UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

Arquitectura NRT para a predición de calidade nun proceso de minaría

Autor:
Pablo Rey Fernández

Titores:
Anselmo Tomás Fernández Pena
Ismael Rodríguez Fernández

Máster en Big Data: Tecnoloxías de Análise de Datos Masivos

Xullo de 2019

Traballo de Fin de Máster presentado na Escola Técnica Superior de Enxeñaría da Universidade de Santiago de Compostela para a obtención do Máster en Big Data: Tecnoloxías de Análise de Datos Masivos

Índice xeral

Índice de figuras

Índice de cadros

Capítulo 1

Introdución

O obxectivo deste proxecto é o de demostrar a utilidade e a versatilidade das arquitecturas Big Data de tipo NRT (Near Real-Time) para o apoio á toma de decisións en situacións complexas nas que o tempo de resposta é un factor crítico. Por exemplo, nas plantas de flotación de minerais é habitual que as estimacións da pureza dos compostos resultantes das extraccións dependan de medicións de laboratorio que poden tardar horas en ser elaboradas. Non ter unha estimación en tempo real do rendemento do proceso supón unha traba importante á hora de xestionar os recursos destes complexos.

Fronte ás estratexias baseadas no procesamento por lotes ou *batch*, empregadas con frecuencia no ámbito do Big Data para a inxección masiva de datos en sistemas informacionais, as arquitecturas NRT preséntanse como unha alternativa adecuada para os casos nos que, ademais de traballar con volumes de datos moi grandes e heteroxéneos, o feito de obter unha resposta do sistema nun tempo razoable é un elemento crítico para acadar un determinado obxectivo de negocio. Os sistemas NRT permiten o procesamento case instantáneo do dato, de xeito que a lóxica pode ser aplicada xa durante a recollida da información, resultando en predicións, alertas ou recomendacións temperás.

No contexto do caso de uso presentado, poder realizar unha predición do grao de pureza do metal que resulta do proceso de flotación cada vez que se reciben medicións da composición da mestura primitiva e das condicións ambientais da propia planta, no canto de ter que esperar durante horas polos resultados das probas de laboratorio, pode ser un feito determinante para o bo funcionamento do complexo. Entre outras cousas, poderían facerse os axustes necesarios para corrixir ou aproveitar unha situación anómala, ben evitando un risco ou ben xerando un beneficio engadido. Outras posibles aplicacións serían a detección en tempo real de situacións de perigo para a integridade da planta (anomalías na temperatura, humidade...) ou a creación de cadros de mando dinámicos.

1.1. Contexto

Na actualidade existen distintas arquitecturas de referencia para o procesamento de datos masivos. En xeral, divídense en dous grandes grupos que se diferencian en canto ao xeito de tratar o dato. Nas arquitecturas baseadas no procesamento por lotes ou *batch*, recóllese e almacénase un conxunto de datos durante un tempo e nun determinado momento procésase para pasar a formar parte dun sistema analítico. Pola contra, nas arquitecturas NRT, baseadas en *streams* ou de procesamento *online*, o dato procésase de xeito individualizado tan pronto como se recibe. A aplicación dun ou doutro paradigma depende das necesidades do caso de uso sobre o que se pretendan poñer en marcha. O procesamento por lotes é máis adecuado cando a necesidade de tratar grandes volumes de datos sen interromper o funcionamento dos sistemas operacionais de orixe supera os posibles beneficios de obter resultados analíticos de xeito rápido. As arquitecturas NRT son apropiadas en casos nos que é crítico que a análise e resposta ante a ocorrencia dun ou varios eventos se dé nun tempo moi pequeno, como por exemplo en casos como de detección de anomalías, xestión de recursos en tempo real ou recomendacións. Nos últimos anos gañaron popularidade outros paradigmas como as arquitecturas Lambda ou Kappa que combinan e amplían aspectos das dúas aproximacións para dar unha mellor resposta en situacións específicas.

