Diagrama

Descripción generada automáticamente

**Proyecto Final**

Brayan Valdes, Oscar Merino

Ingeniería Estadística- Escuela Colombiana de Ingeniero Julio Garavito

Modelos de Regresión

**Abstract**

Este proyecto se centra en el desarrollo de una aplicación móvil para el sistema operativo Android, cuyo objetivo principal es ofrecer herramientas de análisis de datos accesibles desde el dispositivo del usuario. La aplicación permite cargar archivos CSV, seleccionar variables dependientes e independientes, y aplicar distintos métodos analíticos según las necesidades del usuario.

Entre sus funcionalidades, se encuentra el análisis de clustering mediante una variante optimizada del algoritmo K-means, adecuada para ejecutarse en dispositivos móviles con recursos limitados. Adicionalmente, la aplicación incorpora modelos de aprendizaje supervisado como la regresión lineal múltiple, para predecir variables dependientes numéricas, y el análisis discriminante lineal (LDA), para predecir variables categóricas. Los resultados se presentan de forma clara en la interfaz, incluyendo métricas de evaluación relevantes como R² en regresión y Accuracy, Precision, Recall y F1-score en clasificación.

El desarrollo de la aplicación requirió superar múltiples desafíos técnicos debido al uso de Kotlin, lenguaje que carecen de soporte nativo para procesamiento estadístico avanzado. Para ello, se integraron scripts en Python mediante Chaquopy, y se diseñaron estructuras de datos eficientes que permiten ejecutar los modelos de forma completamente offline. Si bien la app no realiza limpieza automática de datos, proporciona al usuario una interfaz intuitiva para explorar y seleccionar los datos que desea analizar.

Esta aplicación representa una herramienta versátil que combina clustering y modelado predictivo en un entorno móvil, sin depender de conectividad a internet. Su desarrollo resalta la importancia de la adaptabilidad y la creatividad para integrar tecnologías diversas, superando las limitaciones del entorno y permitiendo un aprendizaje práctico en análisis de datos y programación multiplataforma.

1. **Introducción**

El análisis de clustering es una técnica fundamental en minería de datos y aprendizaje no supervisado, utilizada para agrupar objetos en subconjuntos homogéneos basados en características comunes. Este enfoque tiene aplicaciones en diversas áreas, como segmentación de mercados, análisis de patrones de comportamiento, y optimización de recursos en sistemas. Uno de los métodos más ampliamente utilizados es el algoritmo K-means, que asigna objetos a un número predefinido de clústeres minimizando la variación intra-cluster y maximizando las diferencias inter-cluster (MacQueen, 1967). Sin embargo, su implementación en dispositivos móviles presenta retos asociados a las limitaciones de hardware y software.

El presente proyecto aborda estos desafíos mediante el desarrollo de una aplicación en Android Studio, utilizando Kotlin como lenguaje de programación e integrando Python mediante Chaquopy para ejecutar algoritmos estadísticos. Android Studio, el entorno oficial de desarrollo para aplicaciones Android, ofrece herramientas potentes para la creación de interfaces gráficas y manejo de archivos, pero carece de soporte nativo para operaciones avanzadas como clustering o modelado predictivo (Android Developers, 2024). En este sentido, la ausencia de bibliotecas especializadas en Java para machine learning implicó la necesidad de desarrollar estructuras de datos personalizadas e integrar soluciones externas para el análisis.

El proyecto se centró en construir una aplicación que permita a los usuarios cargar conjuntos de datos en formato CSV desde sus dispositivos, explorar las variables contenidas y aplicar distintos métodos analíticos según sus necesidades. El usuario es quien determina qué variables utilizar. Esta decisión busca mantener el control sobre el análisis en manos del usuario, favoreciendo la transparencia y flexibilidad del proceso.

