará valores predeterminados de la librería).
4. Se entrena el modelo utilizando el método fit().
5. Se realizan predicciones en los datos de prueba utilizando el método predict() y se almacenan estas predicciones.
6. Se calcula el error cuadrático medio para las variables de salida, calculando la diferencia entre las etiquetas de p los valores reales con las predicciones y se calcula el valor de pérdida del modelo en los datos de prueba.
7. Se convierten las etiquetas de prueba en un arreglo y se muestran los resultados de ambas salidas con gráficas de dispersión.



```
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Create a random forest regressor
rf_regressor = RandomForestRegressor()

# Fit the model on the training data
rf_regressor.fit(X_train, y_train)

# Make predictions on the test set
y_pred = rf_regressor.predict(X_test)

# Evaluate the model (example metric: mean squared error)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error de Random Forest Regressor:", mse)

import matplotlib.pyplot as plt

# Convert y_test to a NumPy array if it's a DataFrame
if isinstance(y_test, pd.DataFrame):
    y_test = y_test.values

# Plotting the results for the first decimal value
plt.figure(figsize=(10, 6))
plt.scatter(range(len(y_test)), y_test[:, 0], color='blue', label='Actual')
plt.scatter(range(len(y_test)), y_pred[:, 0], color='red', label='Predicted')
plt.xlabel('Data Points')
plt.ylabel('Decimal Value')
plt.title('Q11')
plt.legend()
plt.show()

# Plotting the results for the second decimal value
plt.figure(figsize=(10, 6))
plt.scatter(range(len(y_test)), y_test[:, 1], color='blue', label='Actual')
plt.scatter(range(len(y_test)), y_pred[:, 1], color='red', label='Predicted')
plt.xlabel('Data Points')
plt.ylabel('Decimal Value')
plt.title('Q16')
plt.legend()
plt.show()
```

Python

```
Mean Squared Error de Random Forest Regressor: 0.1327723708878878
```

D. Support Vector Regressor

1. Se importan las librerías necesarias: SVR y mean_squared_error de scikit-learn, numpy.
2. Se concatenan los valores decimales de las variables de salida “Q11” y “Q16” en una única variable de destino. Esto se hace utilizando la función iloc[] para seleccionar las columnas correspondientes y luego sumándolas. Esto es necesario para ajustar el modelo de SVR que requiere una única variable de destino.
3. Se crea un objeto de SVR con parámetros específicos: kernel “rbf”, epsilon de 0.15, y un parámetro de regularización 1.
4. Se entrena el modelo utilizando el método fit().
5. Se realizan predicciones en los datos de prueba utilizando el método predict() y se almacenan estas predicciones. Las predicciones combinadas se separan en valores individuales de las variables decimales utilizando la función columna_stack().
6. Se evalúa el modelo utilizando la métrica MSE, calculando la diferencia entre las etiquetas de prueba y las predicciones.

E. Red Neuronal

1. Se importan las librerías necesarias: numpy, train_test_split y mean_squared_error de scikit-learn, y las clases Sequential y Dense de Keras.
2. Se preparan los datos. El conjunto de datos original se divide en dos partes: “X”, que contiene las variables de entrada (todas las preguntas excepto “Q11” y “Q16”) y “Y”, que contiene las variables de salida (“Q11” y “Q16”).
3. Los datos se dividen en conjuntos de entrenamiento y prueba. Se reserva el 20% de los datos para las pruebas y se utiliza una semilla aleatoria para asegurar la reproducibilidad de los resultados.
4. Se construye el modelo de red neuronal utilizando la clase Sequential. Se agregan capas Dense, que son capas completamente conectadas. El modelo consiste en una capa inicial de 18 neuronas y una capa oculta de 36 neuronas, las cuales son activadas por la función de activación tangente “tanh”. Finalmente hay una capa de salida con 2 neuronas para predecir los valores de “q11” y “q16”.
5. Se compila el modelo especificando la función de pérdida “mean_squared_error” y el optimizador “adam” a utilizar durante el entrenamiento.
6. Se entrena el modelo utilizando el método fit(). Se especificó que el modelo se entrena con 150 épocas y un tamaño de lote de 32.
7. Se evalúa el modelo de los datos utilizando el método evaluate() y se calcula la pérdida del modelo en los datos de prueba.
8. Se realizan predicciones en los datos de prueba utilizando el método predict() y se almacenan estas predicciones.
9. Se calcula el error cuadrático medio para las variables de salida, comparando los valores reales con las predicciones y se calcula el valor de pérdida del modelo en los datos de prueba.

```python
# Construir el modelo
model = Sequential()
model.add(Dense(18, activation='tanh', input_dim=X_train.shape[1]))
model.add(Dense(36, activation='tanh'))
model.add(Dense(2))  # 2 neuronas de salida para 'q11' y 'q16'

# Compilar el modelo
model.compile(loss='mean_squared_error', optimizer='adam')

# Entrenar el modelo
results = model.fit(X_train, y_train, epochs=150, batch_size=32, validation_data=(X_test, y_test))

# Evaluar el modelo en los datos de prueba
loss = model.evaluate(X_test, y_test)
print("Loss:", loss)
```

Python

```
Loss: 0.11178908497095108
```

Conclusión individual (Diego): Después de implementar y evaluar 5 modelos de aprendizaje automático diferentes para el análisis de la base de datos de comentarios, se entiende de los resultados que los modelos de agrupación y de redes neuronales tuvieron mejor precisión a la hora de realizar una predicción del sentimiento de los comentarios. Por otro lado, los otros modelos obtuvieron buenos resultados en la predicción de las variables que almacenaban los comentarios, sin embargo no fueron tan precisas como los 2 modelos mencionados anteriormente. A lo largo del proyecto, aprendí y termine de comprender la implementación de estos procesos de ML, con los cuales puedo presentar resultados útiles y fructíferos hacia el departamento de Servicio Social y a las mismas OSFs. El aprendizaje de mayor impacto que aprendí en este reto, es el hecho de que hay muchas maneras y técnicas para analizar información, y no siempre se podrá aplicar todo a un método. De la misma manera, desarrolle mis habilidades autodidactas para la realización de estos entregables.

IV. Modelos Creados por Gian Marco Innocenti

A. K-Means

1. Primero, se crea un objeto KMeans. Se ingresa el parametro N_Clustersen este caso son 2.
2. Se ajusta el modelo K-means a los datos escalados (scaled_data). El.
3. Se obtienen las etiquetas mediante el comando kmeans.labels_
4. Se calcula el coeficiente de silueta promedio utilizando la función silhouette_score. En este caso se obtuvo un score de 0.46

```python
# Normalizar datos
scaler = StandardScaler()
scaled_data = scaler.fit_transform(X)

kmeans = KMeans(n_clusters=2)

kmeans.fit(scaled_data)

cluster_labels = kmeans.labels_

silhouette_avg = silhouette_score(scaled_data, cluster_labels)

print("Coeficiente de Silueta:", silhouette_avg)
```

Python

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(

Coeficiente de Silueta: 0.4610549799477016
```

B. Agrupamiento Aglomerado

1. Se crea un objeto AgglomerativeClustering() con dos parámetros. n_clusters indica el número de clústeres deseados, en este caso es 2.
2. Se ajusta el modelo al dataframe X.
3. Se obtienen las etiquetas de clúster con el comando clustering.labels_
4. Se calcula el coeficiente de silueta promedio utilizando la función silhouette_score. En este caso el score fue de 0.98

```python
# Crear el objeto AgglomerativeClustering
clustering = AgglomerativeClustering(n_clusters=2)

# Ajustar el modelo a los datos
clustering.fit(X)

# Obtener las etiquetas de los clústeres
labels = clustering.labels_

# Calcular el coeficiente de silueta
silhouette_avg = silhouette_score(X, labels)

# Imprimir el coeficiente de silueta
print("Coeficiente de silueta:", silhouette_avg)
```

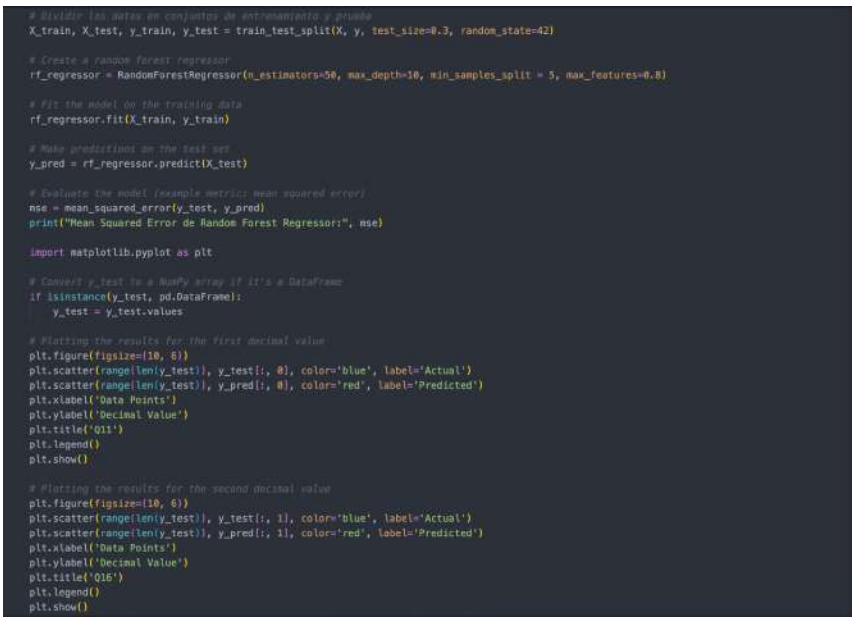
Python

```
Coeficiente de silueta: 0.9804905051036832
```

C. Random Forest Regression

1. Se hace un train test split entre el dataframe X y el dataframe y con un tamaño de testeo de el 30%
2. Se crea un objeto RandomForestRegressor() que representa el modelo de regresión de bosque aleatorio. con los parametros n_estimators:50,max_depth:10,min_samples_split:5, y max_features:0.8
3. Se ajusta el modelo a los datos de entrenamiento X_train y las etiquetas y_train.
4. Se realizan predicciones en el conjunto de prueba X_test utilizando el modelo entrenado.
5. Se calcula el error cuadrático medio (MSE) comparando y_test y y_pred. En este caso el score fue de 0.11

6. Se imprime el resultado del MSE y se grafica el modelo.



