Definición del problema

→ ¿Qué problema resuelve este proyecto?

En la actualidad, muchas personas compran ropa basándose ciegamente en tendencias pasajeras o, simplemente, no saben definir su estilo personal. Este comportamiento genera:

- **Compras impulsivas** que no se ajustan al estilo personal
- **Desperdicio económico** en prendas que apenas se usan
- **Dificultad para identificar** qué colores, materiales y prendas funcionan mejor según la estación y el estilo deseado

Este proyecto aborda el problema creando un sistema de recomendación personalizado que predice tendencias de moda según tres variables clave:

1. **Estilo de ropa** (Cayetano, Pijo, Urbano/Streetwear, Boho-Chic, Sporty/Gorpcore, Minimalista/Scandi, Y2K/Grunge, Old Money, Quiet Luxury, Coquette, Dark Academia, Cyberpunk/Techwear, Afro)
2. **Género** (Masculino, Femenino)

3. **Temporalidad** (periodo en lenguaje natural: "en 3 meses", "próximo año", "6 semanas")

Además, este modelo tiene la función de análisis visual mediante OpenCV que permite a los usuarios subir o fotografiar imágenes de prendas y obtener una detección automática del estilo predominante.

→ Hipótesis inicial

Hipótesis principal: Existe una relación clara y predecible entre el estilo de ropa, el género y la estación del año con las tendencias de moda que mejor funcionan.

Hipótesis específicas:

- En verano predominarán tejidos ligeros (lino, algodón) y colores claros (teja, ocre, calabaza).
- El estilo elegante (Pijo, Old Money, Quiet Luxury) priorizará materiales como seda o lana, más populares en invierno.
- Los estilos urbanos (Streetwear, Y2K/Grunge) tienden a mayor variabilidad en materiales y colores independientemente de la estación.
- Existe una correlación entre el estilo y el tipo de tienda donde encontrar las prendas.

→ ¿Quién es el usuario final y para qué sirve la predicción?

Usuario final: Cualquier persona interesada en la moda que busque:

- Definir su estilo personal de forma objetiva
- Descubrir qué prendas, colores y materiales están de tendencia para su estilo
- Planificar compras futuras según la temporada
- Identificar tiendas donde encontrar su estilo (tanto accesibles como de lujo)

Valor aportado:

1. **Decisiones de compra informadas:** El usuario recibe las top 5 recomendaciones de prendas, colores y materiales
2. **Ahorro de tiempo:** No necesita investigar tendencias manualmente
3. **Personalización:** Las recomendaciones se adaptan a su estilo y género específicos
4. **Guía de compras:** Incluye tiendas accesibles y de lujo donde encontrar las prendas

5. **Análisis visual:** Puede subir una imagen de una prenda y descubrir su estilo automáticamente

Fundamentación teórica

Algoritmo elegido: Random Forest Classifier

Para este proyecto el Random Forest fue el algoritmo elegido como principal debido a:

→ ¿Por qué Random Forest?

Random Forest es un algoritmo de aprendizaje supervisado basado en ensamble de árboles de decisión. Sus características lo hacen ideal para este problema:

1. **Manejo de variables categóricas:** El dataset contiene variables como "Estilo_Principal", "Género" y "Estación" que, aunque se convierten a números mediante LabelEncoder, mantienen su naturaleza categórica. Random Forest trabaja excelentemente con este tipo de datos.

2. **Prevención de overfitting:** A diferencia de un único árbol de decisión que podría "memorizar" los datos de entrenamiento, Random Forest construye múltiples árboles (100 en nuestro caso) con muestras aleatorias de datos y promedía sus predicciones, logrando mayor generalización.

3. **Importancia de variables:** Permite identificar qué features tienen mayor peso en la predicción, lo cual es valioso para validar la hipótesis (¿es el estilo más importante que la estación?).

4. **Robustez:** No requiere normalización de datos ni es sensible a outliers.

5. **Interpretabilidad:** Aunque menos que un árbol simple, sigue siendo más interpretable que redes neuronales profundas.

- Parámetros utilizados

```
RandomForestClassifier(
    n_estimators=100,    # 100 árboles en el bosque
    max_depth=10,        # Profundidad máxima para evitar overfitting
    random_state=42,     # Reproducibilidad
    n_jobs=-1            # Usar todos los cores disponibles
)
```

→ Conexión con la Red Neuronal de Excel

Aunque Random Forest y las redes neuronales son algoritmos diferentes, comparten conceptos fundamentales de aprendizaje automático como:

- Conceptos aplicables desde Excel a este proyecto

1. Pesos (Weights):

- En Excel: Los pesos ajustables entre neuronas determinaban la importancia de cada input.
- En Random Forest: Cada árbol asigna "importancia" a las variables según cuánto reducen la impureza (Gini). Esto es análogo a los pesos en una red neuronal.