Entre las funcionalidades disponibles, se destaca el módulo de clustering mediante una variante optimizada del algoritmo K-means, diseñado para ejecutarse eficientemente en dispositivos móviles con recursos limitados. Adicionalmente, la aplicación incorpora algoritmos de aprendizaje supervisado como la regresión lineal múltiple, que permite predecir variables numéricas a partir de múltiples atributos, y el análisis discriminante lineal (LDA), útil para predecir categorías o clases. En cada caso, el usuario selecciona las variables independientes y dependientes, ejecuta el modelo y obtiene resultados interpretables en la interfaz.

Los resultados incluyen tanto predicciones individuales como métricas de evaluación. En el caso de la regresión, se presentan valores como el coeficiente de determinación R² y errores estándar; mientras que para clasificación con LDA se calculan métricas como precisión, exactitud (accuracy), sensibilidad (recall) y F1-score. Todo el procesamiento se realiza localmente en el dispositivo, sin necesidad de conexión a internet.

Además, este proyecto subraya la importancia de diseñar interfaces de usuario (UI) intuitivas y accesibles, garantizando que incluso los usuarios sin conocimientos técnicos puedan interactuar fácilmente con la aplicación. La UI guía a los usuarios a lo largo de todo el proceso, desde la carga de datos y la selección de parámetros hasta la visualización de resultados en gráficos y textos generados dinámicamente. Este enfoque facilita una experiencia comprensible y fluida para un público amplio, reforzando el propósito práctico de la herramienta.

En términos teóricos, este trabajo se fundamenta en algoritmos de clustering y modelado predictivo, descritos en textos clave de minería de datos como los de Tan, Steinbach y Kumar (2018). La implementación en un entorno móvil resalta las oportunidades de aplicar técnicas de análisis de datos en contextos cotidianos, acercando estas herramientas a un público más amplio. Al democratizar el uso del análisis de datos, la aplicación amplía su impacto práctico, permitiendo que más personas accedan a soluciones avanzadas para interpretar y comprender información compleja de forma autónoma y offline.

**Objetivos**

**Objetivo General:**

Desarrollar una aplicación móvil en Android Studio que permita realizar análisis de clustering, regresión lineal múltiple y clasificación mediante LDA, proporcionando una herramienta funcional para analizar conjuntos de datos en formato CSV desde dispositivos móviles sin requerir conexión a internet.

**Objetivos Específicos:**

* Desarrollar un sistema que permita cargar y explorar archivos CSV directamente en dispositivos Android, ofreciendo al usuario la posibilidad de seleccionar las variables independientes y dependientes para cada tipo de análisis.
* Validar la funcionalidad y precisión del sistema mediante pruebas con conjuntos de datos variados, asegurando resultados confiables y una experiencia fluida en dispositivos con recursos limitados.
* Implementar una versión optimizada del algoritmo K-means en Java, adaptada al entorno móvil, junto con la integración de modelos estadísticos desarrollados en Python (regresión lineal múltiple y LDA) mediante Chaquopy, garantizando eficiencia en procesamiento y compatibilidad multiplataforma.

1. **Material Utilizado**

**Dispositivos utilizados:**

Los siguientes dispositivos fueron utilizados para desarrollar, probar y validar la funcionalidad del sistema, el rendimiento de los algoritmos estadísticos y la compatibilidad en distintos entornos operativos. La aplicación desarrollada implementa herramientas de análisis como K-means para clustering, Regresión Lineal Múltiple y Análisis Discriminante Lineal (LDA) para predicción.

* **Dispositivos móviles**

Estos equipos fueron empleados para validar la ejecución y desempeño de la app en diversos entornos Android, evaluando la compatibilidad de instalación, procesamiento de datos y visualización de resultados en distintas capas de personalización de Android. Además, se buscó verificar la compatibilidad cruzada entre Android estándar y sistemas alternativos como HarmonyOS (Huawei):

**OPPO A38**

Android 14 con ColorOS 14.0 / RAM 4 GB (expandible a 8 GB con RAM virtual) / Memoria interna 128 GB / Procesador MediaTek Helio G85.

Objetivo: Validar la ejecución fluida del algoritmo K-means y los modelos predictivos LDA y regresión lineal en versiones recientes de Android con hardware

**Xiaomi Redmi Note 8**

Android 11 con MIUI 12.5.2 / RAM 4 GB / Memoria interna 64 GB / Procesador Qualcomm Snapdragon 665.

