Contexto.

Desarrollar un asistente virtual que facilite la organización de tareas personales y de esta manera el usuario pueda obtener mejor gestión de su tiempo.

Clasificación.

El objetivo fue crear un modelo que clasifique automáticamente las tareas según su nivel de urgencia e importancia, utilizando un enfoque de árboles de decisión.

¿Por qué arboles de decisión?

Los árboles de decisión nos funcionan para este tipo de clasificaciones porque se adaptan bien a escenarios en los que las decisiones están basadas en reglas claras. Para el asistente virtual, se necesitaba una herramienta que tomara en cuenta varios factores de manera jerárquica y asignara una prioridad en función de estos.

Características del modelo:

- Urgencia: define que tan pronto es necesario completar la tarea.
- Importancia: representa el impacto o valor a la tarea.
- Prioridad externa: es una medida de qué tanto la tarea afecta o depende de otros factores o personas externas al usuario.
- Días restantes: calculado a partir de la fecha límite, este valor ayuda al modelo a determinar cuánta anticipación necesita la tarea.

Preparación y Análisis de Datos

Descripción del Dataset

El conjunto de datos que utilizamos para este proyecto incluye diversas características de cada tarea, diseñadas para capturar la esencia de su relevancia y complejidad. Las columnas principales incluyen:

- urgencia: una escala que indica cuán pronto necesita completarse la tarea,
- importancia: una medida de la relevancia o el impacto de la tarea,
- prioridad_externa: que representa la prioridad asignada desde un contexto externo,
- días_restantes: el tiempo que queda hasta la fecha límite. Estas características, junto con otros factores como impacto en otros y bloqueo, ofrecen un panorama de cada tarea. Este dataset permite que el modelo reconozca patrones y asigne una prioridad de manera automatizada, considerando múltiples variables de cada tarea.

Preprocesamiento de Datos

Antes de poder entrenar un modelo confiable, fue esencial realizar un preprocesamiento exhaustivo de los datos. Esto incluyó varias etapas, como la limpieza de valores inconsistentes, el mapeo de valores categóricos, y la creación de la característica días_restantes a partir de la fecha límite. Adicionalmente, normalizamos ciertos valores para asegurarnos de que todos los datos estuvieran en un formato estándar y eliminar sesgos o anomalías que pudieran distorsionar los resultados. Estas preparaciones son fundamentales porque un modelo es tan bueno como los datos que recibe, y este proceso asegura que el modelo reciba un dataset limpio y equilibrado.

Desarrollo del Modelo de Clasificación

Elección del Algoritmo - Árbol de Decisión

Para abordar el desafío de clasificar y priorizar las tareas, optamos por un modelo basado en árboles de decisión. Este algoritmo es particularmente adecuado para nuestro proyecto por su transparencia y facilidad de interpretación. Un árbol de decisión clasifica datos basándose en una serie de decisiones binarias, lo que permite visualizar cómo se determinan las prioridades según las características de la tarea. Esta capacidad de 'explicabilidad' del modelo es crucial en aplicaciones donde queremos entender por qué una tarea fue priorizada. Los árboles de decisión también tienen la ventaja de manejar variables categóricas y continuas de manera eficiente y son altamente adaptables a modificaciones en las características de los datos.

Matemáticamente, el árbol de decisión opera de la siguiente manera:

Entropía e Impureza:

Para decidir cómo dividir los datos en cada nodo, el árbol utiliza medidas de "impureza" para evaluar la calidad de las divisiones. Dos medidas comunes son la entropía y el índice de Gini. Estas métricas miden el grado de mezcla de clases en cada nodo. Cuanto más homogéneo es el nodo en términos de clase, menor es su impureza.

Optimización de Hiperparámetros

Para maximizar el rendimiento del modelo, realizamos un proceso de optimización de hiperparámetros. Utilizamos la técnica RandomizedSearchCV, que permite ajustar automáticamente parámetros como n_estimators, max_depth, min_samples_split, y min_samples_leaf. Esta técnica explora combinaciones de valores de forma aleatoria dentro de un rango especificado y selecciona la mejor configuración basada en métricas de rendimiento. La optimización de estos parámetros permite reducir el sobreajuste, aumentar la precisión y equilibrar la complejidad del modelo.