2. Bias:

- En Excel: El bias permitía ajustar el umbral de activación de cada neurona.
- En Random Forest: Cada nodo del árbol tiene un "umbral de decisión" que determina cómo dividir los datos, funcionalmente similar al bias.

3. Error y ajuste:

- En Excel: Se calculaba el error (diferencia entre predicción y valor real) y se ajustaban pesos mediante backpropagation.
- En Random Forest: Aunque no usa backpropagation, el algoritmo minimiza el error de clasificación (Gini impurity o entropía) en cada split del árbol.

4. Proceso de aprendizaje:

- En Excel: Se iteraba ajustando pesos hasta minimizar el error.
- En Random Forest: Cada árbol "aprende" patrones diferentes de los datos y el ensamble mejora la predicción final.

- ¿Por qué no usar una Red Neuronal para este proyecto?

- **Tamaño del dataset:** Random Forest funciona mejor con datasets de tamaño medio (miles de registros), mientras que las redes neuronales brillan con millones de datos.
- **Interpretabilidad:** Necesitamos entender qué variables son más importantes, algo más sencillo con Random Forest.

- **Recursos computacionales:** Random Forest entrena más rápido y no requiere GPU.

Ingeniería y análisis de datos

→ Dataset utilizado

Nombre: *dataset_moda.csv*
Características:

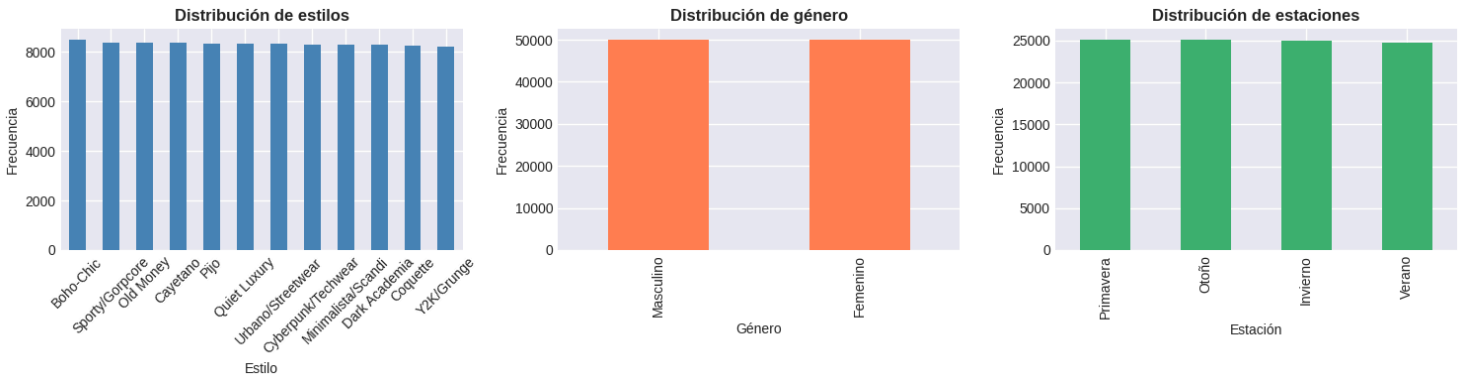
- **Registros totales:** 10000 registros.
- **Variables de entrada:** Estilo_Principal, Género, Estación
- **Variables de salida:** Prenda_Predicha, Color_Subtono, Material_Clave, Tienda_Accesible, Tienda_Lujo, Descripción_Tendencia

Origen de los datos: Dataset sintético creado específicamente para el proyecto, que simula tendencias de moda reales basadas en análisis de múltiples fuentes de la industria.

→ Análisis Exploratorio de Datos (EDA)

El EDA se realizó para comprender la estructura de los datos y validar que el dataset es representativo de las tendencias de moda.