Objetivo: Evaluar compatibilidad y rendimiento en dispositivos con versiones anteriores de Android y hardware más limitado, validando especialmente la visualización de resultados y capacidad de procesamiento local.

* **Computadores personales**

Utilizados en las fases de diseño, programación, entrenamiento de modelos y pruebas de robustez. Aquí se implementaron los algoritmos, se diseñaron las interfaces gráficas, y se realizaron pruebas de carga para evaluar la eficiencia del procesamiento estadístico.

**OMEN 16**

Windows 11 (Versión 22H2) / RAM 16 GB / SSD 512 GB / Procesador Intel Core i7-12700H.

Objetivo: Desarrollo completo de la app, ejecución de pruebas estadísticas avanzadas, análisis de rendimiento de los modelos predictivos y simulaciones con grandes volúmenes de datos.

**Dell Inspiron 15**

Windows 11 (Versión 22H2) / RAM 16 GB / SSD 1 TB / Procesador Ryzen 5.

Objetivo: Validación del sistema en una máquina de especificaciones intermedias, con énfasis en pruebas de optimización del algoritmo K-means y regresión lineal múltiple.

**Entorno de Desarrollo**

Se utilizó Android Studio como entorno principal para programar, depurar e implementar la aplicación móvil en Kotlin, permitiendo el desarrollo completo del sistema, el diseño de interfaces gráficas y la gestión de recursos para múltiples dispositivos Android.

Además, se integraron modelos estadísticos como K-means, Regresión Lineal Múltiple y Análisis Discriminante Lineal (LDA) utilizando scripts en Python, conectados mediante Chaquopy para permitir la ejecución de cálculos estadísticos complejos desde el entorno Android.

Este entorno facilitó:

* La conexión entre la lógica de interfaz (Kotlin/Java) y los modelos estadísticos (Python).
* El diseño de una experiencia de usuario intuitiva para seleccionar variables y visualizar resultados.
* La compatibilidad con diferentes versiones de Android y capas de personalización de fabricantes.

**Dependencias y programas:**

* AndroidX AppCompat (androidx.appcompat:appcompat:1.6.1):

Proporciona compatibilidad con versiones anteriores de Android y componentes esenciales para el desarrollo de aplicaciones modernas.

* RecyclerView (androidx.recyclerview:recyclerview:1.3.2)

Componente para mostrar listas o tablas de datos de manera eficiente.

* MPAndroidChart (com.github.PhilJay:MPAndroidChart:v3.1.0)

Librería para crear gráficos interactivos, utilizada para generar gráficos de dispersión en los resultados de clustering.

* Apache Commons Math (org.apache.commons:commons-math3:3.6.1)

Proporciona la implementación del algoritmo de clustering K-means, además de otras herramientas matemáticas necesarias para el análisis de datos.

* Material Design (com.google.android.material:material:1.8.0)

Permite construir interfaces modernas e intuitivas siguiendo las guías de diseño de Google.

* AndroidConstraintLayout(androidx.constraintlayout:constraintlayout:2.1.4)

Herramienta para diseño flexible de interfaces gráficas en Android.

* Apache POI (org.apache.poi:poi:5.2.3 y org.apache.poi:poi-ooxml:5.2.3)

Librería para leer y manipular archivos Excel en formato XLSX.

* JUnit (junit:junit:4.13.2)

Framework para realizar pruebas unitarias en Java.

* Espresso (androidx.test.espresso:espresso-core:3.5.1)

Herramienta para pruebas de interfaz de usuario en aplicaciones Android.

* Chaquopy (com.chaquo.python:gradle-plugin)

Plugin para integrar código Python en aplicaciones Android y ejecutar bibliotecas de Python mediante Chaquopy.

* NumPy (numpy)

Biblioteca de Python para cálculo numérico eficiente, especialmente con matrices y álgebra lineal.

* Pandas (pandas)

Biblioteca de Python para análisis y manipulación de datos, especialmente en estructuras tabulares como DataFrames.