```
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Create a random forest regressor
rf_regressor = RandomForestRegressor(n_estimators=50, max_depth=10, min_samples_split = 5, max_features=0.8)

# Fit the model on the training data
rf_regressor.fit(X_train, y_train)

# Make predictions on the test set
y_pred = rf_regressor.predict(X_test)

# Evaluate the model (example metric: mean squared error)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error de Random Forest Regressor:", mse)

import matplotlib.pyplot as plt

# Convert y_test to a NumPy array if it's a DataFrame
if isinstance(y_test, pd.DataFrame):
    y_test = y_test.values

# Plotting the results for the first decimal value
plt.figure(figsize=(10, 6))
plt.scatter(range(len(y_test)), y_test[:, 0], color='blue', label='Actual')
plt.scatter(range(len(y_test)), y_pred[:, 0], color='red', label='Predicted')
plt.xlabel('Data Points')
plt.ylabel('Decimal Value')
plt.title('Q11')
plt.legend()
plt.show()

# Plotting the results for the second decimal value
plt.figure(figsize=(10, 6))
plt.scatter(range(len(y_test)), y_test[:, 1], color='blue', label='Actual')
plt.scatter(range(len(y_test)), y_pred[:, 1], color='red', label='Predicted')
plt.xlabel('Data Points')
plt.ylabel('Decimal Value')
plt.title('Q16')
plt.legend()
plt.show()
```

D. Support Machine Regressor

1. Se combinan las dos columnas de la variable objetivo a una sola.
2. Se crea un objeto con el comando SVR() que representa el modelo de Support Vector Regression. con los parámetros de kernel en este caso ‘rbf’,el parámetro C entel caso 1 y el parámetro epsilon en este caso 0.3.
3. Se ajusta el modelo de SVM a los datos X_train y y_train de entrenamiento.
4. Se hace la predicción de la variable objetivo Y según el dataset X de testeo.
5. Se separa la columna de la variable objetivo de predicción a dos columnas.
6. Se calcula el error cuadrático medio (MSE) comparando las etiquetas reales del conjunto de prueba y_test con las predicciones del modelo y_pred. En este casoa el score fue de 0.17.

E. Red Neuronal

Se importan las librerias necesarias: numpy, train_test_split y mean_sde scikit-learn, y las clases Sequential y Dense de Keras.

Se preparan los datos. El conjunto de datos original se divide en dos partes: “X”, que contiene las variables de entrada (todas las preguntas excepto “Q11” y “Q16”) y “Y”, que contiene las variables de salida (“Q11” y “Q16”).

Los datos se dividen en conjuntos de entrenamiento y prueba. Se reserva el 20% de los datos para las pruebas y se utiliza una semilla aleatoria para asegurar la reproducibilidad de los resultados.

Se construye el modelo de red neuronal utilizando la clase Sequential. Se agregan capas Dense, que son capas completamente conectadas. El modelo consiste en una capa inicial de 18 neuronas y una capa oculta de 36 neuronas, las cuales son activadas por la función de activación Exponential Linear Unit “Elu”. Finalmente hay una capa de salida con 2 neuronas para predecir los valores de “q11” y “q16”.

Se compila el modelo especificando la función de pérdida “mean_squared_error” y el optimizador “adam” a utilizar durante el entrenamiento.

Se entrena el modelo utilizando el método fit(). Se especificó que el modelo se entrena con 150 épocas y un tamaño de lote de 32.

Se evalúa el modelo de los datos utilizando el método evaluate() y se calcula la pérdida del modelo en los datos de prueba.

Se realizan predicciones en los datos de prueba utilizando el método predict() con las predicciones almacenadas.

Se calcula el error cuadrático medio para las variables de salida, comparando los valores reales con las predicciones y se calcula el valor de pérdida del modelo en los datos de prueba. Para este caso el error cuadittico medio para la pregunta Q11 fue de 0.12 y para la pregunta Q16 fue de 0.09.

```python
# Construir el modelo
model = Sequential()
model.add(Dense(18, activation='elu', input_dim=X_train.shape[1]))
model.add(Dense(36, activation='elu'))
model.add(Dense(2))  # 2 neuronas de salida para 'q11' y 'q16'

# Compilar el modelo
model.compile(loss='mean_squared_error', optimizer='adam')

# Entrenar el modelo
results = model.fit(X_train, y_train, epochs=150, batch_size=32, validation_data=(X_test, y_test))

# Evaluar el modelo en los datos de prueba
loss = model.evaluate(X_test, y_test)
print("Loss:", loss)
```

Python

```
Loss: 0.11209449172019958
```

Conclusión Individual (Gian): La creación de estos modelos fue un proceso arduo pero que me enriqueció mucho. Describiría esta etapa como una de exploración, ya que tuve que estar experimentando con los parámetros de cada modelo para tratar de conseguir mejores resultados. No solo tuve que experimentar con los parámetros de los modelos sino que también tuve que experimentar con los datasets que entraban a los modelos. El mejor ejemplo de esto es que probaba el mismo modelo con datasets escalados utilizando el método scaler() y el dataset normal, no hubo una opción definitiva tendía a cambiar según el modelo. En conclusión esta ha sido de las etapas en las que siento que adquirir más conocimiento ya que batallamos mucho como grupo y al final con la colaboración entre mis compañeros logramos obtener muy buenos resultados.

V. Modelos creados por Victor Hugo Garza

A. K-Means

1. Primero, se crea un objeto KMeans con varios parámetros. Se establece el número deseado de clusters mediante el parámetro N_Clusters, que en este caso es 4. El parámetro init=’k-means++’ se refiere a la estrategia de inicialización de los centroides. N_init determina cuántas veces se ejecutará el algoritmo con diferentes

semillas iniciales. Max_iter establece el número máximo de iteraciones permitidas en cada ejecución, tol es un umbral para la convergencia del algoritmo y random_state se utiliza para garantizar la reproducibilidad de los resultados.

2. Se ajusta el modelo K-means a los datos escalados (scaled_data). Esto implica ejecutar el algoritmo para encontrar los centroides óptimos que definen los clusters.
3. Se obtienen las etiquetas de cluster asignadas a cada muestra en función del modelo ajustado. Cada muestra se etiqueta con el número del cluster al que pertenece.
4. Se obtienen los centroides finales de cada cluster. Estos centroides representan los puntos centrales de cada agrupación.
5. Se calcula el coeficiente de silueta promedio utilizando la función silhouette_score. Esta métrica evalúa la calidad de la agrupación asignando un valor entre -1 y 1, donde valores más altos indican una mejor separación entre los clusters.
6. Se muestra en pantalla el valor del coeficiente de silueta promedio obtenido anteriormente, lo cual proporciona una medida de la calidad de la agrupación realizada por el modelo K-means.