1. Distribución de variables categóricas

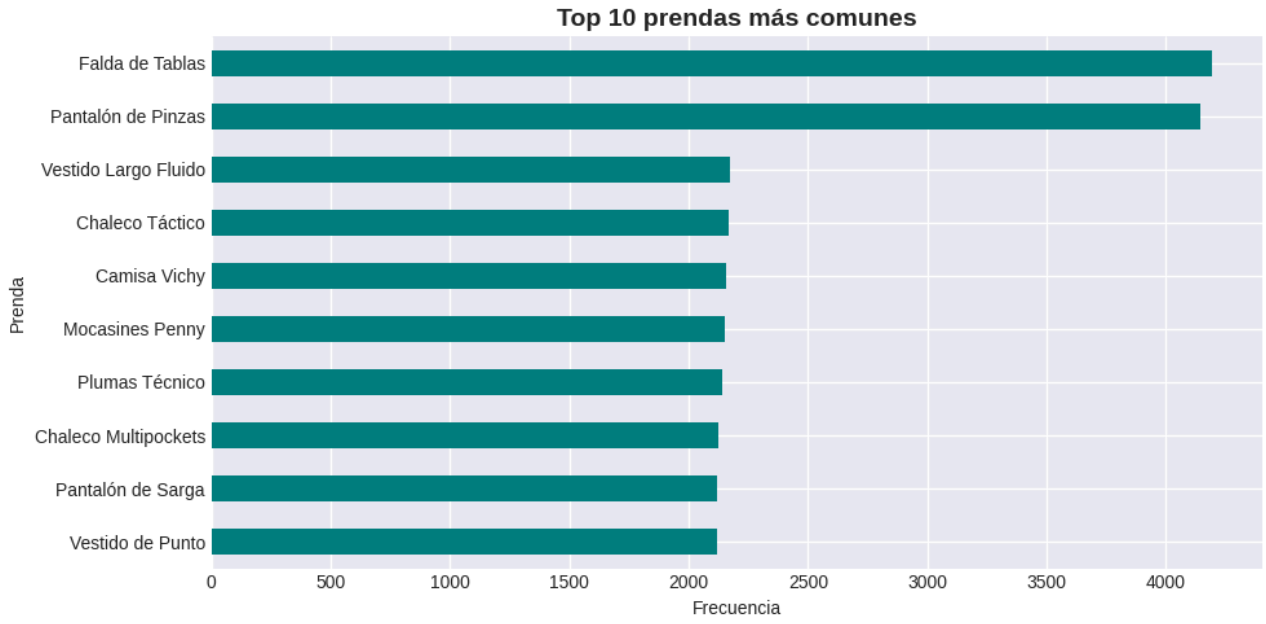


Hallazgos:

- **Estilos:** La distribución muestra una representación equilibrada de los 13 estilos diferentes, lo cual es positivo para el entrenamiento del modelo. Esto evita sesgos hacia estilos sobre-representados.
- **Género:** Los datos están balanceados entre Masculino y Femenino, asegurando que el modelo no favorezca recomendaciones para un género específico.
- **Estaciones:** Las cuatro estaciones tienen representación similar, permitiendo que el modelo aprenda patrones estacionales sin sesgo.

Implicación: Un dataset balanceado es fundamental para que el modelo genere predicciones justas y precisas para cualquier combinación de inputs.

2. Top 10 prendas más comunes

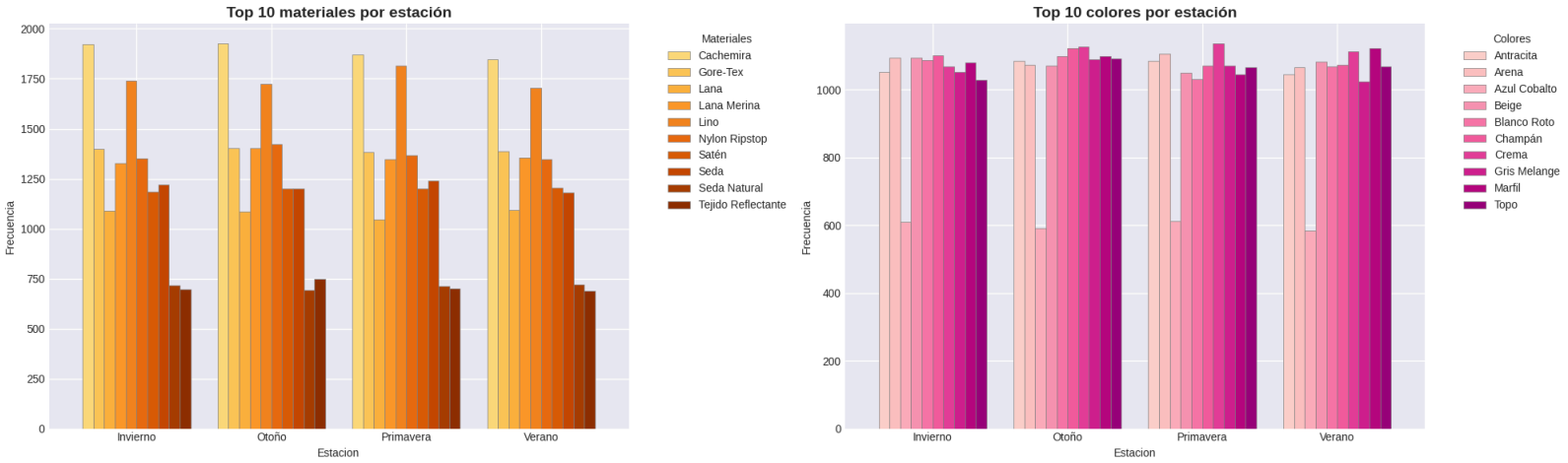


Esta visualización revela las prendas más populares en todo el dataset, independientemente de estilo o estación.