* SciPy (scipy)

Biblioteca de Python utilizada para cálculos científicos y técnicos, complementaria a NumPy.

* Statsmodels (statsmodels)

Módulo de Python para la estimación de modelos estadísticos, pruebas y análisis de datos.

* Scikit-learn (scikit-learn)

Biblioteca de aprendizaje automático en Python que proporciona herramientas para clasificación, regresión y clustering.

1. **Metodología**

A continuación, se describen en detalle los procedimientos y pasos seguidos para implementar la aplicación móvil de clustering en Android Studio utilizando Kotlin.

**Configuración Inicial del proyecto:**

Se creó un proyecto en Android Studio con las dependencias necesarias.

Se configuraron las vistas principales en XML y se inicializaron los controladores en archivos Java.

**Creación del archivo MainActivity.kt:**

Este archivo representa el back-end de la primera página de la app.

Componentes:

* Un botón para seleccionar un archivo CSV.
* Otro botón para navegar a la página de resumen básico estadístico.

Funcionalidad global.

* Permite al usuario cargar un archivo CSV
* Validación de los datos del archivo para asegurar que cumplen los requisitos de formato.
* Uso de la clase TableAdapter.java para mostrar los datos del archivo en formato tabular en la página principal.

**Creación del archivo TableAdapter.kt:**

Este archivo gestiona la presentación de los datos cargados en una tabla.

Implementación:

* Uso de RecyclerView para crear una tabla que muestre los valores del archivo CSV.
* Diseño de columnas dinámicas para adaptarse al número de columnas en el archivo cargado.

**Creación del archivo ResumenEstadistico.kt:**

Este archivo controla la página de resumen basico estadístico.

Componentes:

* Una tabla que muestra el resumen estadístico básico de cada columna numérica, con estadísticas descriptivas tales como: nombre de la columna, media, valor mínimo, valor máximo y desviación estándar.
* Un TextView que verifica si todas las columnas contienen datos continuos, en caso de no serlo, se omite la columna en el análisis estadístico.
* Un botón que lleva a una pagina para seleccionar el número de clústeres.
* Un botón que lleva a una pagina la cual permite seleccionar la variable dependiente para el modelo LDA.
* Un botón que lleva a la pagina la cual permite seleccionar no solo la variable dependiente sino que también las variables independientes para el modelo de regresión lineal múltiple.

Implementación de estadísticas básicas:

* Los cálculos se realizan con las funciones matemáticas del paquete Apache Commons Math de java que tiene compatibilidad con kotlin, en el caso de las estadísticas que no se encuentran en estas librerías, como la desviación estándar, se deben generar manualmente.
* La presentación de las estadísticas utiliza un TableLayout con filas dinámicas generadas en tiempo de ejecución.

**Creación de PreClusterActivity.kt:**

Este archivo gestiona la página donde el usuario selecciona el número de clústeres.

Componentes:

* Un TextView que presenta recomendaciones para el número de clústeres, basadas en el método de Elbow.
* Un campo de entrada donde el usuario ingresa el número de clústeres deseados.
* Un botón que lleva a la página donde se puede visualizar el grafico de los clústeres y las coordenadas de los centroides de ellos.

Cálculo y presentación de las recomendaciones:

* Las métricas se deben calcular creando clases que permitan generar los diferentes índices, dadas las limitaciones del lenguaje en cuanto a funciones estadísticas de este tipo, se deben implementar teniendo el conocimiento matemático de cada uno de los índices y aplicarlo al dataset dado.
* Los índices se presentan en un TextView como recomendaciones al usuario.

**Creación de ClusterActivity.kt:**

Este archivo controla la página de visualización de los clústeres.

Componentes:

* Un gráfico de dispersión que muestra los clústeres en base las dos primeras componentes principales del conjunto de datos al realizar PCA(análisis de componentes principales).
* Un TextView que indica las coordenadas de los centroides para cada clúster.

