Organización de Datos
Trabajo Práctico 2
Predicciones

Integrante	Padrón	Mail
RAMUNDO, Matias Ezequiel	100327	matiramundo@gmail.com
ROSAS, Martín Alejandro	98535	martinrosas321@gmail.com
OTERO, Lucas	99192	otero.lucas@outlook.com
GALLARDO, Emiliano Alexis	96959	eagallardo@fi.uba.ar

Repositorio: https://github.com/MartinRosasSommer/Datos2019

Introducción

Jampp es una empresa que ayuda a anunciantes a promover sus apps a nivel global y también a recuperar a aquellos usuarios que ya instalaron la app pero están inactivos. Para esto desarrolla modelos de Machine Learning y así poder tomar decisiones en base a los datos obtenidos.

Objetivo

El trabajo práctico tiene como objetivo predecir para una lista de dispositivos, el tiempo que tardarán en reaparecer en subastas y el tiempo que tardarán los usuarios para instalar la aplicación mostrada en la publicidad.

Análisis del problema:

Al comenzar, se analizó el enunciado y decidió que ambos eran problemas de regresión. Costó un poco decidir cómo atacar el problema, dado que los únicos datos que se reciben en la competencia de kaggle eran los identificadores de dispositivo. Esto obligaba a preparar los features para predecir esos ids a partir del mismo dataset (installs, events, clicks y auctions) con el que íbamos a entrenar los predictores.

Como en ambos problemas a resolver el tiempo máximo a predecir era de 3 días, se decidió separar los datasets en periodos de ese tiempo para que al calcular diferencias entre apariciones/instalaciones nunca se produzca una de mayor distancia en tiempo.

Al mismo tiempo eso sirvió para poder comparar performances entre esos periodos de tiempo según cual se usaba para entrenar y cual para testear.

Los algoritmos utilizados:

1) **Linear Regression**: encuentra una función lineal (una línea recta no vertical) que predice los valores de las variables dependientes en función de las independientes.
2) **Bayesian Ridge Regression:** es una regresión lineal que actualiza la probabilidad a medida que se va agregando información.
3) **Lasso**: es un método de análisis de regresión que realiza selección de variables y regularización para mejorar la exactitud e interpretabilidad del modelo estadístico producido por este.
4) **XGBoost:** Produce un modelo predictivo en forma de un conjunto de modelos de predicción débiles, típicamente árboles de decisión. Construye el modelo de forma escalonada como lo hacen otros métodos de boosting, y los generaliza
5) **Random Forest(Regression):** es una combinación de árboles predictores tal que cada árbol depende de los valores de un vector aleatorio probado

Resolución del problema #1

Predecir, para cada aparición de un dispositivo en una subasta, cuánto tiempo en segundos tardaría en volver a aparecer.

Al ver que lo que se busca predecir es un valor continuo y no una clase se asumió que este es un problema de regresión, pero antes de empezar a predecir, el problema principal era cómo llegar a los valores objetivo para cada ID, ya que para entrenar al regresor se necesita un X y un Y.
Para encontrar el tiempo en segundos objetivo se pensó en agrupar por cada dispositivo y, al tener ordenado el tiempo de cada aparición, medir un tiempo delta pera cada aparición hasta la siguiente. Otra solución era calcular el tiempo promedio de tardanza en reaparecer por cada dispositivo, pero se decidió utilizar la primera opción a pesar de ser menos performante ya que parecía ser más precisa.
Al intentar esa solución, las limitaciones de hardware de los miembros del grupo y el tamaño de los documentos csv provocaban repetidas veces la muerte del kernel de Jupyter. Por lo cual una gran parte del trabajo práctico se resolvió usando Apache Spark, el cual a pesar de tardar en procesar los datos, al menos permitía llegar a un resultado.
Una vez obtenida la lista de tiempos delta por apariciones, comenzó la búsqueda de features.

