Areas de oportunidad Data Science

Feature Importance Analysis for Facebook Prophet:

In the context of time series forecasting with Facebook Prophet:

Feature Importance Analysis for Facebook Prophet:

1. Additive Components:

Facebook Prophet decomposes time series data into its additive components: trend, seasonality, and holidays. Understanding the contribution of each component to the overall prediction can provide insights into feature importance.

2. Holidays and Special Events:

Prophet allows you to include holidays and special events as additional features. Analyzing the impact of these events on the predictions can help you understand their significance.

3. Component Plots:

Prophet provides component plots that visualize the individual contributions of trend, seasonality, and holidays. These plots can give you an idea of the relative importance of different components.

4. Change Points and Trend Flexibility:

Prophet automatically detects changing points in the data where the trend's trajectory shifts. You can assess the impact of each change point on the model's predictions.

5. Holiday Effects:

When adding holidays as features, Prophet calculates how each holiday affects the predictions. This allows you to identify holidays with significant impact on the time series.

6. Changepoint Prior Scale:

The `changepoint_prior_scale` hyperparameter in Prophet controls the flexibility of the trend. By experimenting with different values, you can understand how changes in the trend's flexibility affect the model's performance.

7. Effect of Seasonalities:

Prophet supports multiple types of seasonality (e.g., daily, weekly). Analyzing the influence of different seasonal patterns on predictions can be valuable.

8. Regressors and Additional Covariates:

While Prophet is designed to handle time-based features, you can experiment with adding additional regressors and covariates to assess their impact on predictions.

9. Model Performance with and without Components:

Compare model performance with and without certain components (e.g., trend, holidays) to gauge their importance in driving accurate forecasts.

10. Business Insights:

Translate the insights from the feature importance analysis into actionable business decisions. For example, if certain holidays consistently have a strong impact on sales, you can allocate resources accordingly.

Remember that while Prophet provides insights into the impact of various components on time series predictions, it might not directly output feature importance scores as some other models do. Nevertheless, by leveraging the available components, plots, and configuration options, you can gain a deep understanding of how different factors contribute to the model's forecasts.

healthchecks de los usecases

Descripción:

Esta tarea implica la implementación de un sistema de healthchecks o controles de salud para los dos Use Cases principales del proyecto: **Promotion Analytics** y **Price Elasticities**. Los healthchecks son procesos automatizados que se ejecutan periódicamente para evaluar si estos casos de uso están funcionando como se espera y si los datos y resultados son coherentes y precisos. Los

healthchecks pueden incluir comprobaciones de la integridad de los datos de entrada, el rendimiento de los modelos y la exactitud de los resultados.

El propósito de esta tarea es garantizar la confiabilidad y la calidad continua de los resultados producidos por los Use Cases. Al implementar healthchecks efectivos, se pueden identificar problemas potenciales de manera proactiva y tomar medidas correctivas en caso de que se detecten problemas. Esto es esencial para mantener la precisión y la utilidad de los análisis de Promotion Analytics y Price Elasticities a lo largo del tiempo.

En resumen, esta tarea se enfoca en establecer mecanismos automatizados para monitorear y garantizar el buen funcionamiento de los Use Cases, lo que contribuirá a mantener la integridad de los resultados y a tomar decisiones informadas basadas en datos de alta calidad.

Complejidad técnica (del 1 al 5) : 2

Se necesita definir un protocolo en conjunto con negocio de como medir y contrastar.

Técnicamente se necesita generar un job con esta lógica dentro de databricks, o en su defecto un dag en airflow.

Prioridad técnica (del 1 al 5) : 1

Es importante, mas no es urgente, solo ayudara a mantener visibilidad a lo largo del tiempo del comportamiento del determinado use-case

Entiendo que la siguiente tarea en el backlog es implementar un sistema de control de versiones, conectar GitHub con Databricks y certificar a todo el equipo que mantiene el codebase en Git. Aquí tienes una descripción detallada de esta tarea:

Implementar el Sistema de Control de Versiones y Certificación de Uso de Git

Descripción:

Esta tarea es fundamental para establecer una gestión eficaz del código y garantizar la colaboración efectiva en el proyecto. Implica la implementación de un sistema de control de versiones (VCS) y la conexión de Databricks con GitHub o el servicio correspondiente (devops probablemente). Además, se llevará a cabo un proceso de certificación para garantizar que todos los miembros del equipo estén utilizando Git de manera adecuada.