Generación del grafico:

* Se utiliza un layout personalizado que emplea líneas del entorno gráfico estándar de Android para representar los ejes y la cuadrícula. En este espacio se grafican los puntos generados por el algoritmo, representados mediante esferas estándar de la biblioteca gráfica de Android. Estas esferas son redimensionadas para adaptarse al layout principal y se posicionan de acuerdo con las coordenadas calculadas por el algoritmo. Los puntos se colorean según el clúster al que pertenecen.

Cálculo de Clustering:

* Se aplica el algoritmo KMeansPlusPlusClusterer de Apache Commons Math.
* Los datos procesados se organizan en objetos DoublePoint para realizar los cálculos de distancias y centroides.

**Creación de PreModeloPredictivoActivity.kt**

Este archivo gestiona la pantalla de selección de variables para el modelo LDA (Análisis Discriminante Lineal).

Componentes:

* Un RadioGroup dinámico que muestra las variables categóricas del dataset para seleccionar la variable dependiente (clasificatoria).
* Un botón de confirmación que, tras la selección, automáticamente elige como variables independientes todas las variables numéricas restantes.
* Un Toolbar con el logo de la app.

Lógica de selección y validación:

* Se construyen dos listas a partir de los datos: una para variables numéricas y otra para categóricas.
* El usuario selecciona una variable dependiente, y el sistema excluye esa variable del conjunto de independientes.

Si no se selecciona una variable, se muestra un Toast con un mensaje de advertencia.

Una vez validadas las selecciones, los datos se almacenan en DataParaRegresion y se inicia la actividad ModeloPredictivoActivity.

**Creación de ModeloPredictivoActivity.kt**

Este archivo controla la ejecución y visualización del modelo LDA, incluyendo el entrenamiento y la predicción.

Componentes:

* Una ProgressBar para indicar el proceso de carga y entrenamiento.
* Un LinearLayout que se llena dinámicamente con campos de entrada (TextInputDinamico) para ingresar valores a predecir.
* Una TableLayout que muestra las métricas del modelo entrenado.
* Un CardView donde se presentan los resultados después de entrenar.
* Un botón para ejecutar la predicción.
* Un TextView que muestra la categoría predicha para los valores ingresados.

Entrenamiento del modelo LDA:

* Se utiliza Chaquopy para ejecutar la función entrenamiento\_lda definida en un módulo Python (lda.py).
* Esta función recibe las variables predictoras, la variable dependiente y el dataset completo (matrizInicial).
* El modelo retorna un diccionario con métricas clave: Accuracy, Precision, Recall y F1-score, que se visualizan en una tabla.

Predicción:

* El usuario ingresa valores numéricos en los campos generados automáticamente.
* Al pulsar el botón, se llama a la función predecir\_categoria en Python, utilizando el modelo previamente entrenado y los valores ingresados.
* El resultado se muestra en un TextView indicando la categoría predicha.

**Creación de PreRegresionActivity.kt**

Este archivo gestiona la selección y visualización de las variables dependientes e independientes que se utilizaran en el modelo de regresión lineal múltiple.

Componentes:

* Un RadioGroup dinámico que muestra las variables numéricas del conjunto de datos para seleccionar la variable dependiente.
* Un layout con botones seleccionables que permite seleccionar varias variables independientes distintas de la dependiente.
* Un Toolbar con el logo de la app.
* Un botón que permite ir a la pagina del modelo de regresión lineal múltiple.

Lógica de selección y validación:

Se seleccionan únicamente las variables numéricas, estas se presentan en dos secciones, un apartado para seleccionar la variable dependiente, la cual solo permite escoger una y un apartado para seleccionar las variables independientes, la cual permite seleccionar distintas variables mientras que sean diferentes a la variable dependiente, en caso contrario presenta un Toast con una advertencia y no lo deja continuar.

**Creación de RegresionActivity.kt:**

Este archivo maneja la creación del modelo con las variables seleccionadas por el usuario, la visualización de las métricas del modelo y la predicción.