No caso deste proxecto, centrarémonos en poñer de manifesto as vantaxes dunha arquitectura de datos NRT fronte a unha arquitectura de procesamento por lotes nun escenario relativamente complexo: un entorno industrial relacionado coa minaría. Dispoñemos dun conxunto de datos que recolle a información de diversos sensores situados no interior da cámara onde se realiza a flotación, un proceso fisicoquímico de tres fases que ten por obxectivo a separación das especies minerais e metais do resto de substancias mediante a adhesión selectiva de partículas a burbullas de aire (ver 1.1). Neste caso trátase dun proceso de separación de ferro a partir dun concentrado que contén ademais auga e outros químicos. Como xa se mencionou, as medicións do grao de pureza do resultado destes procesos son custosas en tempo e recursos xa que esixen unha serie de experiencias de laboratorio. O obxectivo será, por tanto, tratar de facer unha predición en tempo real da pureza do composto de saída segundo as propiedades da solución base e as condicións da propia planta durante o proceso.

En canto ás tecnoloxías, o estudo centrarase en dúas: Kafka e MLflow. Kafka é unha plataforma distribuída de *streaming* de mensaxes baseada nos sistemas de colas publica-subscribe, que ten a capacidade de xestionar fluxos continuos de datos de xeito eficiente e escalable, almacenamento persistente con tolerancia a erros e que permite o procesamento dos rexistros *on the go*. Utilízase habitualmente para a construción de *pipelines* de datos en tempo real para comunicar sistemas e aplicacións ou ben para construír aplicacións de *streaming* en tempo

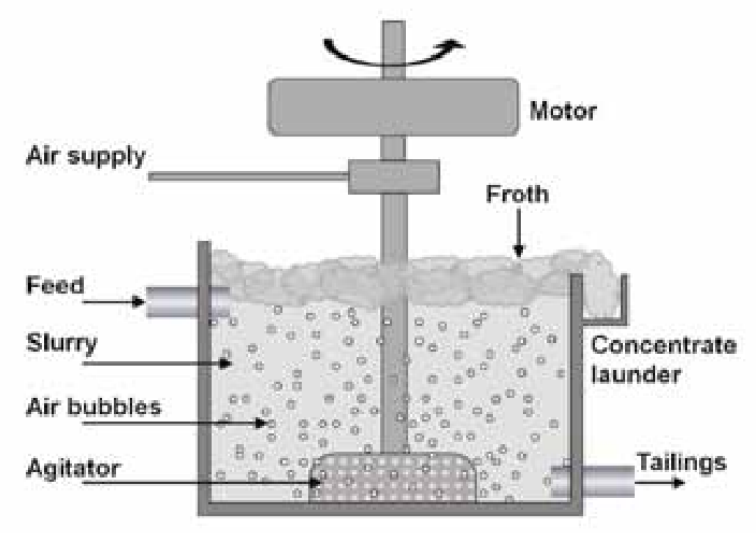

Figura 1.1: Esquema do proceso de flotación

real que transforman ou reaccionan aos datos. En concreto, centrarémonos no uso de KSQL, unha ferramenta relativamente recente proposta por Confluent, empresa que dispón dun ecosistema xestionado baseado en Apache Kafka, para definir a lóxica de procesamento do dato.

Doutra banda, MLFlow é a proposta de Databricks para a xestión integral do ciclo de vida dos modelos de Machine Learning, dende o lanzamento de experimentacións colaborativas, configurables e reproducibles de xeito sinxelo, pasando polo *tracking* dos entrenamentos e poñendo a disposición dos equipos de científicos de datos ferramentas para a produtivización automática dos modelos.

Por último, estudaremos diferentes alternativas, tanto *IaaS* como *PaaS*, de entre as que facilitan os principais proveedores de servizos na nube para aloxar cada unha das partes da solución no espazo que máis lle conveña en función das súas características. Ademais, trataremos de obter unha representación dos resultados en tempo real, idealmente mediante un cadro de mando con gráficas.

1.2. Obxectivos

Unha vez estudada a problemática e as características da solución a desenvolver, identificamos os seguintes obxectivos:

OBX-01	**Construír unha arquitectura Big Data que permita o procesamento de datos en tempo Near Real-Time de xeito eficiente**
Descrición	O principal obxectivo do proxecto é o de deseñar e implementar un sistema que permita o procesamento masivo de información en tempo NRT. O sistema debe permitir que a lóxica resida embebida na propia plataforma, de forma que se poida extraer valor do dato tan pronto sexa posible. Estudaranse as alternativas máis modernas dispoñibles e tratarán de adaptarse ao caso de uso do proxecto.

OBX-02	**Analizar e modelar o problema da predición de pureza durante o proceso de flotación**
Descrición	Estudaranse as características do problema presentado anteriormente para determinar as transformacións necesarias sobre o conxunto de datos inicial para poder adestrar e avaliar un modelo de regresión que permita facer unha predición de calidade que satisfaga as necesidades enunciadas.

