INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENIERÍA EN COMPUTACIÓN

PRIMER PROYECTO:

LENGUAJES PROGRAMACIÓN

Profesor:
Oscar Víquez Acuña

Estudiantes
Isaí José Navarro Serrano
Marco Antonio Quirós Cabezas

Campus San Carlos

28 de abril de 2025

Índice

1. Introducción

En el contexto de la evaluación del rendimiento académico, este proyecto abordará el desafío de transformar datos dispersos sobre calificaciones y contenidos en información estructurada y útil para la toma de decisiones educativas. Se partirá de la premisa de que las notas agregadas de un estudiante en un curso no bastan para comprender en detalle su evolución; por ello, se desglosa cada asignatura en evaluaciones (exámenes, tareas, quices) y subtemas específicos, capturando tanto el valor porcentual de cada prueba como la calificación obtenida en cada área de conocimiento.

Para llevar a cabo este análisis multivariado, se adopta un enfoque híbrido: en OCaml se implementará las rutinas estadísticas que calculan promedios, desviaciones, correlaciones y tendencias de rendimiento, es decir se implementará lo relacionado a los cálculos específicamente; en Python se desarrolla la capa de manejo de datos y lectura de los mimos desde los archivos JSON—y la generación de gráficos e informes interactivos mediante Tkinter y Matplotlib. De esta manera, se aprovecha la potencia del paradigma funcional para la generación de métricas y la versatilidad de Python para la visualización y administración de la información.

2. Selección y diseño de variables para el manejo de información

Para el análisis del rendimiento académico, se diseñó una estructura de datos basada en archivos JSON que representan estudiantes, evaluaciones y conocimientos por subtemas por curso.

Ejemplo de estructura JSON

```
{
  "estudiantes": [
    {
      "id": "12345",
      "nombre": "Juan P\'erez",
      "cursos": [
        {
          "codigo": "MAT101",
          "nombre": "Matem\'aticas I",
          "evaluaciones": [
            {"id": "P1", "nota": 85, "porcentaje": 30},
            {"id": "P2", "nota": 90, "porcentaje": 70}
          ],
          "temas": [
            {"id": "T1", "nombre": "\'Algebra", "nota": 88},
            {"id": "T2", "nombre": "Geometr\'ia", "nota": 92}
          ]
        }
      ]
    }
  ]
}
```

En este modelo de datos, cada estudiante se representa como una entidad con un identificador único, un nombre completo y la carrera que cursa; esta combinación garantiza que se pueda segmentar el análisis por individuo y agrupar resultados por programa académico. Al mismo tiempo, cada curso viene definido por un código estandarizado, su nombre, el semestre y el año en que se imparte; de esta forma, se atribuye a las evaluaciones un contexto preciso, donde se sabe exactamente a qué asignatura y periodo corresponden.

Dentro de cada curso, se utilizará una colección de evaluaciones —parciales, finales, quices o tareas— en la que cada registro incluye un identificador interno, el tipo de actividad evaluativa, el valor porcentual que aporta a la nota final y, por supuesto, la calificación obtenida. Esto permite descomponer el desempeño del alumno no solo en términos generales, sino también según la relevancia de cada prueba. Para profundizar aún más, se define los subtemas o "temas" de cada curso; donde cada uno de estos lleva un identificador propio y una nota específica, de modo que podemos reconstruir un perfil detallado del dominio que el estudiante ha alcanzado en las distintas áreas de la asignatura.

A partir de estos datos primarios, se calculan métricas derivadas que facilitan el análisis estadístico: la nota final de cada curso, el promedio de todas las evaluaciones, la desviación estándar para medir la dispersión de resultados y la moda de las calificaciones para identificar valores recurrentes. También detectamos los subtemas críticos —aquellos con los promedios más bajos—, lo que nos ayuda a señalar posibles focos de dificultad, y evaluamos la tendencia histórica del rendimiento de cada alumno, categorizándolo en mejora, declive o estabilidad.

Este diseño de variables, estructurado en formato JSON, ofrece una gran flexibilidad: nuevas pruebas o métricas pueden incorporarse sin alterar la arquitectura fundamental del intercambio de datos entre OCaml y Python. Gracias a esta propuesta, somos capaces de realizar cálculos estadísticos avanzados en OCaml y, a continuación, desplegar gráficos e informes en Python que reflejan con precisión la evolución y los puntos críticos del desempeño académico de los estudiantes. Este diseño permite manejar tanto evaluaciones generales como análisis detallado por subtemas, facilitando el análisis multivariable de cada estudiante.

3. Justificación de estadísticas seleccionadas

Para entender bien cómo se comportan las notas de los estudiantes, se han seleccionado cinco estadísticas sencillas que, al combinarse, dan una visión completa de los datos:

Primero, el promedio nos muestra la nota media de un alumno o de todo un grupo en un curso. Esto permite comparar de forma rápida quién obtiene mejores o peores resultados. Sin embargo, el promedio no dice nada sobre la dispersión de las notas, por eso usamos la desviación estándar, que indica si las calificaciones están muy agrupadas alrededor del promedio o si hay alumnos con notas muy altas y muy bajas.