Componentes:

* Una ProgressBar para indicar el proceso de carga y entrenamiento.
* Un LinearLayout que se llena dinámicamente con campos de entrada (TextInputDinamico) para ingresar valores a predecir.
* Una TableLayout que muestra las métricas del modelo entrenado.
* Un CardView donde se presentan los resultados después de entrenar el modelo.
* Un botón para ejecutar la predicción.
* Un TextView que muestra la variable predicha para los valores ingresados.

Entrenamiento del modelo de regresión lineal múltiple:

* Se utiliza Chaquopy para generar el modelo en Python e ingresar los datos necesarios como la matriz de variables numéricas y un vector que contiene las entradas de la variable dependiente.
* Se realiza una muestra aleatoria simple del 70% para entrenar el modelo, el 30% restante se utiliza para probarlo y de esta manera retornar las métricas: , MAE, MSE, RMSE y la media de los residuos.

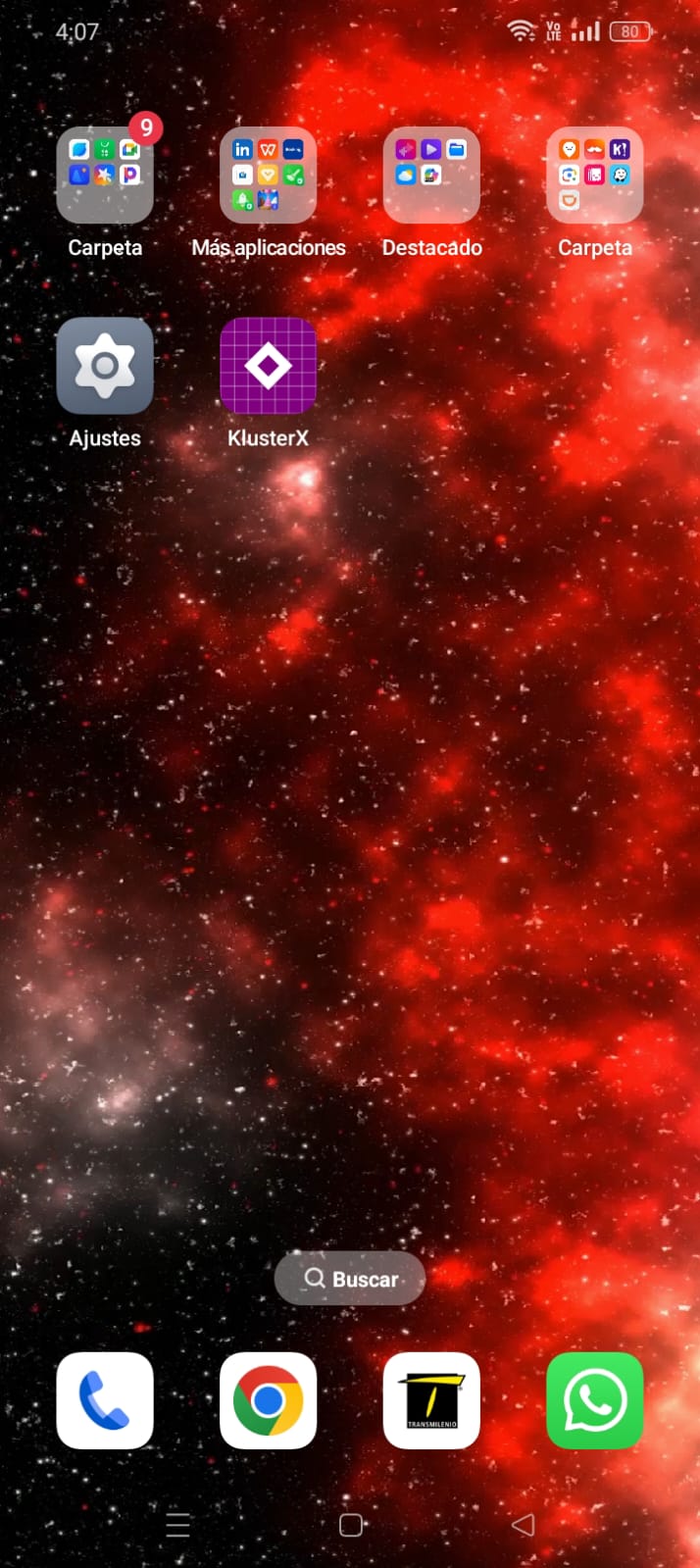
Predicción:

• El usuario ingresa valores numéricos en los campos generados automáticamente.