```python
kmeans = KMeans(n_clusters=2, init='k-means++', n_init=8, max_iter=100, tol=1e-4, random_state=42)

# Fit the K-means model to the data
kmeans.fit(scaled_data)

# Get the cluster labels and centroids
labels = kmeans.labels_
centroids = kmeans.cluster_centers_

silhouette_avg = silhouette_score(scaled_data, labels)

print("Coeficiente de Silueta:", silhouette_avg)
```

Python

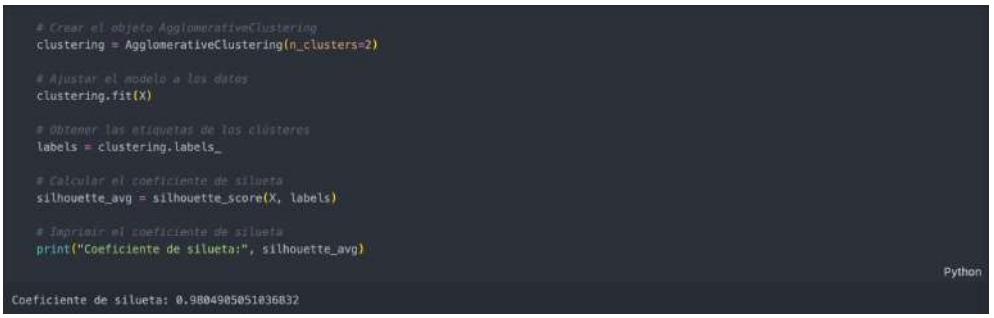
```
Coeficiente de Silueta: 0.46105497994770016
```

7. El coeficiente de Silueta es de 0.46, lo que significa que el modelo no es el mejor.

B. Agrupamiento Aglomerado

1. Se crea un objeto AgglomerativeClustering() con dos parámetros. El parámetro n_clusters indica el número deseado de clústeres, en este caso son 4.
2. El modelo se ajusta a los datos de entrada. Esto implica que el algoritmo de agrupamiento aglomerativo se ejecuta para encontrar los clústeres óptimos.

3. Se obtienen las etiquetas de clúster asignadas a cada muestra en función del modelo ajustado. Cada muestra se etiqueta con el número del clúster al que pertenece.
4. Se calcula el coeficiente de silueta promedio utilizando la función silhouette_score. Se utiliza el conjunto de datos X y las etiquetas de los clústeres (labels) asignadas por el modelo.
5. Se muestra en pantalla el valor del coeficiente de silueta promedio obtenido anteriormente, lo cual proporciona una medida de la calidad de la agrupación realizada por el modelo de agrupamiento aglomerativo.



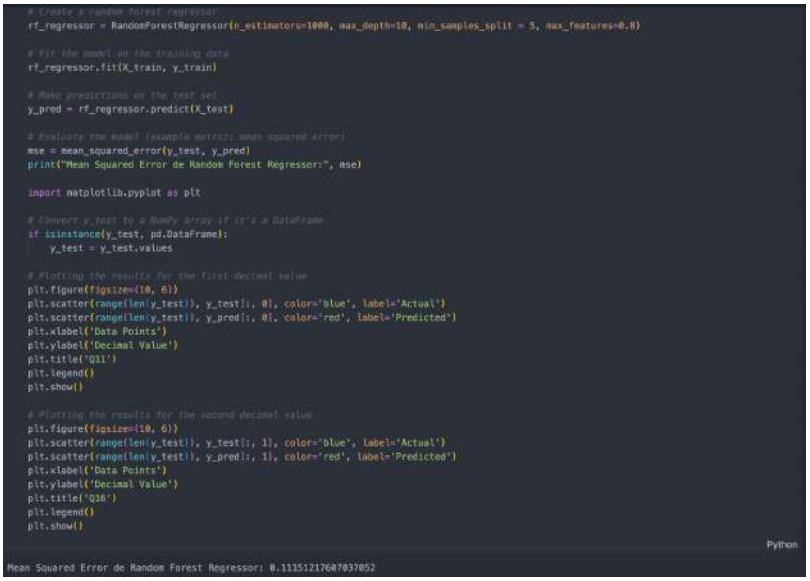
6. El resultado es 0.967, lo que significa un gran resultado pero no el mejor de todas las opciones.

C. Random Forest Regression

1. Los datos X y y se dividen en conjuntos de entrenamiento y prueba utilizando la función train_test_split de la biblioteca scikit-learn. El parámetro test_size=0.2 indica que el 20% de los datos se utilizará como conjunto de prueba, mientras que el 80% se utilizará como conjunto de entrenamiento.
2. Se crea un objeto RandomForestRegressor(), el cual representa el modelo de regresión de bosque aleatorio.
3. El modelo de regresión de bosque aleatorio se ajusta a los datos de entrenamiento X_train y las etiquetas y_train. Esto implica entrenar el modelo con los datos disponibles para que pueda aprender a hacer predicciones precisas.
4. Se realizan predicciones en el conjunto de prueba X_test utilizando el modelo entrenado. Esto implica usar el modelo para predecir los valores correspondientes a X_test.

5. Se calcula el error cuadrático medio (MSE) comparando las etiquetas reales del conjunto de prueba y_test con las predicciones del modelo y_pred. El MSE es una medida de qué tan cerca están las predicciones del modelo de los valores reales.
6. Se imprime el resultado del MSE y se grafica el modelo. Esto permite evaluar la precisión del modelo y visualizar cómo se ajusta a los datos de entrenamiento y prueba.

7. El resultado del MSE es 0.112 y, considerando que entre menor la cantidad mejor el modelo, significa que es un bueno y sus predicciones serán mayormente correctas.
8. Se imprime el resultado del MSE .



```
# Create a random forest regressor
rf_regressor = RandomForestRegressor(n_estimators=1000, max_depth=10, min_samples_split = 5, max_features=0.8)

# Fit the model on the training data
rf_regressor.fit(X_train, y_train)

# Make predictions on the test set
y_pred = rf_regressor.predict(X_test)

# Evaluate the model (example metric: mean squared error)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error de Random Forest Regressor:", mse)

import matplotlib.pyplot as plt

# Convert y_test to a NumPy array if it's a DataFrame
if isinstance(y_test, pd.DataFrame):
    y_test = y_test.values

# Plotting the results for the first decimal value
plt.figure(figsize=(10, 6))
plt.scatter(range(len(y_test)), y_test[:, 0], color='blue', label='Actual')
plt.scatter(range(len(y_test)), y_pred[:, 0], color='red', label='Predicted')
plt.xlabel('Data Points')
plt.ylabel('Decimal Value')
plt.title('Q11')
plt.legend()
plt.show()

# Plotting the results for the second decimal value
plt.figure(figsize=(10, 6))
plt.scatter(range(len(y_test)), y_test[:, 1], color='blue', label='Actual')
plt.scatter(range(len(y_test)), y_pred[:, 1], color='red', label='Predicted')
plt.xlabel('Data Points')
plt.ylabel('Decimal Value')
plt.title('Q16')
plt.legend()
plt.show()
```

Python

```
Mean Squared Error de Random Forest Regressor: 0.11151217607037052
```

D. Support Machine Regressor

1. Se combinan las dos columnas de la variable objetivo y en una sola columna.
2. Se crea un objeto utilizando el comando SVR(), el cual representa el modelo de Support Vector Regression. Se especifican los siguientes parámetros: el kernel utilizado es 'sigmoid', el valor de C es 1 y el valor de epsilon es 0.5.
3. El modelo de SVM se ajusta a los datos de entrenamiento X_train y y_train. Esto implica entrenar el modelo utilizando los datos disponibles para que pueda aprender a hacer predicciones precisas.

4. Se realiza la predicción de la variable objetivo Y utilizando el conjunto de datos X de prueba.
5. Se separa la columna de la variable objetivo de predicción en dos columnas.
6. Se calcula el error cuadrático medio (MSE) comparando las etiquetas reales del conjunto de prueba y_test con las predicciones del modelo y_pred. El MSE es una medida de qué tan cerca están las predicciones del modelo de los valores reales.
7. Se imprime el resultado del MSE, proporcionando una medida del error promedio entre las predicciones del modelo y los valores reales.

8. El resultado del MSE es de 2461.93, este valor es muy alto, denotando que este modelo no lo deberíamos utilizar ya que el error es gigantesco.

E. Red Neuronal

1. Se crea un objeto Sequential() que representa un modelo de red neuronal secuencial. Este tipo de modelo es adecuado cuando las capas de la red se apilan una encima de la otra en secuencia.
2. Se agrega una capa densamente conectada (Dense) al modelo. Esta capa tiene 18 neuronas y utiliza la función de activación relu. El parámetro input_dim se establece en el número de características en los datos de entrada X_train.
3. Se agrega otra capa densamente conectada al modelo, esta vez con 36 neuronas y la función de activación relu.
4. Se agrega una capa densamente conectada final al modelo con 2 neuronas de salida. Esta capa se utiliza para predecir las variables 'Q11' y 'Q16', las cuales tienen valores continuos.

5. Se compila el modelo utilizando el error cuadrático medio (mean_squared_error) como función de pérdida. Se utiliza el optimizador "adam" para ajustar los pesos del modelo y minimizar la pérdida durante el entrenamiento.
6. Se entrena el modelo utilizando los datos de entrenamiento X_train y las etiquetas y_train. El modelo se entrena durante 150 épocas y se utiliza un tamaño de lote de 32. Además, se proporcionan los datos de prueba X_test y las etiquetas y_test como datos de validación para evaluar el rendimiento del modelo durante el entrenamiento.
7. Se evalúa el modelo utilizando los datos de prueba y las etiquetas. Esto calcula la pérdida del modelo en los datos de prueba, lo cual permite evaluar qué tan bien está haciendo predicciones en nuevos datos.