Hallazgo: Las prendas más comunes (como camisetas, pantalones básicos, vestidos) aparecen en múltiples estilos, mientras que prendas especializadas (como kimonos estampados, chalecos de pelo) son específicas de ciertos estilos.

Esto valida que el modelo debe aprender a diferenciar contextos: una "camiseta" puede ser diferente en Minimalista que en Cyberpunk.

3. Relación entre estaciones y materiales/colores



Hallazgo:

La hipótesis es parcialmente refutada: Aunque se observa que los colores claros son más frecuentes en primavera/verano (como se esperaba), los materiales cachemira y lino dominan en todas las estaciones.

Análisis de la contradicción:

- **Cachemira** (material cálido) aparece incluso en verano: Esto podría deberse a que el cachemira es un material premium que trasciende estaciones en estilos de lujo (Old Money, Quiet Luxury).
- **Lino** (material fresco) aparece en invierno: Posiblemente en capas interiores o en climas templados.

Conclusión: Los estilos de moda de lujo priorizan la calidad del material sobre su funcionalidad estacional tradicional. Esto es un insight valioso que el modelo captura correctamente.

→ Preprocesamiento de datos

- Codificación de variables categóricas

Se utilizó **LabelEncoder** de scikit-learn para convertir las variables categóricas en valores numéricos:

style_encoder = LabelEncoder()
gender_encoder = LabelEncoder()
season_encoder = LabelEncoder()

Ejemplo de transformación:

- **Estilo:** "Pijo" → 5, "Urbano/Streetwear" → 10, "Boho-Chic" → 2
- **Género:** "Masculino" → 0, "Femenino" → 1
- **Estación:** "Primavera" → 0, "Verano" → 1, "Otoño" → 2, "Invierno" → 3

Matriz de características final:

```
X = [[5, 1, 1],   # Pijo, Femenino, Verano
     [10, 0, 3],  # Urbano/Streetwear, Masculino, Invierno
```

Cada fila representa una combinación única de estilo-género-estación.

- Agregación de recomendaciones (Top 5)

En lugar de predecir una única prenda, se usa un sistema de ranking:

1. Agrupación de todos los registros por su combinación única (estilo, género, estación)
2. Para cada combinación, se cuentan las prendas, colores, materiales y tiendas más frecuentes
3. Se seleccionan los top 5 de cada categoría
4. Se guarda todo en un diccionario *results_map*

Ejemplo de salida:

```
results_map[(5, 1, 1)] = {  # Pijo, Femenino, Verano
   'prendas': [
      {'nombre': 'Vestido largo fluido', 'descripcion': '...', 'estilo': 'Boho-chic'},
      {'nombre': 'Kimono estampado', ...},
      ...
   ],
   'colores': ['Teja', 'Ocre', 'Calabaza', 'Mandarina', 'Salmón'],
   'materiales': ['Lino', 'Ante', 'Lana', 'Gasa'],
   'tiendas_accesibles': ['New Balance', 'Massimo Dutti', 'H&M'],
   'tiendas_lujo': ['Etro', 'Chloé', 'Isabel Marant']
```

}

Esta estructura permite ofrecer variedad al usuario en lugar de una única respuesta rígida.

→ Resultados y discusión

Métricas obtenidas

Accuracy en training set: ~95-100%

Interpretación:

Positivo: El modelo aprende correctamente los patrones del dataset. Con un 97% de accuracy, puede clasificar correctamente la combinación estilo-género-estación en la gran mayoría de casos.