La moda es la calificación que más se repite. Con ella detectamos esos "picos" donde muchos estudiantes obtienen la misma nota, lo que ayuda a definir rangos claros para dar apoyo o reforzar contenidos. Para ver si dos variables están relacionadas —por ejemplo, si un mal desempeño en un subtema suele coincidir con una mala nota final— calculamos el coeficiente de correlación de Pearson, que mide la fuerza de esa relación lineal.

Por último, para analizar cómo evolucionan las calificaciones a lo largo del tiempo aplicamos una regresión lineal simple. Ajustamos una línea sobre la serie de notas de cada alumno y miramos su pendiente: si es positiva, el rendimiento mejora; si es negativa, empeora; y si es cercana a cero, se mantiene estable.

En conjunto, estas cinco métricas —promedio, desviación estándar, moda, correlación y regresión lineal— cubren desde la tendencia central y la dispersión hasta la frecuencia, la asociación entre variables y la evolución temporal. Gracias a ellas podemos identificar áreas críticas donde los estudiantes tienen dificultades y diseñar acciones de refuerzo académico bien fundamentadas.

4. Resultados obtenidos y reportes generados

A partir del diseño anteriormente planteado, se implementaron en OCaml las fórmulas que calculan promedio, desviación estándar, moda, correlación y regresión, basándose en métodos estadísticos estándar como mencionaremos a continuación:

4.1. Promedio aritmético

El promedio aritmético $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ se calcula como en [1]:

$$\bar{x} \;=\; \frac{1}{n}\sum_{i=1}^{n} x_i$$

donde x_i es cada calificación y n el número total de evaluaciones.

En nuestro proyecto, usamos esta fórmula para calcular la *nota final* de cada estudiante a partir de los pesos de cada prueba (parciales, finales, trabajos). En OCaml acumulamos la suma $\sum(x_i \times peso_i)$ y luego dividimos por la suma de pesos, redondeando a un decimal y limitando el resultado a 100 con las funciones `redondear_decimales` y `limitar_100`.

Finalmente, Python lee ese valor desde el JSON y lo muestra en los reportes como la barra que representa la nota promedio del alumno en cada gráfico. De este modo un concepto tan básico como el promedio se integra directamente en el flujo de cómputo y visualización.

4.2. Desviación estándar

La desviación estándar mide cuán dispersas están las calificaciones alrededor del promedio. Se define como la raíz cuadrada de la varianza muestral:

$$s \;=\; \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\big(x_i - \bar{x}\big)^2}$$

donde x_i es cada nota y $\bar{x}$ el promedio aritmético. En nuestro proyecto, OCaml calcula esta fórmula para cada curso y subtema; luego Python lee el valor desde el JSON y lo muestra junto al promedio para indicar si las notas están muy agrupadas (desviación baja) o presentan mucha variación (desviación alta).

Según Trejos Buriticá [1], la desviación estándar es fundamental para detectar desigualdades en los resultados y planificar acciones de refuerzo donde la variabilidad sea mayor.

4.3. Moda

La moda señala el valor o intervalo que aparece con mayor frecuencia en un conjunto de datos. En el caso de datos agrupados en clases de igual amplitud, Trejos Buriticá [1] propone la fórmula:

$$M_o = L_i + \frac{f_i - f_{i-1}}{(f_i - f_{i-1}) + (f_i - f_{i+1})}\, t_i$$

donde:

- L_i es el límite inferior de la clase modal (la que tiene mayor frecuencia).
- f_i es la frecuencia absoluta de esa clase.
- f_{i-1} y f_{i+1} son las frecuencias de las clases anterior y posterior.
- t_i es la amplitud del intervalo.

En nuestro proyecto, como las notas se manejan como valores discretos, simplificamos la definición: OCaml cuenta cuántas veces aparece cada nota final y elige la más frecuente como moda. Python luego muestra ese valor en los reportes para identificar concentraciones de desempeño.

4.4. Correlación

El coeficiente de correlación mide la fuerza de la relación lineal entre dos variables cuantitativas. Por ejemplo, nos permite saber si las notas en un subtema van de la mano con las notas finales. Se define como:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\,\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Según Delgado de la Torre [2], valores de r cercanos a 1 o -1 indican una relación lineal fuerte; valores cerca de 0 indican poca o ninguna relación.

4.5. Regresión

La regresión lineal simple ajusta una línea que mejor aproxima la relación entre dos variables, por ejemplo, el rendimiento a lo largo de varias evaluaciones. A partir de datos (x_i, y_i) se calculan:

$$\beta_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}, \qquad \beta_0 = \bar{y} - \beta_1\,\bar{x}$$

y la recta resultante es $y = \beta_0 + \beta_1\, x$. En OCaml usamos estas expresiones para trazar la tendencia de las notas, y Python presenta esa línea en los gráficos. Delgado de la Torre [2] explica cómo interpretar β_1: pendiente positiva para mejora, negativa para declive, y alrededor de cero para estabilidad.