Balanceo de Clases

Otro aspecto crucial fue el balanceo de clases. Cuando los datos están desequilibrados – por ejemplo, con muchas tareas de baja prioridad y pocas de alta prioridad – el modelo puede sesgarse hacia la clase mayoritaria. Para mitigar esto, empleamos RandomOverSampler, una técnica que genera ejemplos sintéticos de la clase minoritaria hasta equilibrar las clases. Esto asegura que el modelo pueda reconocer y clasificar tanto las tareas de alta como baja prioridad de manera efectiva, y mejora su precisión y capacidad para generalizar en nuevos datos.

Evaluación del Modelo

Métricas de Evaluación

Para medir el rendimiento del modelo de manera objetiva, utilizamos varias métricas de evaluación. La precisión, recall y F1-score nos permiten comprender no solo la cantidad de tareas clasificadas correctamente, sino también la efectividad del modelo en detectar tareas críticas y minimizar errores en clasificaciones prioritarias. Estas métricas son fundamentales en un contexto donde un error en la clasificación puede tener consecuencias significativas, por ejemplo, al subestimar la urgencia de una tarea.

Resultados de Validación

Para garantizar la consistencia y la robustez del modelo, implementamos validación cruzada de cinco pliegues. La validación cruzada divide los datos en cinco subconjuntos y entrena el modelo en cuatro de ellos mientras prueba en el quinto, repitiendo este proceso en cada combinación posible. Esto nos permite evaluar el rendimiento del modelo en distintas particiones del dataset, asegurando que los resultados no dependan de un solo conjunto de datos de prueba. Los resultados mostraron un rendimiento estable, lo que indica que el modelo es fiable y puede mantener su rendimiento con datos nuevos.

Integración del Modelo en Flask

Implementación de Rutas API

Para hacer que el modelo de clasificación sea accesible y utilizable de manera práctica, lo integramos en una API RESTful utilizando Flask. Las principales rutas incluyen /tasks para crear y clasificar tareas, /tasks/<id> para realizar actualizaciones o eliminaciones, y una ruta de retroalimentación para capturar ajustes sugeridos por los usuarios. Esta estructura modular de rutas permite que otros sistemas o aplicaciones accedan al modelo, facilitando su uso y despliegue en un entorno de producción.

Pruebas de Clasificación con Postman

Para verificar el funcionamiento de la API y la precisión de las clasificaciones, probamos las rutas con Postman. Este enfoque de prueba nos permitió realizar solicitudes POST y GET, simulando diferentes clasificaciones de tareas y verificando la respuesta del modelo. Además, configuramos parámetros como la urgencia, importancia y prioridad externa para observar cómo el modelo reaccionaba ante diferentes combinaciones de factores. Estas pruebas nos brindaron la seguridad de que el modelo estaba listo para responder correctamente a diversas entradas.

Demostración y Conclusión

Demostración Práctica

Para concluir, mostraremos una demostración práctica de la aplicación. Comenzaremos agregando una nueva tarea desde la API o la interfaz y observaremos cómo el modelo clasifica su prioridad. Luego, veremos la lista de tareas clasificadas en orden de prioridad, mostrando cómo el modelo ha interpretado cada una y asignado su clasificación. Esta demostración refuerza la usabilidad de la herramienta y la efectividad de nuestro modelo en la organización de tareas en función de su relevancia.

Conclusiones y Resultados Clave

En resumen, nuestro proyecto ha demostrado que es posible automatizar la priorización de tareas con un modelo de clasificación bien entrenado y ajustado. El modelo alcanzó altos niveles de precisión, y la API integrada permite su implementación en entornos prácticos, mejorando la productividad y la gestión del tiempo en escenarios reales. A través de este proyecto, hemos desarrollado una herramienta útil y escalable para optimizar la organización de tareas diarias, facilitando una mejor toma de decisiones y priorización en cualquier contexto de trabajo.