```python
# Construir el modelo
model = Sequential()
model.add(Dense(18, activation='relu', input_dim=X_train.shape[1]))
model.add(Dense(36, activation='relu'))
model.add(Dense(2))  # 2 neuronas de salida para 'g11' y 'g16'

# Compilar el modelo
model.compile(loss='mean_squared_error', optimizer='adam')

# Entrenar el modelo
results = model.fit(X_train, y_train, epochs=150, batch_size=32, validation_data=(X_test, y_test))

# Evaluar el modelo en los datos de prueba
loss = model.evaluate(X_test, y_test)
print("Loss:", loss)
```

Python

```
Loss: 0.12410008907318115
```

8. El resultado de pérdida es 0.124, lo que demuestra la precisión del modelo. Cabe aclarar que es bastante buena ya que la métrica se aproxima mucho al 0

Conclusión Individual (Victor): Después de realizar los 5 modelos con sus respectivas características y calcular sus respectivas métricas de rendimiento es posible observar que hay modelos que se adaptan mejor a nuestros datos que otros. El modelo de clustering que mejor se adapta a nuestros datos es el *Agrupamiento algometativo* obteniendo un rendimiento excepcional y métricas de rendimiento que denotan una eficacia al agrupar los datos. Por otra parte, para el modelo de predicción se obtuvieron los mejores resultados con la *red neuronal* propuesta, con este modelo se obtuvieron métricas igualmente buenas denotando una eficacia de predicción de alrededor del 90%, siendo esto muy bueno. Adicionalmente, el objetivo y el funcionamiento que cada modelo propuesto por mi parte se mostraron claros, además de que el aprendizaje para poder aplicar estos modelos quedó

muy claro de igual forma. Finalmente, gracias a los modelos de ML pudimos llegar a resultados útiles para nuestra OSF con la cual podremos brindarles ayuda para que tomen decisiones informadas sobre las experiencias que ofrecen a sus alumnos.

4.6 Evaluación y selección de modelos

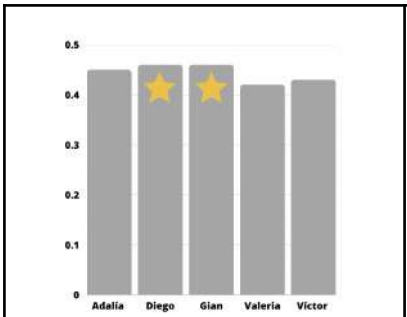
Comparación de resultados de los modelos por alumno

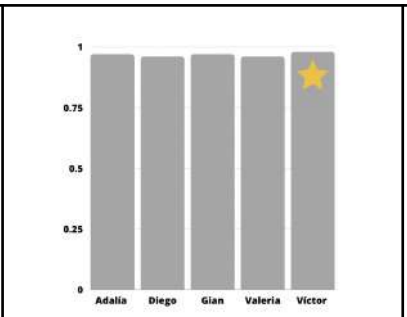
Modelo	Adalía	Diego	Gian	Valeria	Víctor
Kmeans	test_size=100% n_clusters=3 max_itter=100,	test_size=100% n_clusters=2 init='kmeans' n_init=8, max_itter=100, tol=1e_4, random_state=42	test_size=100% n_clusters=2	test_size=100% n_clusters=3 init='kmeans' n_init=9, max_itter=400, tol=1e_4, random_state=41	test_size=100% n_clusters=3 init='kmeans' n_init=10, max_itter=300, tol=1e_4, random_state=42
Resultado Kmeans	Silhouette = 0.45	Silhouette = 0.46	Silhouette = 0.46	Silhouette = 0.42	Silhouette = 0.43
Agrupamiento Aglomerado	test_size=100% n_clusters=3	test_size=100% n_clusters=5	test_size=100% n_clusters=3 linkage='complete'	test_size=100% n_clusters=4	test_size=100% n_clusters=2
Resultado Agrupamiento Aglomerado	Silhouette = 0.97	Silhouette = 0.96	Silhouette = 0.97	Silhouette = 0.96	Silhouette = 0.98
Random Forest	test_size = 0.2 random_state = 42 n_estimators = 100 max_depth = None min_samples_split = 2 max_features = 1	test_size = 0.2 random_state = 0 n_estimators = 100 max_depth = None min_samples_split = 2 max_features =	test_size = 0.2 random_state = 42 n_estimators = 1000 max_depth = 10 min_samples_split = 5 max_features = 0.8	test_size = 0.3 random_state = 42 n_estimators = 100 max_depth = None min_samples_split = 2 max_features = 1	test_size = 0.3 random_state = 42 n_estimators = 50 max_depth = 10 min_samples_split = 5 max_features = 0.8

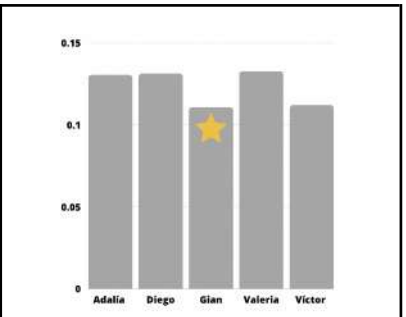
		1			
Resultados Random Forest	Mean Squared Error = 0.1305	Mean Squared Error = 0.1313	Mean Squared Error = 0.1108	Mean Squared Error = 0.1327	Mean Squared Error = 0.1121
SVR	test_size=0.2 kernel='sigmoid', C=1.0, epsilon=0.05	test_size=0.2 kernel='rbf, C=1.0, epsilon=0.15	test_size=0.2 kernel='rbf, C=1.0, epsilon=0.3	test_size=0.2 kernel='rbf, C=1.0, epsilon=0.1	test_size=0.2 kernel='sigmoid', C=1.0, epsilon=0.5
Resultados SVR	Mean Squared Error = 2474	Mean Squared Error = 0.185	Mean Squared Error = 0.17	Mean Squared Error = 0.184	Mean Squared Error = 2461
Red Neuronal	text_size=0.2 activation_function =sigmoid random state=42 epochs=150 batch_size=32	text_size=0.2 activation_functi on=tanh random state=42 epochs=150 batch_size=32	text_size=0.2 activation_functio n=elu random state=42 epochs=150 batch_size=32	test_size=0.2 activation_functio n=softmax random_state=42 epochs=150 batch_size=32	test_size=0.2 activation_functio n=relu random_state=42 epochs=150 batch_size=32 0
Resultados Red neuronal	Mean Squared Error Q11: 0.12 Mean Squared Error Q16: 0.09	Mean Squared Error Q11: 0.12 Mean Squared Error Q16: 0.09	Mean Squared Error Q11: 0.12 Mean Squared Error Q16: 0.09	Mean Squared Error Q11: 0.12 Mean Squared Error Q16: 0.09	Mean Squared Error Q11: 0.14 Mean Squared Error Q16: 0.10

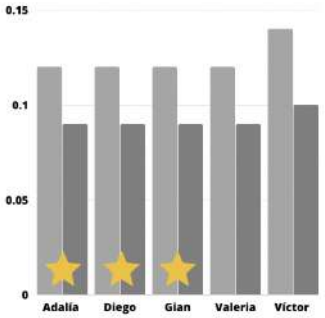
4.7 Análisis de resultados y selección de modelo de predicción

K-Means	Agrupamiento Aglomerado	Random Forest







Support Vector Regressor	Red Neuronal
	

4.8 Justificación de la validez de los modelos

Tras un análisis de los modelos creados individualmente, se obtiene que todos los modelos son correctos y cuentan con métricas positivas, lo que significa que todos los modelos tienen un funcionamiento correcto. Los resultados específicamente son:

- KMeans cuenta con un Silhouette de 0.46
- Agrupamiento Aglomerado cuenta con un Silhouette de 0.98
- Random Forest cuenta con un MSE de 0.1108
- Support Vector Regressor cuenta con un MSE de 0.17
- Red Neuronal cuenta con un MSE de 0.12 y 0.09, para cada pregunta respectiva.

Con esto en mente y considerando las necesidades que se tienen en el proyecto, se establece que la red neuronal es el mejor modelo para predecir la variable objetivo, mientras que el Agrupamiento Aglomerado se muestra esencial para crear un conjunto de datos con similitudes entre sí. Sin embargo, tomando en cuenta el límite de tiempo y las necesidades de la organización socio formadora, se ha decidido solo utilizar la red neuronal y hacer una visualización de las predicciones.

Cabe aclarar que con la red neuronal se tiene un entrenamiento en línea, significando que se analizan los datos en lotes pequeños y en tiempo real, actualizando parámetros continuamente, permitiendo adaptabilidad, eficiencia de uso de recursos, resultados significativos y en tiempo real.

5) Evaluación

5.1 Evaluación de resultados

Después de realizar el proyecto, se tienen dos entregables finales. Primero, el modelo de predicción significará una gran ayuda en la eficiencia del análisis de las respuestas recibidas en la encuesta. Esto, pues, computacionalmente hablando, un análisis de sentimientos es pesado de realizar y la Red Neuronal ya reconoce los patrones y puede predecir los valores. Para la Dirección de Servicio Social representa una gran ayuda y, al mismo tiempo, una mejor organización y entendimiento de los datos.

Además, el dashboard facilita la comprensión de la información recibida, y significa una gran herramienta para los usuarios y directivos que lo utilizarán. Esto, pues es una visualización clara, con filtros y otras cosas, que hacen sencillo ver el comportamiento de cada Organización Socio Formadora, así como las cosas en las que deben de mejorar.

Finalmente, los resultados obtenidos significan una gran ayuda a la manera en que seguirán aplicando encuestas futuras, y, así, el negocio creado por nosotros será parte de la ayuda a seguir facilitando servicio social a los estudiantes del tec. Mejorando la experiencia del estudiante, de la Dirección de Servicio Social y de las OSF’s; representando así una propuesta de valor.

5.2 Revisión del Proceso

La realización del producto final inició al entender las bases de datos que nos otorgó el departamento del Servicio Social, comprender a donde se debía de dirigir la atención y hacer una limpieza extensiva y exhaustiva de los datos. Al aplicar diferentes funciones y tratamientos a los datos, terminamos con un data frame completo, de 10,000 filas y 20 columnas aproximadamente. Con esto, realizamos un análisis de sentimientos y notamos que era muy pesado computacionalmente, era muy tardado y tedioso obtener resultados. Entonces, la decisión fue crear un modelo que se encargara de predecir estos valores para

que la organización socio formadora no lo tuviera que hacer con las respuestas a las encuestas futuras. También, notamos la necesidad de hacer un dashboard para analizar y comprender el comportamiento de cada organización dentro del servicio social, para así ver cómo podían mejorar y buscar la manera de seguir motivando a las mejores.

Entonces, aunque el último modelo de predicción funcionó correctamente, una problemática que nos encontramos fue la cantidad de hiperparámetros que se tenían que modificar con tal de que el resultado fuera positivo. Intentamos diversos tipos de redes neuronales y otros modelos de predicción, pero notamos que el elegido fue el menor costoso computacionalmente y el que mejor predecía resultados. Una posible mejora es buscar la manera de hacerla más complicada para así mejorar aún más la cantidad de aciertos de la predicción.

Otro problema con el que nos enfrentamos fue la complejidad de hacer un análisis de sentimientos que tuviera sentido y que representara el sentimiento real de cada comentario. Los que eran rápidos no eran tan exactos y, los que usaban deep learning eran sumamente tardados pero más precisos. Una posible mejora es encontrar alguna librería de Python de Deep Learning que proporcione una manera de hacer el análisis de sentimientos en español y, aunque sea tardado, recibir una mayor exactitud en los resultados.

Finalmente, nos topamos con el problema del tiempo. Al estar contra reloj habían diferentes pasos que se pudieron haber hecho que, por falta de tiempo, se tuvieron que posponer. Un ejemplo de esto es el agrupamiento aglomerativo, problemática que será mencionada en los siguientes pasos. Una manera de solucionar la problemática del fue aprender a priorizar, organizarnos, y trabajar únicamente en lo necesario para el programa.

5.3 Impacto Social Principal

Con ayuda de nuestro modelo clasificador que puede predecir el grado de satisfacción de las experiencias que brinda la OSF, podremos tomar decisiones informadas sobre los Servicios Sociales que ofrece la organización y medir de forma precisa los puntos vulnerables de los proyectos. Con esto, podremos ir mejorando para que en futuras experiencias de servicio social los alumnos aumenten su grado de satisfacción y al mismo tiempo motivemos a muchas más personas a formar parte del Servicio Social que ofrece la OSF y apoyar a nuestra sociedad.

5.4 Impacto hacia los Objetivos de Desarrollo Sostenible

	El Objetivo de Desarrollo Sustentable que consideramos más propio para este reto y que más se pueda ver impactado es el 17: *"Alianza para lograr los objetivos"*, pues buscamos crear un compromiso con el Socio Formador para ayudar a mejorar sus herramientas y conexiones con los alumnos. Todo esto a fin de innovar con las estrategias que actualmente cuentan, mejorar sus prácticas y tener un mejor conocimiento y entendimiento de las opiniones de las personas.
	El objetivo 4: *'Educación de calidad'* también se ve impactado pues se busca crear un progreso en el ámbito educativo y mejorar oportunidades de aprendizaje tanto de alumnos como de la institución, al aprovechar sus retroalimentaciones.

6) Despliegue

6.1 Descripción del prototipo funcional

Para el despliegue de los resultados, se propone un dashboard interactivo; este se basa en las predicciones hechas por la red neuronal. Recordando que la red neuronal se enfoca en predecir la satisfacción de un usuario con la OSF se encarga de basarse en las respuestas de opción múltiple para predecirlo.

Al final, el dataset utilizado para la generación del dashboard contiene la fecha en la que fue hecha la encuesta, el nombre de la OSF evaluada, y la predicción de la red neuronal para las preguntas 4 y 6, por lo que solo tiene 4 columnas. Fue generado de la siguiente manera:

```
y_predfull=model.predict(X)
y_predfull=pd.DataFrame(y_predfull)
columns_to_extract=['Fecha registrada','OSF']
new_df = df[columns_to_extract].copy()
new_df2 = new_df.drop(df.index[0]).reset_index(drop=True)
dfcomplete2= pd.concat([y_predfull, new_df2], axis=1)
dfcomplete2.to_csv('dfdashboard4.csv')
```

Al tener el set de datos con el que se trabajará el dashboard, se empieza a realizar haciendo uso de la librería de Python llamada *streamlit*. Este dashboard se ha dividido en dos secciones. En la primera se despliegan todas las encuestas y se hace un análisis general. En la segunda se muestra una búsqueda más especializada ya que las visualizaciones solo toman en cuenta las encuestas que cumplen con los filtros del lado izquierdo de la aplicación. Estos filtros son el rango de fecha, la OSF en específico y el puntaje para cada pregunta. El código utilizado se presenta a continuación y está completamente comentado y explicado en las fotografías, pero cabe mencionar que para correr la aplicación se deben descargar las librerías necesarias y se debe escribir en la terminal **sstreamlit run streamlit2.py.**

```
import pandas as pd
import streamlit as st
import plotly.express as px
import datetime
import matplotlib.pyplot as plt
import seaborn as sns
import altair as alt
from bokeh.plotting import figure
import re
from fuzzywuzzy import fuzz
from fuzzywuzzy import process

st.set_page_config(page_title='Dashboard OSFS' #metodo para crear la pagina definir el nombre que aparecera en la pestana del navegador del dashboard
                   ,page_icon='moneybag'
                   ,layout='wide' #definir como se vera la pagina en este caso se vera angosta
                   ,initial_sidebar_state='expanded') #esto define que el tab con los filtros estara abierta cuando se corra la aplicacion al principio


def get_data(): #funcion para cargar el csv con el que trabajaremos

    df=pd.read_csv('dfdashboard4.csv')
    #Limpieza de datos del archhivo con el que trabajaremods
    df = df.iloc[2:]
    df = df.drop('Unnamed: 0', axis=1)
    #convertir la columna de fecha al formato que reconoce streamlit
    df['Fecha registrada']=pd.to_datetime(df['Fecha registrada'], format='ISO8601').dt.date

    df = df.dropna()
    # Convert question_0 and question_1 columns to int or float
    df['0'] = df['0'].astype(float)
    df['1'] = df['1'].astype(float)
    return df
#mandar a llamar la funcion get_data para guardar el df en la variable df
df=get_data()



#determinar valores necesarios para los filtros
df.columns = ['Pregunta1', 'Pregunta2'] + list(df.columns[2:]) #renombrar la columna 0,1 por Pregunta1 Pregunta2
df['Pregunta1'] = df['Pregunta1'] * 100
df['Pregunta1'] = df['Pregunta1'].astype(int) #convertir los puntajes de la pregunta1 una a Integers en el rango de 0-100
df['Pregunta2'] = df['Pregunta2'] * 100
df['Pregunta2'] = df['Pregunta2'].astype(int) #convertir los puntajes de la pregunta2  una a Integers en el rango de 0-100
df = df[df['Pregunta2'] <= 100]
```

```python
df = df[df['Pregunta2'] <= 100]

# Assuming df is your DataFrame
df = df.dropna()
# Convert question_0 and question_1 columns to int or float
df.rename(columns={'Fecha registrada': 'Fecha_Registrada'}, inplace=True)

#valores limites para filtros
min_date = pd.to_datetime(df['Fecha_Registrada']).min() #esta variable es la fecha de la primera encuesta registrada
max_date = pd.to_datetime(df['Fecha_Registrada']).max() #esta variable es la fecha de la ultima encuesta registrada

min_val_pregunta1=int(df['Pregunta1'].min()) #Esta variable el puntaje de la pregunta1 uno mas bajo
max_val_pregunta1=int(df['Pregunta1'].max()) #Esta variable el puntaje de la pregunta1 uno mas alto


min_val_pregunta2=int(df['Pregunta2'].min()) #Esta variable el puntaje de la pregunta2 uno mas bajo
max_val_pregunta2=int(df['Pregunta2'].max()) #Esta variable el puntaje de la pregunta2 uno mas bajo
df = df[df['OSF'] != ""]

df = df[df['OSF'] != '?']


st.title(':clipboard: OSF DASHBOARD') #titulo del dashboard
st.markdown('##') #enter