OBX-03	**Desenvolver unha solución que permita xestionar o ciclo de vida dos modelos**
Descrición	O modelado estará ligado á construción dunha solución que permita a traza e lanzamento automatizado de experimentos de Machine Learning así como a produtivización de modelos.

OBX-04	**Determinar se as solucións e ferramentas empregadas son o suficientemente robustas como para ser levadas a un proxecto real**
Descrición	Durante o desenvolvemento do proxecto deberán estudarse e probarse tecnoloxías modernas tanto para a parte da experimentación e produtivización de modelos como para a arquitectura, co obxectivo de determinar se estas están o suficientemente maduras como para ser aplicadas nun proxecto real.

1.3. Alcance

A solución proposta deberá dar resposta aos requirimentos identificados. Dunha banda, deberase propor unha arquitectura capaz de manexar de xeito eficiente un fluxo de eventos masivo en tempo real que simule as medicións da planta. Deberemos analizar e transformar o conxunto de datos para decidir a mellor estratexia de cara ao deseño dun modelo preditivo capaz de xerar estimacións de pureza razoables. Este proceso irá acompañado do desenvolvemento dunha plataforma que permita un proceso cómodo para a experimentación e comparativa de modelos, así como a súa posterior produtivización. Por último, buscarase unha alternativa que permita que a predición poida realizarse durante a inxestión dos datos e algún tipo de solución para a visualización dos resultados.

Identificamos, polo tanto, como entregables do proxecto:

1. Deseño e implementación da arquitectura de datos.
2. Sistema de seguimento de experimentos e produtivización de modelos.
3. Análise, deseño, entrenamento e produtivización dun modelo de regresión que permita resolver o problema presentado.
4. Modelo de predición embebido no propio *pipeline* de inxestión de datos.

1.4. Glosario de termos

- **Computación na nube**: Modelo de computación que permite a contratación baixo demanda de recursos computacionais, almacenamento ou procesamento, sen que o usuario deba manexar os detalles de baixo nivel.
- **IaaS**: Siglas de Infraestructure As A Service, paradigma dentro do modelo de computación na nube que ofrece aos usuarios infraestrutura TIC administrada sobre a cal teñen o control do sistema operativo, almacenamento e software.
- **PaaS**: Siglas de Platform As A Service, paradigma dentro do modelo de computación na nube que se centra en facilitar aos usuarios o desenvolvemento e administración do software sen ter que se preocupar da xestión da infraestrutura subxacente.
- **Modelo de Machine Learning**: Cada un dos algoritmos ou ferramentas estatísticas que permiten que os computadores realicen certas tarefas en base a patróns e inferencia en lugar de ter recibido instrucións explícitas. Dise que os algoritmos se *adestran* cun subconxunto da información dispoñible para construír modelos matemáticos dos que poder tirar predicións.

Capítulo 2

Deseño do pipeline

Nesta sección presentaremos o conxunto de datos utilizado e a arquitectura construída para simular e tratar o intercambio de información dentro da planta. O conxunto de datos está dispoñible a través da web *Kaggle*, comunidade *online* de científicos de datos que comparten datos e coñecementos relacionados co Machine Learning ademais de organizar competicións e retos. Os datos[1] puxéronse a disposición do público co obxectivo de investigar se a partir da información dispoñible se podería facer unha estimación do grao de pureza do concentrado de saída do proceso de flotación tal e como se comentou no capítulo 1.

O *pipeline* de datos estará construído sobre a plataforma Confluent Kafka. Tentaremos aproveitar as distintas abstraccións que nos facilita a ferramenta para demostrar que se poden aplicar en arquitecturas complexas. Deste xeito, trataremos de axudarnos da propia plataforma para enriquecer o dato durante a inxesta e estar así en condicións de facer un procesado con lóxica complexa do mesmo.

Utilizaremos KSQL para definir os fluxos e transformacións de información sobre o clúster na medida do posible. KSQL non só conta actualmente cunha listaxe extensa de funcións, senón que nos permite codificar a nosa propia lóxica e engadila á súa gramática sempre que sigamos unha regras. Faremos uso destas funcións personalizadas, as UDF (User Defined Functions), para chamar ao servizo que nos devolverá a predición en tempo real con cada nova medición.