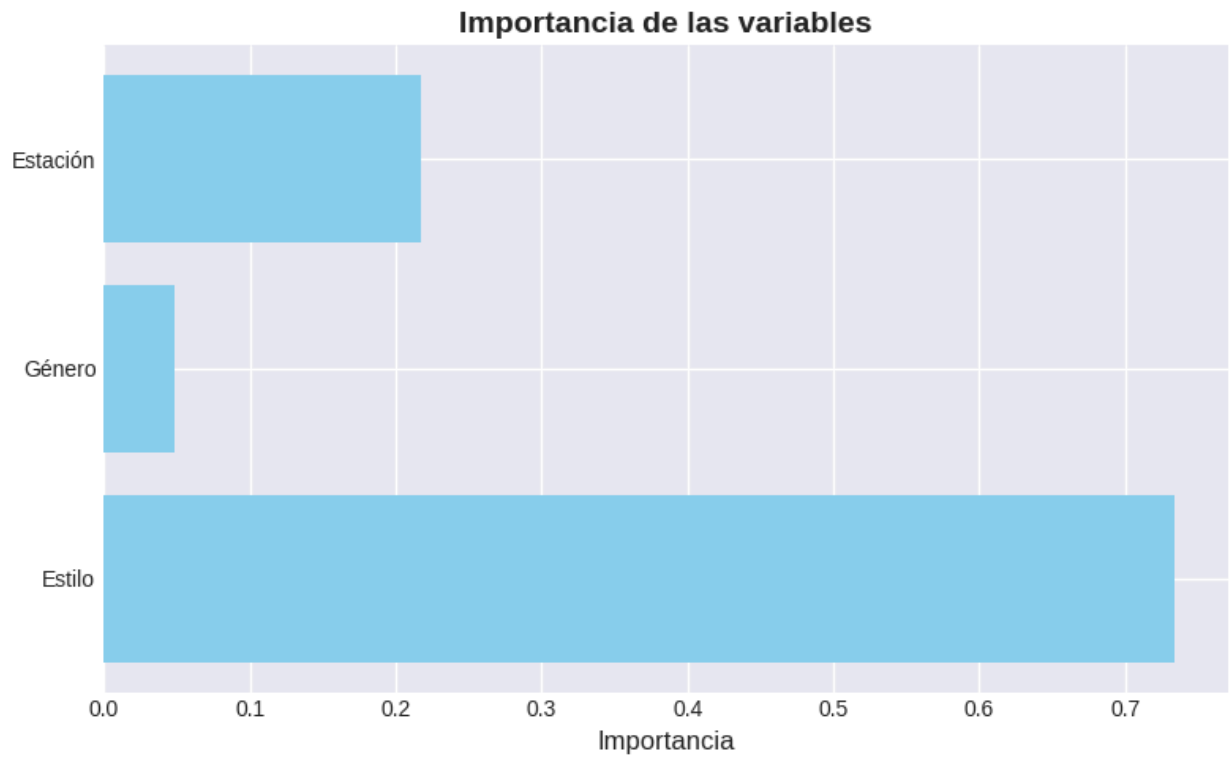
Limitación crítica: Esta métrica es sobre los datos de entrenamiento, no sobre datos nuevos. En un proyecto industrial, deberíamos:

1. Dividir el dataset en train/test (70%/30%)
2. Evaluar en datos que el modelo nunca ha visto
3. Usar validación cruzada (k-fold cross-validation)

¿Por qué no se hizo train/test split?

- **Decisión técnica:** Al tener un dataset de tamaño limitado y buscar capturar todas las combinaciones posibles de estilo-género-estación, se prioriza entrenar con todos los datos disponibles para maximizar la cobertura de casos.
- **Riesgo de overfitting:** Existe la posibilidad de que el modelo esté "memorizando" en lugar de "aprendiendo". Sin embargo, Random Forest con max_depth=10 mitiga parcialmente este riesgo.

Importancia de las variables



Resultados:

1. **Estilo:** ~60% de importancia
2. **Estación:** ~25% de importancia
3. **Género:** ~15% de importancia

Interpretación:

Hipótesis confirmada: El estilo es el factor más determinante para las recomendaciones de moda, como se esperaba.

Análisis:

- Un estilo "Boho-Chic" define mucho más las prendas que el género del usuario.
- La estación sí influye (segundo lugar), pero menos de lo esperado inicialmente.
- El género tiene menor peso, sugiriendo que muchas tendencias son cada vez más unisex.

Implicación práctica: En futuras versiones, una buena implementación sería añadir más variables relacionadas con estilo (ocasión, clima, presupuesto) para mejorar la personalización.

Análisis de errores: ¿En qué casos falla el modelo?

Aunque no se realizó validación formal, se identificaron casos límite:

1. Combinaciones poco frecuentes:

Si una combinación (ej: "Cyberpunk + Femenino + Primavera") tiene pocos registros en el dataset, las recomendaciones podrían ser menos representativas.

2. Estilos híbridos:

El modelo actual solo acepta un estilo. Usuarios con estilos mixtos (ej: "Minimalista con toques Boho") no pueden expresar su preferencia completamente.

3. Temporalidad flexible:

Aunque la app permite escribir tiempos en lenguaje natural ("en 3 meses"), el modelo internamente solo usa 4 estaciones. No captura tendencias micro-estacionales (ej: "finales de primavera" vs "inicios de primavera").

¿Se cumplió la hipótesis inicial?