#Filtros_____
st.sidebar.header("Opciones a filtrar:") #sidebar lo que nos va a hacer es crear en la parte izquierda un cuadro para agregar los filtros que queramos tener
#Este filtro es para escoger las OSF que se deseen visualizar para ciertas graficas
oSF_filtro = st.sidebar.multiselect(
    "Seleccione la OSF:", #mensaje que aparecera arriba de la caja de seleccion
    options = df['OSF'].unique(), #las opciones que se pueden escoger
    default = ["AMMJE, Asociación Mexicana de Mujeres Jefas de Empresa, A.C.","Dirección de Servicio Social - Aprendizajes para todos"]#Aquí podría por default dejar un filtro específico pero vamos a dejarlos to
)
start_date = st.sidebar.date_input('Start Date',min_value=min_date,max_value=max_date,value=min_date) #Este filtro generara un calendario donde el usuario puede escoger la primera fecha para un rango donde ver l
end_date = st.sidebar.date_input('End Date',min_value=min_date,max_value=max_date,value=max_date) #Este filtro generara un calendario donde el usuario puede escoger la segunda fecha para un rango donde ver las e


pregunta1 = st.sidebar.slider( #este filtro sirve para crear un rango de puntajes en las encuestas asi el usuario puede escoger hasta que puntaje desea ver las encuestas
        "Rango de puntaje para Pregunta 1",
        min_value=min_val_pregunta1,
        max_value=max_val_pregunta1,
        value=max_val_pregunta1,
        step=1
)
```

```python
pregunta2 = st.sidebar.slider(
        "Rango de puntaje para Pregunta 2", #este filtro sirve para crear un rango de puntajes en las encuestas asi el usuario puede escoger hasta que puntaje desea ver las encuestas
        min_value=min_val_pregunta2,
        max_value=max_val_pregunta2,
        value=max_val_pregunta2,
        step=1
)

#_____


#crearemos un nuevo df que esta conectado a los filtros para generar diferentes graficas
df_seleccion1=df[ (df['OSF'].isin(oSF_filtro)) & (df['Pregunta1']<pregunta1)& (df['Pregunta2']<pregunta2) ] #conectar los filtros de las osf y del rango de puntos de las dos preguntas
df_seleccion1=df_seleccion1.query("Fecha_Registrada>=@start_date & Fecha_Registrada<=@end_date") # conectar el filtro de las fechas




# Esta funcion grafica todas las  predicciones de todas las encuestas, siempre permanecera igual esta conectada al df sin filtro
def satis(df):
    df['Fecha_Registrada'] = pd.to_datetime(df['Fecha_Registrada'])

    alt.data_transformers.disable_max_rows()

    brush = alt.selection_interval()
    points = alt.Chart(df).mark_point().encode(
        x='Fecha_Registrada',
        y=alt.Y('Pregunta1', title='Satisfaccion'),
        color=alt.condition(brush, 'OSF', alt.value('lightgray'))
    ).add_params(
        brush
    )

    st.write(points)
```

```python
    def satis2(df):
        df['Fecha_Registrada'] = pd.to_datetime(df['Fecha_Registrada'])

        alt.data_transformers.disable_max_rows()

        brush = alt.selection_interval()
        points = alt.Chart(df).mark_point().encode(
            x='Fecha_Registrada',
            y=alt.Y('Pregunta2', title='Satisfaccion'),
            color=alt.condition(brush, 'OSF', alt.value('lightgray'))
        ).add_params(
            brush
        )

        st.write(points)


    st.subheader('Todas las Encuestas')
    col3, col4 = st.columns(2)

    with col3: #ingresar datos necesarios para la columna
        st.header('Encuestas Pregunta 1') #titulo de la columna 1
        satis(df)  # mandar a llamar la funcion con graficas para la pregunta 1

    with col4: #ingresar datos necesarios para la columna
        st.header('Encuestas Pregunta 2') #titulo de la columna 2
        satis2(df) # mandar a llamar la funcion con graficas para la pregunta 2

    st.divider()  # 👈 Draws a horizontal rule
```

```python
def mejoresOSF(df): # el proposito de esta funcion es desplegar en texto las 5 OSF con mayor puntaje en la prediccion
    #esta grafica toma en cuenta todas las encuestas y no utiliza el df con filtros
    df['Fecha_Registrada'] = pd.to_datetime(df['Fecha_Registrada'])
    #crear periodos de tiempo escolares
    num_sections = 3
    min_date = df['Fecha_Registrada'].min()
    max_date = df['Fecha_Registrada'].max()
    date_range = max_date - min_date
    section_length = date_range / num_sections

    df['section'] = ((df['Fecha_Registrada'] - min_date) / section_length).astype(int)
    #sacar el promedio para cada periodo de tiempo de satisfaccion
    df_means = df.groupby('section')['Pregunta1'].mean().reset_index()
    df_means_sorted = df_means.sort_values('Pregunta1', ascending=False)

    # Initialize an empty DataFrame to store the top 5 OSF for each time period
    top_osf_per_period = pd.DataFrame(columns=['section', 'OSF', 'Mean_Score'])

    # Extract top 5 OSF for each time period
    for section in df['section'].unique():
        section_data = df[df['section'] == section]
        section_means = section_data.groupby('OSF')['Pregunta1'].mean().reset_index()
        section_means_sorted = section_means.sort_values('Pregunta1', ascending=False).head(5)
        section_means_sorted['section'] = section
        top_osf_per_period = pd.concat([top_osf_per_period, section_means_sorted])

    # Reset the index of the resulting DataFrame
    top_osf_per_period.reset_index(drop=True, inplace=True)

    # Display the names for each time period in Streamlit
    for section in df['section'].unique():
        st.markdown(f"**Top 5 OSF Para el Periodo  {section}:**")
        osf_list = top_osf_per_period[top_osf_per_period['section'] == section]['OSF'].tolist()
        for osf in osf_list:
            st.markdown(f"- {osf}")


mejoresOSF(df)
```

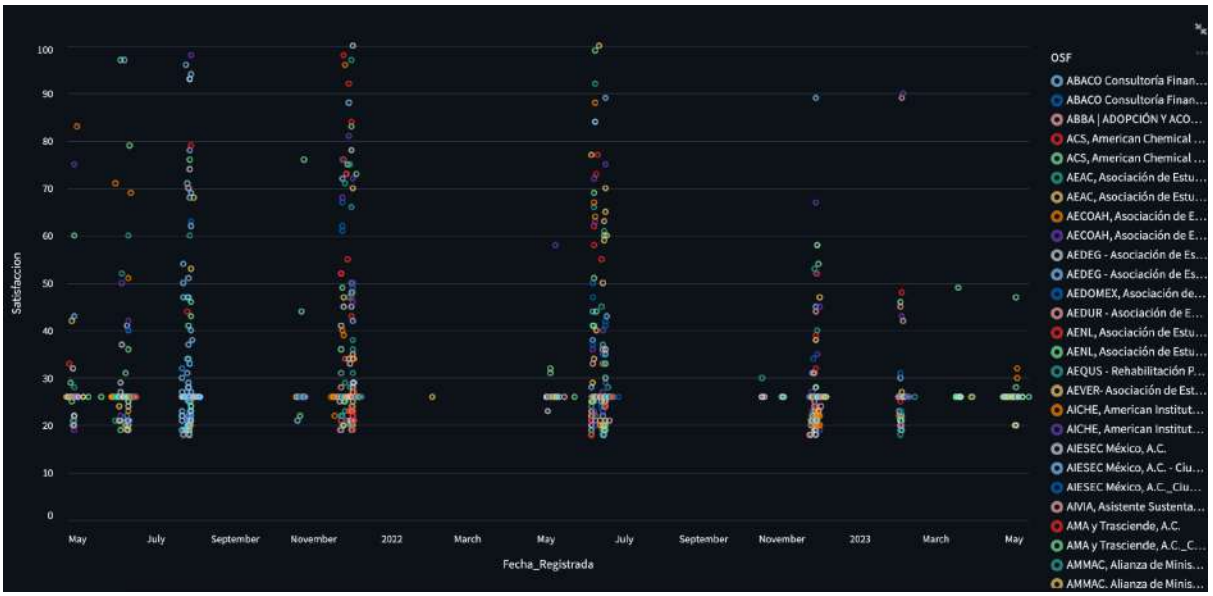
```python
def studentsatisfaction1(df): # el proposito de esta funcion es crear una grafica con la satisfaccion promedio de los estudiantes en cada periodo de tiempoe escolar
    #esta grafica toma en cuenta todas las encuestas y no utiliza el df con filtros
    df['Fecha_Registrada'] = pd.to_datetime(df['Fecha_Registrada'])
    #crear periodos de tiempo escolares
    num_sections = 2
    min_date = df['Fecha_Registrada'].min()
    max_date = df['Fecha_Registrada'].max()
    date_range = max_date - min_date
    section_length = date_range / num_sections

    df['section'] = ((df['Fecha_Registrada'] - min_date) / section_length).astype(int)
    #sacar el promedio para cada periodo de tiempo de satisfaccion
    df_means = df.groupby('section')['Pregunta1'].mean().reset_index()

    #crear grafica de linea para demostrar la satisfaccion
    base = alt.Chart(df_means).mark_point().encode(
        x='section',
        y='Pregunta1'

    )
    st.write(base) # mandar a llamar grafica


st.subheader('Promedio Satisfaccion de los Estudiantes')
studentsatisfaction1(df) # mandar a llamar grafica

st.divider()  # 👈 Draws a horizontal rule
st.divider()  # 👈 Draws a horizontal rule