Al correr todo el flujo integrado, obtenemos un archivo `resultados.json` que contiene:

- Una lista de objetos por cada estudiante con su nota final, notas por evaluación, promedio por subtema y tendencia.
- Un objeto de estadísticas generales del curso: promedio, moda, nota mínima y máxima, promedio por subtema y subtemas críticos.

A continuación describimos con detalle los reportes que Python genera usando estos datos y para qué sirven:

4.6. Informe por estudiante

Python muestra varias gráficas que ayudan a entender el rendimiento de cada alumno:

1. **Evolución general:** una línea que une las notas de todas las evaluaciones en orden cronológico. Sirve para ver si el estudiante mejora o empeora con el tiempo.
2. **Parciales:** un gráfico de barras con las notas de los exámenes parciales (P1, P2, P3...). Permite descubrir si un parcial específico resulta más difícil.
3. **Otras evaluaciones:** barras para trabajos, laboratorios u otros tipos de pruebas. Ayuda a comparar desempeño en diferentes tipos de actividades.
4. **Rendimiento por subtema:** barras con la nota promedio en cada subtema. Esto señala con precisión en qué parte del contenido el alumno tiene más o menos dominio.
5. **Tendencia:** después de la gráfica se muestra un mensaje que clasifica la tendencia como "mejora", .^{em}peora.^{o} .^{es}table", según la pendiente de los primeros tres parciales.

Uso de este informe: el profesor puede revisar cada caso y diseñar refuerzos a medida; el alumno puede ver sus puntos débiles y concentrarse en esos temas.

4.7. Informe global del curso

Para tener una visión general, Python genera:

1. **Comparativo de notas finales:** un gráfico de barras con las notas finales de todos los estudiantes. Facilita ver la distribución del rendimiento en el grupo.
2. **Estadísticas resumidas:** un cuadro con promedio, moda, nota mínima y máxima. Sirve para resumir rápidamente cómo le está yendo al curso en su conjunto.
3. **Subtemas críticos:** un listado de los tres subtemas con menor promedio global. Esto señala las áreas donde la mayoría de alumnos tiene más dificultades.

Uso de este informe: la coordinación académica puede detectar asignaturas o bloques de contenido problemáticos y ajustar el plan de estudio o los recursos de apoyo.

4.8. Reporte de subtemas

Finalmente, se genera un gráfico de barras ordenado de mayor a menor con el promedio de cada subtema en todo el curso. Esto:

- Muestra en un solo vistazo qué temas dominan los estudiantes y cuáles necesitan más trabajo.
- Permite priorizar intervenciones: se puede enfocar primero en los subtemas con notas más bajas.

Con estos tres reportes, el análisis multivariable cobra sentido práctico: no solo calculamos estadísticas, sino que traducimos los números en gráficos e información útil para mejorar el proceso de enseñanza y aprendizaje.

5. Conclusiones y elementos de mejora

En este proyecto logramos unir OCaml y Python para calcular y mostrar de forma fácil las estadísticas de rendimiento académico. Con OCaml procesamos las notas y obtuvimos el promedio, la desviación estándar, la moda, la correlación y la pendiente de la regresión. Luego, en Python leímos esos resultados en JSON y generamos gráficos claros con Tkinter y Matplotlib.

Gracias a estos cálculos, pudimos ver en qué subtemas los estudiantes tienen más dificultades y si su rendimiento tiende a mejorar, empeorar o mantenerse igual con el paso del tiempo. Esa información nos ayuda a tomar decisiones y a diseñar estrategias de apoyo.

Para seguir mejorando la aplicación, proponemos:

- Pasar la interfaz de Tkinter a una página web sencilla (por ejemplo, usando Flask) para que sea accesible desde cualquier navegador.
- Guardar los datos en una base de datos (en lugar de solo archivos JSON) para manejar más estudiantes y facilitar búsquedas y actualizaciones.
- Añadir más análisis, como agrupar estudiantes con patrones similares o medir cambios en intervalos cortos de tiempo.
- Organizar el código en módulos más pequeños y usar Docker para que instalar y ejecutar el proyecto sea más rápido y sencillo.

Con esto, es posible optimizar el sistema convirtiendolo en una herramienta más fácil de usar, más rápida y que a la vez puede ofrecer análisis más profundos para apoyar el aprendizaje de los estudiantes y facilitar las tareas de los docentes.

Referencias

[1] O. I. T. Buriticá, *Probabilidad y estadística para ingenieros*. Bogotá: Ecoe Ediciones, 2019, [En Línea]. [Online]. Available: https://elibro.net/es/ereader/itcr/126468?

[2] R. D. de la Torre, *Probabilidad y estadística para ciencias e ingenierías*. Las Rozas, Madrid: Delta Publicaciones, 2007, [En Línea]. [Online]. Available: https://elibro.net/es/ereader/itcr/170147?