Feature engineering

Se priorizó aprovechar datos importantes encontrados en el análisis exploratorio del TP1, como por ejemplo obtener el sistema operativo del Dispositivo a partir de la información de instalaciones. También features a partir de fechas como por ejemplo la hora en la que había más actividad y valores booleanos indicando si era fin de semana o si era de noche.
A partir de los eventos se pudo conseguir información de las aplicaciones en las que más actividad tienen los usuarios, y hacer una diferencia entre las aplicaciones que más usó según el día y momento del día.

Preparación del dataset y Entrenamiento

Para entrenar el modelo hubo que codificar los features no numéricos. Para esto se usó el Label Encoder de la misma biblioteca Sklearn.
El error de las predicciones fue medido con mean-squared-error aunque al hacer submits descubrimos que no necesariamente nos daban mejor puntaje las predicciones con menor error local.
La porción del dataset que mejor funcionó para predecir fue los primeros 3 días.

El algoritmo de Machine Learning que usamos para predecir y hacer la mayoría de los submits fue Random Forest de la biblioteca Sklearn de Python, pero luego conseguimos una mejor performance ajustando los hiperparametros de XGBoost, con el cual conseguimos nuestro mejor puntaje hasta el momento. XGBoost y Random Forest fueron los que devolvieron mejores resultados pero al mismo tiempo eran los que ocasionaron que menos veces se traben las notebooks y haya que reiniciarlas. Otros algoritmos que se intentaron usar fueron: Linear Regression, Lasso y Gradient Boosting,

Resolución del Problema #2

Predecir , para cada aparición de un dispositivo en una subasta, cuánto tiempo en segundos tardará en convertir.

Aquí también lo que se busca predecir es un valor continuo por lo cual se asumió que es un problema de regresión. La diferencia con el problema #1 es que en éste la diferencia de tiempo buscada es entre una aparición en una subasta y una instalación. Para los casos que no instalen se decidió asumir que tardan el tiempo máximo posible (3 días). En este caso también se usaron los primeros 3 días de datos porque fueron los que dieron mejores resultados al predecir.

Feature Engineering

Los features utilizados en la mayoría de las pruebas coincidieron con el problema #1, aunque en este caso pareció influir negativamente agregar las aplicaciones más usadas por cada dispositivo como feature. En este caso sirvió más enfocarse en la actividad en subastas anteriores, separándolas por dia y por momento del dia. Las codificaciones también se hicieron con LabelEncoder.

Preparación del dataset y entrenamiento.

Este problema 2 es más complejo porque para entrenar primero hay que encontrar, para cada aparición en subasta, el caso más próximo de instalación para ese dispositivo por lo cual no alcanza con encontrar coincidencias de ID.
Para resolver este problema, también se usó apache spark, ya que en este caso se necesitaba incluso más memoria que en el problema anterior porque entraban en juego los datos de las instalaciones.
Se concatenaron los dataframes de instalaciones y subastas y, luego de agrupar por dispositivo, se calculó la distancia en segundos desde cada aparición en subasta hasta la instalación más próxima. Así se consiguieron todos los deltas para los primeros y segundos 3 días, para luego entrenar y testear.

Los mejores algoritmos fueron Random Forest y XGBoost aunque, al igual que en el problema anterior, las limitaciones de hardware que tenía el equipo resultaron en no poder dedicar el tiempo deseado evaluando distintos híper parámetros para ciertos modelos.
A pesar de eso se pudo mejorar bastante el puntaje desde los primeros submits (Mayoría con Random Forest) hasta el último (XGBoost).

Conclusión

Luego de utilizar distintos algoritmos de regresión, parece ser que los mejores para predecir este tipo de problemas son XGBoost y Random Forest, aunque no tienen una performance

muy alejada de la del resto de los algoritmos probados. Más bien parece ser que la gran diferencia la hacen los features. En este caso los que más influyeron fueron el sistema operativo del dispositivo y la historia del dispositivo dividiéndola por dias.
Los demás features relacionados con las fechas en su mayoría tuvieron influencias negativas en la performance de los modelos.