#crear una seccion para los dfs seleccionadas por filtros
st.title('Busqueda con Filtros')
st.divider()
st.subheader('Encuestas seleccionadas con Filtros:')
st.dataframe(df_seleccion1) #mostrar el df de las encuestas

st.divider()
```

```python
#esta funcion graficara las predicciones de la pregunta 1 para las osf seleccionadas y segun los filtros seleccionados, esta grafica es interactiva
def pregunta1(df):
    df['Fecha_Registrada'] = pd.to_datetime(df['Fecha_Registrada'])

    alt.data_transformers.disable_max_rows()

    brush = alt.selection_interval()
    #esta parte crea un scatter plot de todas las predicciones de la pregunta 1
    points = alt.Chart(df).mark_point().encode(
        x='Fecha_Registrada',
        y='Pregunta1',
        color=alt.condition(brush, 'OSF', alt.value('lightgray'))
    ).add_params(
        brush
    ) #esta funcion crea un bar chart con el numero de encuestas de cada osf en ese rango de tiempo
    bars = alt.Chart(df).mark_bar().encode(
        y='OSF',
        color='OSF',
        x='count(OSF)'
    ).transform_filter(
        brush
    )
    st.write(points & bars) # crea la grafica

#esta funcion graficara las predicciones de la pregunta 2  para las osf seleccionadas y segun los filtros seleccionados, esta grafica es interactiva
```

```python
def pregunta2(df):
    df['Fecha_Registrada'] = pd.to_datetime(df['Fecha_Registrada'])

    alt.data_transformers.disable_max_rows()

    brush = alt.selection_interval()
        #esta parte crea un scatter plot de todas las predicciones de la pregunta 1

    points = alt.Chart(df).mark_point().encode(
        x='Fecha_Registrada',
        y='Pregunta2',
        color=alt.condition(brush, 'OSF', alt.value('lightgray'))
    ).add_params(
        brush
    ) #esta funcion crea un bar chart con el numero de encuestas de cada osf en ese rango de tiempo
    bars = alt.Chart(df).mark_bar().encode(
        y='OSF',
        color='OSF',
        x='count(OSF)'
    ).transform_filter(
        brush
    )
    st.write(points & bars) # crea la grafica
#crear dos columnas en el df uno para cada grafica
col1, col2 = st.columns(2)

with col1: #ingresar datos necesarios para la columna
    st.header('Puntaje Pregunta 1') #titula de la columna 1
    pregunta1(df_seleccion1)  # mandar a llamar la funcion con graficas para la pregunta 1

with col2: #ingresar datos necesarios para la columna
    st.header('Puntaje Pregunta 2') #titula de la columna 2
    pregunta2(df_seleccion1) # mandar a llamar la funcion con graficas para la pregunta 2