[1] Ver `https://www.kaggle.com/edumagalhaes/quality-prediction-in-a-mining-process`

2.1. Conxunto de datos

O conxunto de datos de partida contén seis meses de información real sobre o proceso de flotación de ferro nun complexo mineiro (dende marzo a setembro de 2017). As observacións teñen información da data e hora de recollida, non obstante, a frecuencia coa que se extraen as mostras non é regular; depende da columna. Así, hai observacións con cambios en certas columnas cada 20 segundos, mentres que para outras medidas só aparecen modificacións cada hora. O obxectivo é predicir a cantidade de sílice na solución, a cal representa o grao de impureza do concentrado trala flotación e que os enxeñeiros deben tratar de minimizar. Poder ter de antemán esta información axudaríalles á hora de planificar accións correctivas, como aumentar ou diminuír a cantidade de masa que se desbota como refugallo. O conxunto de datos contén as seguintes 24 columnas:

- **Data**: día e hora da medición.
- **% Iron Feed**: porcentaxe de ferro na masa de entrada.
- **% Silica Feed**: procentaxe de sílice (impureza) na masa de entrada.
- **Starch Flow**: medida do fluxo de amidón, que se utiliza como reactivo, na masa de entrada (en m^3/h).
- **Amina Flow**: medida do fluxo de aminas, que se utilizan como reactivo, na masa de entrada (en m^3/h).
- **Ore Pulp Flow**: medida do fluxo da masa de entrada (en t/h).
- **Ore Pulp PH**: grao de acidez da masa de entrada (de 0 a 14).
- **Ore Pulp Density**: densidade da masa de entrada (de 1 a 3 kg/m^3).
- **Flotation Column X Air Flow**: medida do fluxo de aire que entra nas cámaras de flotación (en Nm^3/h, a X refírese a cada unha das 7 cámaras).
- **Flotation Column X Level**: nivel de escuma nas cámaras de flotación (en mm, a X refírese a cada unha das 7 cámaras).
- **% Iron Concentrate**: porcentaxe de ferro no concentrado resultante (medido tras un estudo de laboratorio).
- **% Silica Concentrate**: porcentaxe de sílice (impureza) no concentrado resultante (medido tras un estudo de laboratorio).

2.2. Deseño do clúster Kafka

En canto á tecnoloxía seleccionada para a construción do *pipeline* NRT, elixi-mos Confluent Kafka. Como xa mencionamos en 1.1, Kafka é unha plataforma distribuída de *streaming* para o tratamento de colas de mensaxes. Os principais elementos de Kafka son os *topics*, cada unha das colas nas que se almacenan os eventos, e os produtores-consumidores, compoñentes que xeran ou consomen in-formación das colas. Coa chegada de Kafka Streams, introduciuse a capacidade de engadir lóxica no proceso de inxesta de datos e dúas novas abstraccións para o manexo de información en *streaming*: *Streams* e *KTables*. Cada unha das dúas permite traballar co mesmo *topic* de xeitos distintos sen facer modificacións sobre a súa estrutura interna, permitindo unha gran flexibilidade á hora de definir as transformacións (joins *stream-stream*, *stream-table*, ventás móbiles...). A evolu-ción máis recente sobre este paradigma é KSQL, unha linguaxe declarativa cunha sintaxe similar a SQL desenvolvida por Confluent e que funciona sobre Kafka. A principal vantaxe é que supón unha interface moi sinxela para o prototipado e construción de *pipelines* de transformación de datos que pode ser explotada mesmo por usuarios sen coñecementos en linguaxes como Java ou Scala para ob-ter resultados analíticos en tempo real. Ademais do conxunto de funcións típicas que inclúe, é posible estender as súas funcionalidades con función definidas polo usuario (UDFs) programadas en Java. Nós optaremos por esta alternativa para realizar unha petición HTTP ao servidor no que aloxaremos o modelo para que nos devolva a predición.