• Al pulsar el botón, se genera una predicción hecha en Python.

• El resultado se muestra en un TextView indicando la variable predicha y el resultado respectivo.

1. **Resultados**



La figura muestra la aplicación KlusterX instalada correctamente en un dispositivo Android, como se visualiza en la pantalla principal del teléfono. El ícono de la app aparece identificado por su logotipo púrpura con una figura geométrica blanca y el nombre KlusterX, evidenciando que el sistema es funcional y se integra al entorno operativo del usuario como cualquier otra aplicación nativa.

Este tipo de visualización valida la compatibilidad de la app con dispositivos reales y demuestra que puede ser distribuida, instalada y ejecutada fuera del entorno de desarrollo.

El desarrollo de la aplicación móvil de análisis estadístico produjo una herramienta funcional capaz de cargar archivos CSV con una cantidad significativamente mayor de filas y columnas que versiones anteriores (superando el límite de 200 filas y 5 columnas), calcular estadísticas descriptivas, sugerir automáticamente el número óptimo de clústeres usando el método del codo (elbow method), y ejecutar modelos de clustering (K-means) y predicción (Regresión Lineal Múltiple y LDA) con visualización de resultados y métricas relevantes.

Tabla

El contenido generado por IA puede ser incorrecto.

Una de las principales mejoras fue la transición del formato XLSX al formato CSV, lo cual amplió la capacidad de procesamiento del sistema, redujo el consumo de memoria y mejoró la compatibilidad multiplataforma. El módulo de carga valida que los datos sean consistentes (numéricos o categóricos según corresponda) y estén completos, permitiendo su uso en análisis posteriores.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

El módulo ResumenEstadisticasActivity permite al usuario explorar estadísticas básicas como media, desviación estándar, valor mínimo y máximo por columna, facilitando una comprensión inicial del dataset. Este análisis sirve como punto de partida para decidir qué variables incluir en los modelos predictivos o de agrupamiento.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Actualmente, se emplea únicamente el método del codo (elbow method), que calcula la suma de errores cuadráticos dentro de los clústeres (WCSS) para distintos valores de k. Esto simplificó el diseño y eliminó errores de implementación anteriores, como los experimentados con el índice de Calinski-Harabasz.

Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

funcionalidad de agrupamiento K-means, visualizada en ClusterActivity, permite identificar patrones ocultos en los datos. Se utiliza un gráfico de dispersión que muestra los puntos de datos coloreados por clúster y los centroides respectivos, tomando como ejes las dos primeras variables numéricas seleccionadas.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto, Teams

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Este módulo permite al usuario seleccionar una variable dependiente numérica y una o más variables independientes. Se calcula un modelo de regresión lineal múltiple que entrega coeficientes, una predicción basada en valores ingresados manualmente, y métricas de evaluación como R², MAE y RMSE. Los cálculos se realizan en segundo plano con Python mediante Chaquopy, aprovechando bibliotecas como NumPy y scikit-learn.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

El módulo LDA permite clasificar instancias en categorías basadas en un conjunto de variables independientes. El usuario selecciona una variable dependiente categórica y varias independientes. El sistema entrena un modelo LDA que entrega como salida:

Matriz de métricas: Accuracy, precisión, recall, F1-score.

Predicción de clase a partir de valores ingresados por el usuario.

Evaluación general del sistema:

El cambio hacia archivos CSV y el uso de Python para cálculos avanzados permitió superar muchas de las limitaciones anteriores asociadas al entorno Android y al uso exclusivo de Java. A pesar de que el procesamiento de modelos complejos como LDA o regresión aún supone desafíos de memoria y eficiencia en algunos dispositivos de gama baja, los resultados obtenidos muestran una herramienta robusta, adaptable y educativa, que guía al usuario desde la exploración de datos hasta la construcción e interpretación de modelos.