Pasos clave:

Selección de plataforma: Decidir si github, gitlab, o azure

Conexión de Databricks y plataforma: se Integra Databricks con *GitHub* para permitir la gestión y colaboración de código desde la plataforma Databricks. Esto implica configurar integraciones o conexiones según la infraestructura de tu proyecto.

Certificación del Uso de Git: Organiza sesiones de capacitación o talleres para el equipo con el objetivo de certificar que todos los miembros comprenden y utilizan Git de manera efectiva. Esto puede incluir evaluaciones prácticas.

Complejidad Técnica (del 1 al 5): 3

La implementación de un VCS y la conexión de Databricks con *GitHub* requieren conocimientos técnicos sólidos, pero son prácticas comunes en la industria.

Prioridad Técnica (del 1 al 5): 5

Esta tarea es alta prioridad ya que establecer un sistema de control de versiones eficaz y garantizar que todo el equipo lo utilice correctamente es fundamental para el éxito del proyecto y la colaboración en equipo.

En resumen, esta tarea es esencial para mejorar la gestión del código, la colaboración y la trazabilidad en el proyecto de ciencia de datos. Al implementar un VCS y certificar a todo el equipo en el uso de Git, se establecerán bases sólidas para un desarrollo de código más eficiente y un control de versiones adecuado.

Entiendo que la siguiente tarea es aplicar Hyperopt para la optimización de hiperparámetros y realizar la selección de características para los modelos de los dos Use Cases, Promotion Analytics y Price Elasticities. Aquí tienes una descripción de esta tarea:

Optimización de Hiperparámetros y Selección de Características para Modelos

Descripción:

Esta tarea implica aplicar técnicas de optimización de hiperparámetros y seleccionar las características más relevantes para mejorar el rendimiento de los modelos en los dos Use Cases: Promotion Analytics y Price Elasticities.

Pasos clave:

Optimización de Hiperparámetros (Hyperopt): se Utilizará la biblioteca Hyperopt u otra herramienta de optimización de hiperparámetros para ajustar automáticamente los valores de los hiperparámetros de los modelos. Esto se realiza a través de la búsqueda de hiperparámetros que maximicen o minimicen una métrica de rendimiento específica (por ejemplo, precisión, RMSE, F1-score).

Selección de Características: se Realizará un análisis exhaustivo de las características disponibles en los datos. Utiliza técnicas como la importancia de características, análisis de correlación y eliminación de características irrelevantes o redundantes. Identifica las características más informativas y elimina aquellas que no contribuyen significativamente a la precisión del modelo.

Ajuste de Modelos: se Aplicarán los hiperparámetros optimizados y las características seleccionadas a los modelos de Promotion Analytics y Price Elasticities. Se entrenan los modelos con los nuevos ajustes para mejorar su rendimiento.

Validación y Evaluación: Se Evalúan los modelos ajustados utilizando métricas de evaluación apropiadas para cada Use Case. Asegúrate de que los modelos optimizados estén mejorando el rendimiento en comparación con los modelos originales.

Documentación y Comunicación: Documenta claramente los hiperparámetros seleccionados y las características utilizadas en cada modelo. Comunica los resultados y las mejoras de rendimiento al equipo y a los interesados relevantes.

Complejidad Técnica (del 1 al 5): 4

La optimización de hiperparámetros y la selección de características pueden ser tareas técnicamente desafiantes que requieren un buen conocimiento de los algoritmos de aprendizaje automático y la comprensión de los datos.

Commented [DMJE1]: Cambiar a que implica la complejidad de implementacion de hyperopt (aunque podemos delegar eso a databricks)

Prioridad Técnica (del 1 al 5): 4

La optimización de hiperparámetros y la selección de características son actividades fundamentales para mejorar el rendimiento de los modelos y, por lo tanto, son de alta prioridad para el éxito de los Use Cases.