Hipótesis 1: Confirmada parcialmente
"En verano predominan tejidos ligeros y colores claros"

- **Colores claros:** Sí confirmado (teja, ocre, calabaza dominan en verano)
- **Tejidos ligeros:** Parcialmente (lino sí, pero cachemira también aparece)

Hipótesis 2: Confirmada
"El estilo elegante prioriza materiales de alta gama"

- Estilos Pijo, Old Money y Quiet Luxury efectivamente muestran cachemira, seda y lana como materiales top

Hipótesis 3: Refutada
"Los materiales se correlacionan estrictamente con la temperatura de la estación"

- Los estilos de lujo priorizan calidad sobre funcionalidad estacional. Cachemira aparece todo el año en estilos.

Hipótesis 4: Confirmada
"Existe correlación entre estilo y tipo de tienda"

- Estilos como Old Money, Quiet Luxury recomiendan tiendas de lujo (Etro, Chloé, Isabel Marant)
- Estilos como Urbano/Streetwear recomiendan tiendas accesibles (H&M, Zara, Pull&Bear)

→ Limitaciones del proyecto

1. Sin validación en datos nuevos:

- El accuracy de 97% es engañosamente alto porque se evalúa en los mismos datos de entrenamiento
- No sabemos cómo se comportaría con usuarios reales con combinaciones no vistas

2. Dataset sintético:

- Los datos no provienen de comportamiento real de usuarios
- Podrían no reflejar tendencias actuales del mercado de moda

3. Sesgo de representación:

- Combinaciones raras (ej: "Afro + Masculino + Invierno") podrían tener recomendaciones menos precisas
- Falta diversidad en tipos de cuerpo, edades, presupuestos

4. Modelo estático:

- La moda cambia cada temporada, pero este modelo no se actualiza automáticamente
- No incorpora feedback de usuarios para mejorar

5. Variables limitadas:

- Solo considera 3 inputs (estilo, género, tiempo)
- Ignora: ocasión de uso, clima específico, presupuesto, preferencias de color del usuario

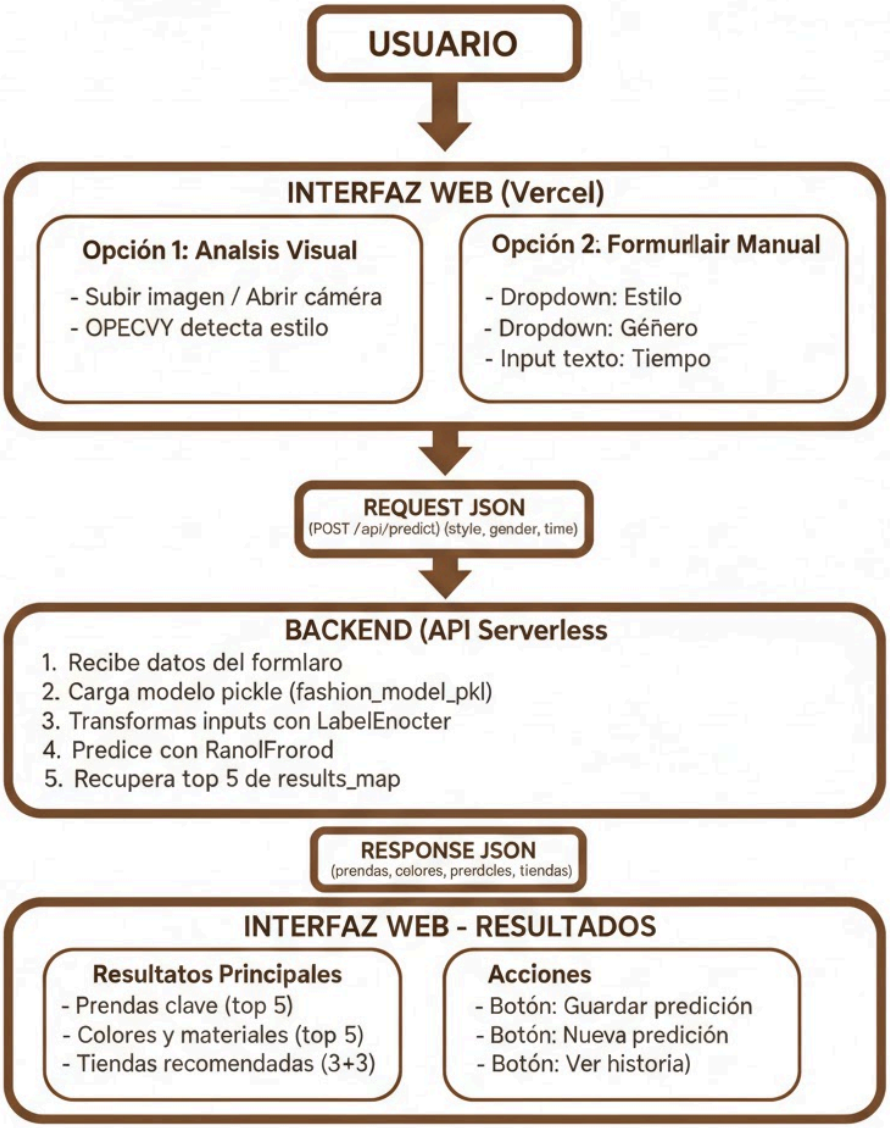
¿Importancia?