O clúster configurouse para simular a arquitectura do sistema de monitorización da planta de flotación dun complexo mineiro tal e como foi presentada na in-trodución. Hai un produtor Kafka que simula a plataforma IoT que recopila os datos dos sensores. Esta información envíase en cru a un primeiro *topic* (*topic A* na figura). Por outra banda, hai outro produtor que simula o envío dos resultados das experimentacións de laboratorio sobre a pureza de extraccións pasadas (ao *topic B* da figura). O *topic C* é o resultado da unión das dúas colas anteriores, de xeito que a medición en cru enriquécese co último valor de pureza contrastado no laboratorio. As mensaxes do *topic D* da figura constrúense chamando á API de predición e concatenando o resultado devolto. Neste punto recupéranse tamén as cinco últimas medidas de pureza verificadas no laboratorio. O obxectivo é com-parar o valor predito para a extracción que se está a facer nese momento cos resultados que se obtiveron na planta durante as horas anteriores. A idea disto é poder establecer un nivel de alerta en función de se detectamos que o resultado da extracción que se está a producir dista moito dos últimos obtidos. Cómpre destacar que calquera dos *topic* mencionados pode ser consultado en tempo real por consumidores *Kafka*, polo que teremos a posibilidade de ler a información a todos os niveis, procesala e redirixila a calquera almacén de datos ou sistema de visualización.

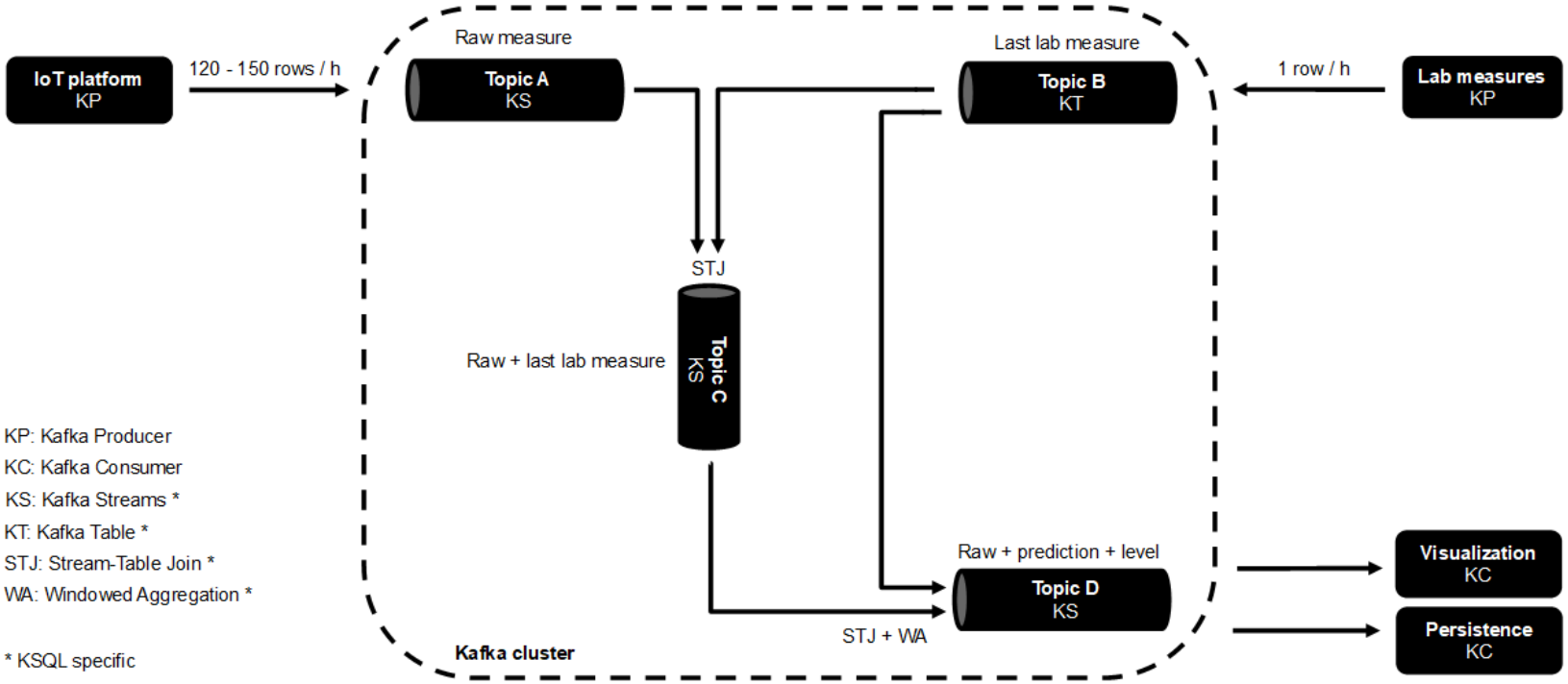

Figura 2.1: Clúster Kafka

Capítulo 3

Modelado e avaliación

O modelado é o proceso de aplicación das técnicas de minaría de datos sobre o conxunto de datos. Isto implica a selección da técnica ou algoritmo adecuados, o deseño dos mecanismos de adestramento e avaliación, a construción do modelo e a validación. Unha vez finalizada esta fase, debemos comprobar que o modelo cumpre cos requirimentos identificados, revisar o proceso e executar as accións correctivas necesarias.