st.divider()  # 👈 Draws a horizontal rule
```

El dashboard terminado se ve de la siguiente manera:

La sección de la parte izquierda es desplegable y es en donde se encuentran los filtros que funcionan para la sección de búsqueda especializada que se mostrará a continuación. Las gráficas que se ven se pueden ampliar y son interactivas.



En la siguiente sección es posible ver a las OSF mejor calificadas en cada periodo académico:

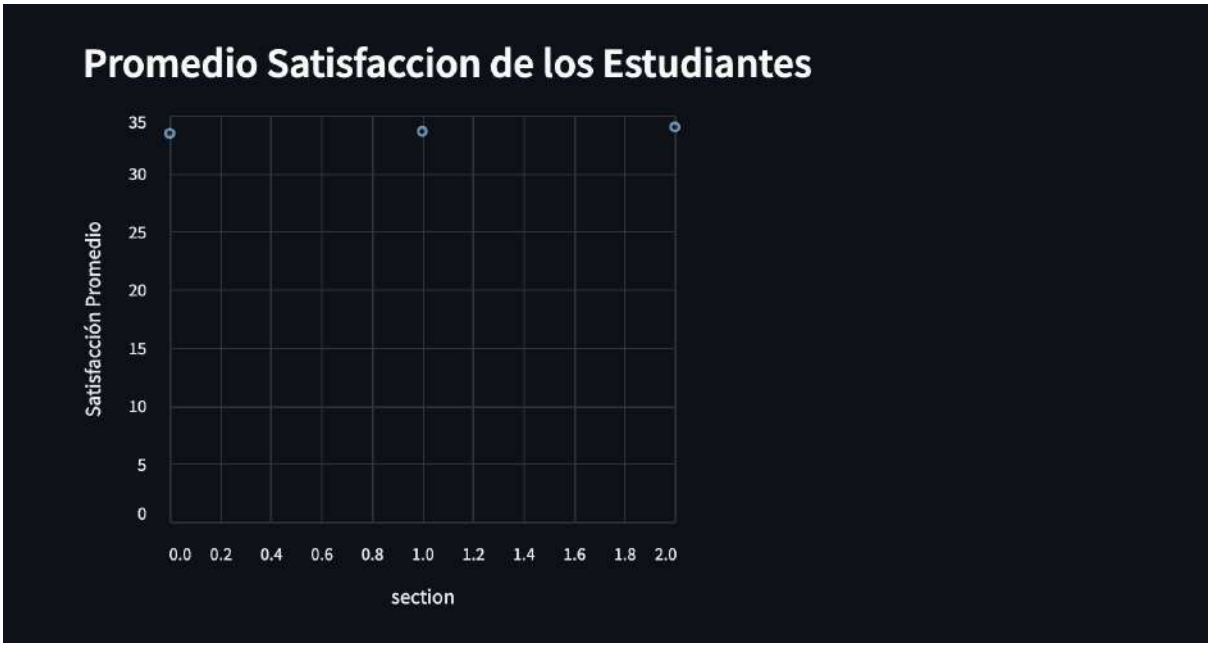
Top 5 OSF Para el Periodo 0:

- Fundación Rolivel, A.C.
- Fundación El Mundo Escribe, A.C.
- Iniciativa Campana Altamira
- Guerreros en la Vida y en la Cancha, A.C.
- Fundación Yepez, A.C.

Top 5 OSF Para el Periodo 1:

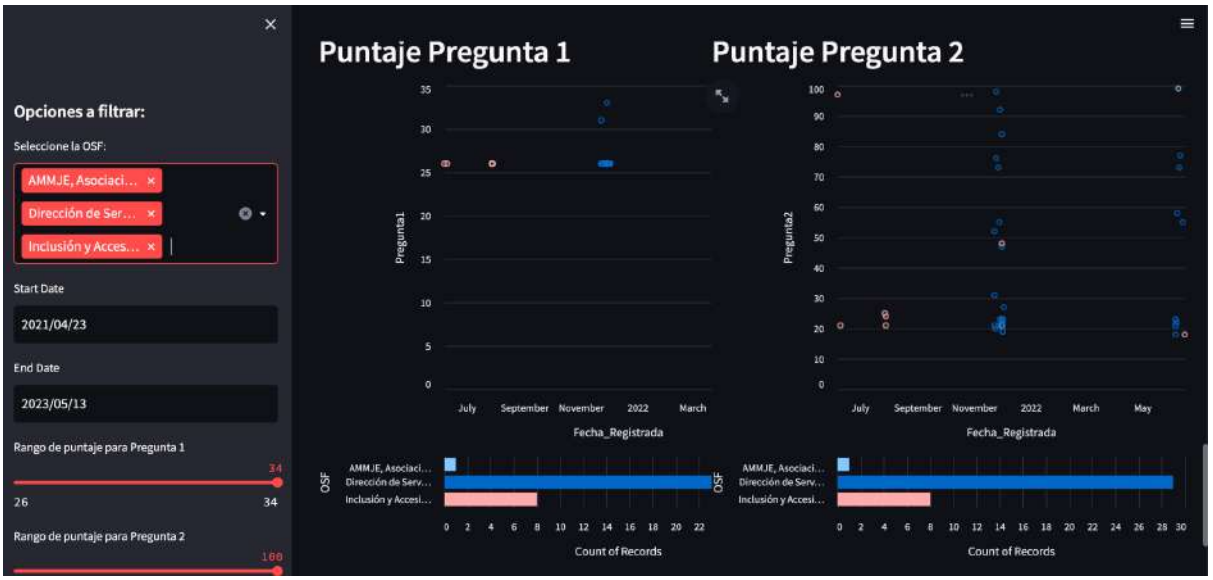
- ABACO Consultoría Financiera
- Instituto Nacional de Antropología e Historia
- PRODAN, Prodefensa Animal, A.C.
- Neurofinanzas, A.C.
- Museo de Historia Mexicana

La siguiente gráfica muestra el promedio de satisfacción de la predicción que se ha realizado a través de diferentes periodos escolares.



Más adelante, está la parte de búsqueda especializada que es afectada por los filtros. En esta parte se despliega el data frame con las encuestas que cumplen el criterio establecido.

Después, se presenta la predicción de satisfacción para la primera pregunta abierta mediante un scatterplot y una gráfica de barras con el número de encuestas para las que la OSF cumple, dentro de los criterios de los filtros. La pregunta 2 muestra los resultados de la segunda pregunta abierta.



En conclusión, este dashboard proporciona, al socio formador, una herramienta que permite entender de una manera más sencilla las respuestas de los usuarios. Permite tener una idea general de cómo los estudiantes se sienten con el servicio social que hicieron y, de tal manera, facilita las siguientes acciones que tomarán los directivos.

7) Recomendaciones

7.1 Recomendaciones al socio formador

Se sugiere aprovechar un modelo de predicción de satisfacción de clientes para identificar áreas de mejora en las experiencias de servicio social. Analizando las clasificaciones que predice el modelo, se pueden realizar ajustes que potencien la calidad y la experiencia del alumnado. Asimismo, se aconseja segmentar a las organizaciones socio formadoras con base a sus niveles de satisfacción previstos y personalizar la interacción para satisfacer sus necesidades específicas. Esto incluye ofrecer promociones personalizadas, recomendaciones adaptadas y comunicaciones enfocadas, lo que generará una mayor satisfacción y fidelidad. Lo anterior, con el propósito de que las organizaciones brinden a los alumnos una buena experiencia dentro de las experiencias y así mismo motivar a las personas a participar en ellas.

Adicionalmente, para obtener el máximo provecho de la información proporcionada por el modelo de predicción, se sugiere establecer un ciclo continuo de retroalimentación y mejora. Recopilar regularmente datos y opiniones de los alumnos y las OSFs permitirá mantener el modelo actualizado y adaptado a las necesidades cambiantes. Además, utilizar esos datos para generar informes y análisis detallados sobre la satisfacción de las empresas brindará una visión holística de las fortalezas y debilidades del programa de servicio social. Esta información valiosa se puede utilizar para desarrollar programas de capacitación específicos para las empresas, brindar recomendaciones personalizadas y promover el crecimiento y la mejora continua en todas las áreas relacionadas con el servicio social.

Al implementar estas recomendaciones, se podrá mejorar la calidad del servicio social ofrecido a las empresas, fortalecer las relaciones con ellas y proporcionar una experiencia enriquecedora para los alumnos, al tiempo que se fomenta una cultura de mejora continua en todos los aspectos relacionados con el servicio social.

7.2 Recomendaciones técnicas

Los valores nulos siempre significan una pérdida en la cantidad de información que se tiene. La primera recomendación técnica que se realiza a la organización socio formadora es hacer que las preguntas sean obligatorias de responder. Así, se evitarían las respuestas blanco y habría una mayor riqueza en las bases de datos.

Debido a que hay algunas respuestas que se contradicen entre sí (cuando los comentarios son positivos pero las respuestas de opción múltiple negativas, o viceversa), la segunda recomendación es agregar un mínimo de tiempo en que el alumno puede responder. Así, habría más oportunidad de que en realidad piense sus respuestas y no responda la encuesta al azar.

8) Siguientes pasos

La parte más importante es seguir trabajando y mejorando los modelos que realizamos, así como tomar el tiempo de hacer un modelo de agrupamiento aglomerativo que permita identificar a grupos que comparten similitudes entre sí. Esto, sería agregado al dashboard y significaría una propuesta aún más completa que la que se ha dado durante este reporte. De igual manera, el equipo considera que el resultado fue satisfactorio, pues se consiguió la meta de hacer un análisis de datos completo, con un modelo predictivo y una manera eficiente de visualizar los datos.

9) Conclusiones

En este proyecto, se ha seguido el modelo CRISP, un enfoque estándar utilizado en minería de datos y análisis de datos. Las etapas del modelo CRISP, desde la comprensión del negocio hasta el despliegue de los resultados, han sido fundamentales para el éxito de este proyecto. La primera etapa, *la comprensión del negocio*, nos permitió establecer los objetivos y requisitos del proyecto. Fue crucial entender el contexto del Servicio Social y los problemas que queríamos abordar, como la evaluación de la experiencia de los alumnos en las Organizaciones Sociales de Formación (OSF). Esto nos proporcionó una base sólida para el desarrollo de nuestro proyecto y a partir de este momento podríamos comenzar a idear una estrategia para atacar la problemática.

Por consiguiente, la *comprensión de los datos* fue de vital importancia para obtener una visión general de los datos disponibles y su calidad. Realizamos un análisis exploratorio para identificar patrones y tendencias, así como para comprender las relaciones entre las variables relevantes. Esto nos permitió tomar decisiones informadas sobre la preparación y transformación de los datos. Terminada la segunda etapa, comenzamos con la *preparación de los datos*, realizamos limpieza, integración y selección de características relevantes. Eliminamos valores atípicos y datos faltantes, y combinamos múltiples fuentes de datos para obtener una visión más completa. Además, seleccionamos las características más

relevantes para nuestro análisis. Esta etapa fue esencial para garantizar que los datos estuvieran en un estado adecuado para su posterior análisis.

Además, en la *etapa de modelado* se plantearon diferentes modelos de ML para obtener una comprensión más profunda de los datos y llegar a predecir las percepciones de los alumnos sobre sus experiencias con las OSF que forman parte de los colaboradores del departamento de Servicio Social. Durante esta etapa se utilizan modelos del tipo supervisado y no supervisado, dentro de los *modelos propuestos encontramos modelos de K-Means, SVM, Agrupación Aglomerada, Random Forest Regresor y Redes Neuronales.* Con ayuda de estos modelos logramos predecir de forma efectiva las percepciones de los alumnos con sus experiencias de servicio social. Los modelos más efectivos para lograr lo exterior fueron los modelos de Agrupación Aglomerada y la Red Neuronal dando resultados muy acertados de clustering como de predicción respectivamente.

También, durante la *etapa de evaluación y despliegue*, se creó una interfaz clara y visualmente atractiva para presentar los resultados a los usuarios y directivos del socio formador. Se utilizaron los datos generados durante el análisis para facilitar el entendimiento y permitir a los usuarios tomar decisiones informadas. Esta etapa se mostró fundamental para garantizar que los resultados son accesibles y útiles para el Departamento de Servicio Social del Tecnológico de Monterrey.

En conclusión, la inteligencia artificial y los modelos de machine learning han adquirido una importancia fundamental en nuestra sociedad. Estas tecnologías han revolucionado la forma en que interactuamos con la información y han transformado diferentes sectores. Los modelos de machine learning, alimentados por grandes volúmenes de datos, son capaces de analizar patrones complejos y generar predicciones precisas, lo que impulsa la toma de decisiones más informadas y eficientes. La inteligencia artificial y los modelos de machine learning continúan evolucionando rápidamente, promoviendo el avance científico y tecnológico, y se han convertido en elementos clave para enfrentar los desafíos y aprovechar las oportunidades en un mundo cada vez más digitalizado.

Finalmente, este proyecto fue un claro ejemplo de la importancia del trabajo en equipo. A través de la colaboración, comunicación efectiva y aprovechamiento de las fortalezas individuales, se lograron alcanzar resultados superiores a los que se lograrían de manera individual. El trabajo en equipo fomenta la creatividad, resolución de problemas eficiente y fortalecimiento de las relaciones interpersonales, generando un ambiente de confianza y apoyo mutuo. En un mundo cada vez más interdependiente, el trabajo en equipo se vuelve esencial para enfrentar desafíos y lograr metas de manera efectiva.

9) Fuentes bibliográficas

Arora, S. (2022, July 7). *Sentiment Analysis Using Python*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2022/07/sentiment-analysis-using-python/

Brisebois, R., Nadembega, A., Abran, A., & N'techobo, P. (2017). A Semantic Metadata Enrichment Software Ecosystem based on Sentiment and Emotion Metadata Enrichments. *International Journal of Scientific Research in Science, Engineering and Technology*, *3*.

Objetivos de Desarrollo Sostenible | Programa De Las Naciones Unidas Para El Desarrollo. (s. f.). UNDP. https://www.undp.org/es/sustainable-development-goals/alianza-para-lograr-los-objetivos

Rosenbrock1, G. (2021, November 3). *(PDF) Técnicas de Análisis de Sentimientos Aplicadas a la Valoración de Opiniones en el Lenguaje Español*. ResearchGate. Retrieved May 22, 2023, from https://www.researchgate.net/publication/355887680_Tecnicas_de_Analisis_de_Sentimientos_Aplicadas_a_la_Valoracion_de_Opiniones_en_el_Lenguaje_Espanol

Shukla, A., Bhosale, S., & Boob, V. (2017). A Review of Sentiment Analysis Techniques in Social Media Analytics. International Journal of Computer Applications, 171(3), 13-16.

Liu, X., & Zhang, H. (2017). Categorical data analysis using the neural network approach. Knowledge-Based Systems, 135, 179-186.

Sebastiani, F. (2002). Text classification algorithms: A survey. ACM Computing Surveys, 34(1), 1-47.

Thelwall, M., Buckley, K., & Paltoglou, G. (2012). Improving the quality of online reviews using sentiment analysis and a hybrid approach. Journal of the Association for Information Science and Technology, 63(3), 586-599.

Süerdem, A., & Kaya, E. (2015). Using Sentiment Analysis to Detect Customer Attitudes in Social Media Comments. İstanbul Bilgi University, İstanbul, Turkey. Recuperado de: https://www.rcs.cic.ipn.mx/2015_90/Using%20Sentiment%20Analysis%20to%20Detect%20Customer%20Attitudes%20in%20Social%20Media%20Comments.pdf

¿Qué es el análisis de sentimiento? (s. f.). TIBCO Software. https://www.tibco.com/es/reference-center/what-is-sentiment-analysis