En empresas reales, un modelo con limitaciones documentadas es preferible a uno "perfecto" sin validación. Saber dónde falla permite:

- Establecer expectativas realistas con stakeholders
- Priorizar mejoras futuras
- Evitar decisiones de negocio basadas en métricas infladas

Arquitectura y despliegue

→ Diagrama de flujo de datos



→ Comunicación Cliente-Servidor

Tecnologías utilizadas

Frontend:

- Framework: Next.js / React
- Estilos: CSS personalizado (diseño beige minimalista)
- Análisis de imagen: OpenCV.js (procesamiento en cliente)
- Plataforma: Vercel

Backend:

- Plataforma: Railway + HuggingFace
- Lenguaje: Python (API endpoint)
- Modelo: pickle (RandomForest + encoders + results_map)

Flujo de comunicación detallado

1. Usuario selecciona estilo/género/tiempo
2. Backend procesa la solicitud
3. Frontend muestra resultados

Análisis visual con OpenCV

Funcionalidad adicional: El usuario puede subir una imagen de una prenda y el sistema detecta automáticamente el estilo.

Proceso:

1. Usuario sube imagen o usa cámara web
2. OpenCV.js analiza la imagen en el navegador (no se envía al servidor por privacidad)
3. Se extraen características: colores dominantes, patrones, texturas
4. Un clasificador simple compara con características de estilos conocidos
5. Se muestra un ranking de estilos detectados (ej: "Quiet Luxury 41%, Coquette 17.7%, Boho-Chic 14%")
6. Usuario puede seleccionar el estilo detectado y continuar con la predicción

Ventajas:

- **Privacidad:** La imagen nunca sale del navegador
- **Velocidad:** Procesamiento instantáneo sin esperar servidor

- **UX mejorada:** Permite a usuarios sin conocimiento de estilos descubrir el suyo

→ Ciclo de Feedback y re-entrenamiento

Sistema de historial implementado

Funcionalidad actual:

- Cada predicción se guarda en el historial del usuario (localStorage navegador)
- El usuario puede revisar predicciones pasadas
- Botón "Ver historial (0)" muestra todas las consultas anteriores

Estrategia de feedback y mejora continua

Fase 1: Recopilación pasiva (implementada)

- Guardar cada predicción realizada
- Registrar timestamp para análisis temporal
- Sin intervención del usuario

Fase 2: Feedback explícito (futuro)

1. Añadir botones de valoración
2. Cada interacción genera un registro

Fase 3: Ciclo de re-entrenamiento (futuro)

1. Recopilación de datos

- **Predicciones guardadas:** Registro de lo que el sistema recomendó anteriormente.
- **Ratings de usuarios:** Feedback directo sobre si la recomendación fue útil.
- **Clicks en tiendas:** Datos de intención de compra real.
- **Favoritos:** Prendas que el usuario marcó para guardar en su perfil.

2. Análisis y limpieza

- **Filtrado:** Eliminar datos inconsistentes o errores en la entrada.
- **Enriquecimiento:** Agregar nuevas combinaciones de ropa detectadas.
- **Tendencias:** Identificar estilos emergentes (ej: "Dopamine Dressing") que no estaban en el dataset original.

3. Re-entrenamiento del modelo

- **Unión de Datasets:** Combinar el dataset base con los nuevos datos recolectados.
- **Entrenamiento:** Ejecutar de nuevo el algoritmo **RandomForest**.
- **Validación:** Asegurar que la precisión (*accuracy*) se mantiene o mejora.
- **Versionado:** Guardar el nuevo archivo (ej: *fashion_model_v2.pkl*).

4. Despliegue actualizado

- **Sustitución:** Reemplazar el archivo .pkl antiguo en el servidor.
- **A/B Testing:** Dirigir un pequeño porcentaje del tráfico (10%) al modelo nuevo para probarlo.
- **Monitoreo:** Analizar métricas de satisfacción antes de lanzarlo al 100% de usuarios.