En resumen, esta tarea es esencial para mejorar la precisión y el rendimiento de los modelos en los Use Cases de Promotion Analytics y Price Elasticities. Al aplicar técnicas de optimización de hiperparámetros y selección de características, podrás obtener modelos más efectivos y resultados más precisos en tus análisis de ciencia de datos.

Utilizar PySpark en los Procesos de Transformación de Datos en lugar de Pandas

Descripción: Esta tarea se centra en reemplazar el uso de Pandas por PySpark en los procesos de transformación de datos en el proyecto de ciencia de datos.

Pasos clave:

Reescritura de Código: se Adaptan los procesos de transformación de datos que originalmente se implementaron con Pandas para que funcionen con PySpark. Esto implica modificar el código existente para utilizar las estructuras de datos y las operaciones proporcionadas por PySpark en lugar de las de Pandas.

Pruebas y Validación: se Realizarán pruebas para asegurarte de que los resultados obtenidos con PySpark sean coherentes con los resultados anteriores obtenidos con Pandas.

Capacitación del Equipo: Es necesario alinear al equipo en el uso de PySpark y las diferencias clave entre PySpark y Pandas.

Complejidad Técnica (del 1 al 5): 3

La migración de Pandas a PySpark mas que complejo es un proceso laborioso, que implica cambiar y testear de nuevo el funcionamiento del mismo.

Prioridad Técnica (del 1 al 5): 4

Esta tarea es importante ya que PySpark es beneficioso cuando se trabaja con grandes volúmenes de datos y se requiere un procesamiento distribuido.

En resumen, la tarea de utilizar PySpark en lugar de Pandas en los procesos de transformación de datos es una decisión estratégica que puede mejorar la eficiencia y escalabilidad de los procesos de preparación de datos en el proyecto de ciencia de datos. Brindemos especial atención a que la transición se realice de manera suave y que los resultados sean consistentes con los obtenidos previamente con Pandas.

Implementación de MLflow y MLOps

Descripción:

Esta actividad implicaría la implementación conjunta de MLflow en Databricks y la estrategia de MLOps en el proyecto. Ya que MLflow es una parte fundamental de la estrategia de MLOps.

Esta actividad se centra en la implementación de MLflow en la plataforma Databricks. MLflow ayuda en el ciclo de vida de Machine Learning y permite el seguimiento, la reproducción y la gestión de experimentos de Machine Learning. Al implementar MLflow, se facilita la colaboración y el seguimiento de modelos en los Use Cases de Promotion Analytics y Price Elasticities.

Complejidad Técnica (del 1 al 5): 3

Igual que la migración de Pandas a PySpark, más que complejo es un proceso laborioso, que implica cambiar partes puntuales del código (selección de parámetros y creación de modelos) y validar el correcto funcionamiento de este.

Prioridad Técnica (del 1 al 5): 4

Refactorizar el código y añadir las optimizaciones de paralelismo

Descripción: Esta tarea se centra en mejorar la calidad y el rendimiento del código existente mediante la refactorización y la introducción de optimizaciones de paralelismo. El objetivo es hacer que el código sea más eficiente y capaz de aprovechar al máximo los recursos de hardware disponibles.

Pasos clave:

Análisis de Código: Se Comienza por realizar un análisis exhaustivo del código existente para identificar áreas que necesiten refactorización. Esto puede incluir la simplificación de algoritmos, la eliminación de redundancias y la mejora de la estructura del código.

Optimizaciones de Paralelismo: se Identificaron las oportunidades para introducir paralelismo en el código

Reescritura de Código: Refactoriza el código de acuerdo con las mejoras identificadas. Asegúrate de que la refactorización sea coherente con las buenas prácticas de programación y la legibilidad del código.

Pruebas y Validación: Realiza pruebas exhaustivas para asegurarte de que el código refactorizado funcione correctamente y produzca los mismos resultados que el código original. Además, mide y compara el rendimiento del código antes y después de las optimizaciones de paralelismo.

Gestión de Recursos: Asegúrate de gestionar adecuadamente los recursos del sistema al introducir paralelismo, evitando problemas como la concurrencia no deseada o la falta de recursos.