El proceso también reveló la importancia de diseñar interfaces intuitivas que permitan a usuarios sin formación técnica realizar análisis significativos. La integración entre el frontend (Java/Kotlin) y el backend (Python, Kotlin) fue un desafío técnico superado con éxito mediante el uso de Chaquopy, permitiendo una experiencia fluida sin abandonar el entorno Android.

1. **Conclusiones**

El desarrollo de la aplicación móvil KlusterX representó un ejercicio completo de integración entre teoría estadística y desarrollo tecnológico, enmarcado dentro de las restricciones de programación móvil. Inicialmente centrada en el clustering mediante el algoritmo K-means, la app evolucionó hasta incorporar funcionalidades adicionales como el análisis de estadísticas descriptivas, la predicción mediante Regresión Lineal Múltiple, y la clasificación mediante LDA (Análisis Discriminante Lineal), ampliando significativamente su alcance y valor práctico.

Uno de los principales logros fue el reemplazo del formato XLSX por archivos CSV, lo que permitió trabajar con conjuntos de datos de mayor tamaño y complejidad, superando las limitaciones anteriores de 200 filas y 5 columnas. Este cambio no solo mejoró el rendimiento, sino que también simplificó la experiencia del usuario y mejoró la compatibilidad con diversos dispositivos móviles.

Además, la implementación de los módulos predictivos permitió al usuario interactuar directamente con modelos estadísticos avanzados sin necesidad de conocimientos técnicos profundos. La posibilidad de ingresar valores manualmente para obtener predicciones y visualizar métricas como R² o precisión, hacen de la app una herramienta educativa y funcional para el análisis de datos en tiempo real.

Durante el proceso de desarrollo, se enfrentaron desafíos importantes relacionados con la limitación de recursos en dispositivos móviles, la ausencia de bibliotecas estadísticas nativas en Java/Kotlin, y la necesidad de garantizar una experiencia de usuario fluida. Estos obstáculos fueron superados mediante la integración con Python a través de Chaquopy, lo cual habilitó el uso de bibliotecas especializadas como NumPy y Scikit-learn.

Asimismo, se logró validar la ejecución exitosa de la app en diversos entornos Android, incluyendo capas personalizadas como MIUI, ColorOS y HarmonyOS, evidenciando su versatilidad multiplataforma.

Sin embargo, se identifican también áreas de mejora para futuras versiones:

Optimizar aún más la interfaz gráfica, haciendo uso de mejores prácticas de diseño UX/UI.

Expandir la compatibilidad con formatos de datos alternativos o conectores a bases de datos.

Agregar visualizaciones interactivas y dinámicas para facilitar la interpretación de modelos más complejos.

Permitir personalizar nombres de clústeres o etiquetas de clases predichas, mejorando la utilidad en contextos reales de análisis.

KlusterX demuestra que es posible desarrollar herramientas estadísticas funcionales, educativas y accesibles desde dispositivos móviles, siempre que exista una planificación técnica sólida y un enfoque claro en la experiencia del usuario. Este proyecto reafirma el potencial de las apps móviles para democratizar el acceso al análisis de datos y servir como puente entre la estadística aplicada y su uso cotidiano.

**Referencias**

[1] MacQueen, J. (1967). Some Methods for Classification and Analysis of Multivariate Observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*.

[2] Park, S., Lee, D., & Kim, H. (2017). Resource-efficient clustering algorithms for mobile systems. *Journal of Mobile Computing*.

[3] Tan, P. N., Steinbach, M., & Kumar, V. (2018). *Introduction to Data Mining*. Pearson Education.

[4] Android Developers (2024). Official Android Development Documentation..

[5] Steinbach, & Kumar (2018). Introduction to Data Mining. Pearson Education.

[6] Android Developers. (n.d.). Interdependencias de herramientas y bibliotecas. Android Developers. Retrieved from <https://developer.android.com/build/tool-and-library-dependencies?hl=es-419>.