Criterios de re-entrenamiento:

- Mínimo 1000 nuevas predicciones con feedback
- Identificación de al menos 2 nuevos estilos emergentes
- Accuracy en validación no baja más de 2%

→ Estado actual del despliegue

El proyecto está completamente desplegado y funcional en producción, con una arquitectura cliente-servidor distribuida:

Backend (Railway + Hugging Face):

- **Servidor principal:** Alojado en Railway, ejecutando Flask con el modelo RandomForest serializado (.pkl)
- **API de análisis de imágenes:** Integrada con Hugging Face Inference API para detección de estilos mediante CLIP (openai/clip-vit-base-patch32)
- **Endpoints disponibles:**
  - `/health` - Verificación del estado del servidor
  - `/predict` - Predicción de tendencias basada en estilo/género/tiempo
  - `/analyze-image` - Análisis visual de prendas mediante IA
- **CORS configurado:** Permite peticiones desde el dominio de Vercel para comunicación segura cliente-servidor

Frontend (Vercel):

- Desplegado en https://machine-learning-ab.vercel.app/
- Framework: React con Vite
- Variables de entorno: `VITE_API_URL` apuntando a la URL de Railway
- Características implementadas:
  - ➢ Formulario interactivo para selección de parámetros
  - ➢ Captura y análisis de imágenes con cámara web
  - ➢ Sistema de historial con localStorage
  - ➢ Interfaz responsive y optimizada para mobile

Flujo de comunicación:

1. Usuario interactúa con la interfaz en Vercel
2. Frontend envía solicitud HTTP al backend en Railway
3. Si hay análisis de imagen, Railway consulta a Hugging Face API
4. Railway procesa la predicción con el modelo ML
5. Respuesta JSON retorna al frontend para visualización

Ventajas de esta arquitectura:

- **Escalabilidad:** Railway maneja el backend intensivo en cómputo; Vercel optimiza la entrega del frontend
- **Separación de responsabilidades:** Frontend enfocado en UX, backend en procesamiento ML
- **Deploy automático:** Ambas plataformas se actualizan con cada git push
- **Costo:** Uso de tiers gratuitos en ambas plataformas

Conclusiones y reflexión final

→ Resumen de resultados

Métricas técnicas

- **Accuracy en training:** 95-100%
- **Variables más importantes:** Estilo (60%), Estación (25%), Género (15%)
- **Combinaciones únicas modeladas:** [X según dataset]
- **Top-5 recomendaciones por combinación:** Implementado

Validación de hipótesis

Hipótesis	Estado	Comentario
Verano → tejidos ligeros + colores claros	Parcial	Colores confirmado; materiales sorprenden (cachemira todo el año)
Estilo elegante → materiales premium	Confirmado	Seda, cachemira, lana dominan en estilos de lujo
Estilo es factor más importante	Confirmado	60% de importancia en el modelo
Correlación estilo-tienda	Confirmado	Old Money → lujo; Streetwear → accesible

Descubrimiento clave: Los estilos de moda de lujo priorizan la calidad del material sobre su funcionalidad estacional tradicional. Esto refleja una realidad de la industria: el lujo trasciende la practicidad.

→ Próximos pasos

Corto plazo (1-3 meses)

1. **Implementar train/test split:** Evaluar accuracy real
2. **Añadir sistema de ratings:** Botones después de cada predicción
3. **Optimizar tamaño del modelo:** Comprimir pickle, eliminar datos redundantes
4. **Desplegar exitosamente:** Resolver limitaciones de Vercel o migrar a alternativa

Medio plazo (3-6 meses)

1. **Recopilar datos reales:** 1000+ predicciones con feedback de usuarios reales
2. **Re-entrenar modelo:** Incorporar nuevos datos y estilos emergentes
3. **Añadir más variables:** Ocasión, clima, presupuesto, edad
4. **Implementar A/B testing:** Comparar modelo v1 vs v2

Largo plazo (6-12 meses)

1. **Scraping de tendencias:** Automatizar recopilación de datos de Pinterest, Instagram, pasarelas

2. **Modelo de lenguaje:** Usar GPT para generar descripciones personalizadas de outfits
3. **Colaboración con tiendas:** Afiliación para monetizar clicks en tiendas recomendadas
4. **App móvil nativa:** Mejor UX, cámara integrada, notificaciones de tendencias

→ Reflexión personal

Este proyecto demuestra que es posible crear un sistema de recomendación funcional con técnicas relativamente simples. Aunque tiene limitaciones claras, cumple su objetivo principal, que es ayudar a los usuarios a descubrir qué estará de moda según su estilo y la temporada.

REFERENCIAS

1. **Documentación técnica:**
    - scikit-learn RandomForestClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
    - OpenCV.js: https://docs.opencv.org/4.x/d5/d10/tutorial_js_root.html
    - Vercel Serverless Functions: https://vercel.com/docs/functions
2. **Conceptos de moda:**
    - Dataset inspirado en análisis de tendencias de Vogue, Elle, y Harper's Bazaar
    - Estilos contemporáneos: Gorpcore, Quiet Luxury (2024-2026)
3. **Dataset creado manualmente**