Documentación: Actualiza la documentación del código para reflejar los cambios realizados durante la refactorización y las optimizaciones de paralelismo.

Complejidad Técnica (del 1 al 5): 5

La complejidad de esta tarea puede variar según el tamaño y la complejidad del código existente y la introducción de paralelismo.

Prioridad Técnica (del 1 al 5): 5

La refactorización y la optimización de paralelismo son importantes para mejorar el rendimiento y la eficiencia del software, especialmente en aplicaciones que manejan grandes cantidades de datos o cálculos intensivos.

En resumen, esta tarea tiene como objetivo mejorar la calidad y el rendimiento del código existente a través de la refactorización y la introducción de optimizaciones de paralelismo. Al hacerlo, se busca hacer que el código sea más eficiente y capaz de aprovechar mejor los recursos disponibles, lo que puede llevar a un mejor rendimiento de la aplicación.

Areas de oportunidad Data enigeering

Implementar el Gobierno de Datos en el Proyecto Revenue Management

Descripción: Esta tarea se enfoca en establecer un marco de gobierno de datos sólido en el proyecto de Revenue Management, abarcando tanto el Single Source of Truth (SSOT) como los dos Use Cases (Promotion Analytics y Price Elasticities). El gobierno de datos es fundamental para garantizar la calidad, la seguridad, la trazabilidad y el cumplimiento normativo de los datos en todo el ciclo de vida del proyecto.

Pasos clave:

Definir Políticas y Normativas: Establecer políticas y normativas claras que rijan la recopilación, el almacenamiento, el procesamiento y el uso de datos en el proyecto. Esto incluye políticas de privacidad, políticas de seguridad de datos y reglas para el acceso a los datos.

Catalogar y Documentar Datos: Crear un catálogo de datos que describa detalladamente los conjuntos de datos utilizados en el SSOT y los Use Cases. Esto incluye metadatos relevantes como la fuente de datos, la estructura, la calidad y las transformaciones aplicadas. Documentar los procesos de transformación de datos y mantener registros actualizados.

Gestión de Calidad de Datos: Implementar prácticas y herramientas para garantizar la calidad de los datos en el SSOT y los Use Cases. Esto puede incluir validación, limpieza y estandarización de datos. Establecer métricas de calidad de datos y realizar auditorías periódicas.

Seguridad y Cumplimiento: Asegurar la protección de los datos y el cumplimiento de las regulaciones pertinentes, como el RGPD. Establecer políticas de acceso a datos basadas en roles y llevar un registro de auditoría de accesos.

Gestión de Metadatos: Implementar una estrategia de gestión de metadatos para facilitar el descubrimiento y la comprensión de los datos en el proyecto. Esto incluye información sobre la fuente de datos, definiciones de campos y transformaciones aplicadas.

Capacitación y Concientización: Capacitar al equipo en prácticas de gobierno de datos y concientizar sobre la importancia del cumplimiento de políticas y normativas de datos.

Complejidad Técnica (del 1 al 5): 4

La implementación de un gobierno de datos completo puede ser técnicamente compleja, ya que involucra múltiples aspectos de la gestión de datos y la seguridad.

Prioridad Técnica (del 1 al 5): 5

El gobierno de datos es fundamental para asegurar la calidad y la integridad de los datos, así como para cumplir con las regulaciones y las mejores prácticas de gestión de datos.

Esta tarea es esencial para garantizar que el proyecto de Revenue Management opere de manera efectiva y segura en términos de datos. Asegura que los datos sean confiables, estén disponibles cuando se necesiten y cumplan con las normativas pertinentes.

Implementar una Arquitectura de Datos en el Proyecto Revenue Management

Descripción:

Esta tarea se enfoca en diseñar y desplegar una arquitectura de datos que sea adecuada para el proyecto de Revenue Management. La arquitectura de datos define cómo se recopilan, almacenan, procesan y acceden los datos en el proyecto, y es esencial para garantizar un flujo de datos efectivo y una gestión adecuada.

Pasos clave:

Definición de Requisitos: Comprender las necesidades del proyecto, incluyendo los requisitos de datos de los Use Cases de Promotion Analytics y Price Elasticities, así como del Single Source of Truth (SSOT).

Diseño de la Arquitectura: Diseñar una arquitectura de datos que incluya componentes como la capa de almacenamiento de datos, la capa de procesamiento, la capa de acceso a datos y la gestión de metadatos. Definir cómo se integran estas capas y cómo se asegura la escalabilidad y la disponibilidad.

Modelado de Datos: Diseñar modelos de datos que reflejen la estructura y la relación de los datos en el proyecto. Esto puede incluir modelos de bases de datos relacionales o esquemas de datos no relacionales, según las necesidades.

Implementación de la Arquitectura: Implementar la arquitectura de datos de acuerdo con el diseño. Esto incluye la configuración de bases de datos, la creación de pipelines de procesamiento de datos y la implementación de políticas de seguridad y acceso.

Integración con los Use Cases: Asegurar que los Use Cases de Promotion Analytics y Price Elasticities se integren de manera efectiva con la arquitectura de datos. Esto incluye la transferencia de datos desde el SSOT a los Use Cases y la implementación de pipelines de datos.

Pruebas y Validación: Realizar pruebas exhaustivas para asegurarse de que la arquitectura de datos funcione correctamente y cumpla con los requisitos del proyecto. Esto incluye pruebas de rendimiento y pruebas de integridad de datos.

Documentación: Documentar la arquitectura de datos, los modelos de datos, los flujos de datos y los procedimientos de gestión. Esta documentación es esencial para la comprensión y el mantenimiento continuo de la arquitectura.

Complejidad Técnica (del 1 al 5): 4

La implementación de una arquitectura de datos puede ser técnica y requerir conocimientos sólidos en diseño de datos y tecnologías relacionadas.

Prioridad Técnica (del 1 al 5): 5

La arquitectura de datos es fundamental para garantizar una gestión efectiva de los datos en el proyecto de Revenue Management y es de alta prioridad.

En resumen, esta tarea es esencial para diseñar y desplegar una arquitectura de datos eficiente y escalable en el proyecto de Revenue Management. Una arquitectura sólida permitirá una gestión de datos efectiva y asegurará que los Use Cases puedan acceder y utilizar los datos de manera eficiente.

Refactorización y Optimización de Procesos de Ingeniería de Datos

Descripción: Esta tarea tiene como objetivo revisar y mejorar los procesos de ingeniería de datos en el proyecto, con un enfoque en la refactorización y la optimización. La refactorización implica reorganizar y simplificar el código existente, mientras que la optimización busca mejorar el rendimiento y la eficiencia de los procesos.

Pasos clave:

Análisis de Procesos: Realiza un análisis detallado de los procesos de ingeniería de datos existentes en el proyecto. Identifica áreas que puedan beneficiarse de refactorización y optimización.

Refactorización de Código: Modifica el código existente para hacerlo más legible y mantenible. Elimina redundancias, mejora la estructura y sigue las mejores prácticas de codificación.

Gestión de Recursos: Asegúrate de que los procesos de ingeniería de datos utilicen eficientemente los recursos disponibles, como CPU y memoria.

Pruebas y Validación: Realiza pruebas exhaustivas para asegurarte de que los cambios realizados durante la refactorización y la optimización no introduzcan errores y que los procesos funcionen correctamente.

Documentación: Actualiza la documentación de los procesos de ingeniería de datos para reflejar los cambios realizados. Esto es importante para garantizar que otros miembros del equipo comprendan los procesos actualizados.

Complejidad Técnica (del 1 al 5): 3

La refactorización y la optimización de procesos de ingeniería de datos pueden ser técnicamente desafiantes, pero la complejidad depende de la cantidad y la complejidad de los procesos existentes.

Prioridad Técnica (del 1 al 5): 4

La mejora de la eficiencia y la calidad de los procesos de ingeniería de datos es importante para asegurar que los datos estén disponibles y sean confiables para su uso en el proyecto.

En resumen, esta tarea tiene como objetivo mejorar la eficiencia y la calidad de los procesos de ingeniería de datos en el proyecto a través de la refactorización y la optimización. Al hacerlo, se busca garantizar que los datos se gestionen de manera eficiente y confiable en todo el ciclo de vida del proyecto.