O problema que tratamos de resolver consiste na predición dunha variable continua en base a unha serie de variables independentes, é dicir, trátase dun problema de regresión. Dentro do Machine Learning, coñécese como aprendizaxe supervisado a todas aquelas técnicas e algoritmos que nos permiten aproximar mediante inferencia unha función obxectivo en base a un certo número de observacións. Os algoritmos deben ter tamén a capacidade de xeralizar, de forma que se comporten de xeito correcto cando se atopen con novas observacións, e boa parte do traballo de modelado consiste en asegurar que o adestramento se fai de forma que permita que isto ocorra.

Nesta capítulo describiremos a plataforma de xestión dos modelos construída, faremos unha exploración dende un punto de vista matemático dos datos, describiremos o proceso de selección de variables e as transformacións do conxunto de datos realizadas, as principais técnicas avaliadas, as decisións técnicas tomadas en canto ao adestramento destes algoritmos e, por último, presentaremos os resultados obtidos.

3.1. Plataforma de xestión de modelos

Na figura 3.1 amósanse as principais compoñentes da plataforma para a xestión colaborativa do ciclo de vida dos modelos que temos construído. O elemento básico do sistema é un servidor MLflow, software de código aberto presentado pola empresa Databricks a mediados do ano 2018 que foi deseñado co obxectivo de centralizar nunha única plataforma todas as fases polas que pasa calquera desenvolvemento que implique traballar con técnicas de Machine Learning. Nestes proxectos é habitual utilizar ferramentas e librarías moi heteroxéneas e con orixes moi diversos. Isto fai que aspectos como a integración de código, o trazado das condicións nas que se lanzan os experimentos, a reproducibilidade dos mesmos ou a produtivización dos modelos sexan moi problemáticos. MLflow propón unha plataforma para Machine Learning aberta e centralizada, baseada en APIs REST e formatos de datos estándar cunha interface que facilita a integración con calquera *framework* existente.

MLflow susténtase sobre tres piares básicos que aproveitamos enteiramente para a nosa plataforma: *Tracking*, *Projects* e *Models*. Valéndonos dos dous primeiros elementos, temos preparado unha serie de paquetes que inclúen código Python e ficheiros de configuración para o lanzamento automatizado de experimentos parametrizados. Estes sete paquetes, un por cada un dos algoritmos presentados en 3.3, permiten que os científicos de datos poidan executar experimentos de regresión sobre calquera conxunto de datos sen ter que se preocupar de temas como a xestión das dependencias das librarías, das particularidades do adestramento mencionadas en 3.5 ou do almacenamento dos resultados. Mediante una interface de liña de comandos, só teñen que facilitar a ruta ao arquivo cos datos de adestramento, seleccionar un modelo e definir os valores a avaliar dos seus hiperparámetros. A versión do modelo que presente os mellores resultados de entre as probadas, xunto cunha serie de gráficas e métricas, será serializada e enviada automaticamente ao servidor de almacenamento. Nese punto, o modelo almacenado poderá ser explotado mediante unha API REST provista por MLflow *Models*.

Para esta parte temos decidido unha arquitectura baseada en tecnoloxías *cloud* en AWS en lugar dunha instalación *on-premise*. As razóns principais foron a necesidade de incluír mecanismos de autenticación, tanto a nivel de usuario, para acceder ao portal web e visualizar os resultados dos experimentos, como a nivel máquina, á hora de rexistrar execucións dende remoto, e a necesidade de escalar en función da demanda. Neste caso, o servizo de AWS *Elastic Beanstalk* permitiunos solventar ambos os dous problemas, levantando un servidor *Nginx* que configuramos para que actuase de *proxy* inverso con autenticación HTTP e de balanceador de carga (o servizo encárgase de xestionar o número de máquinas acesas en función da demanda). Os modelos adestrados e as gráficas almacénanse en S3, o servizo de almacenamento flexible de AWS.

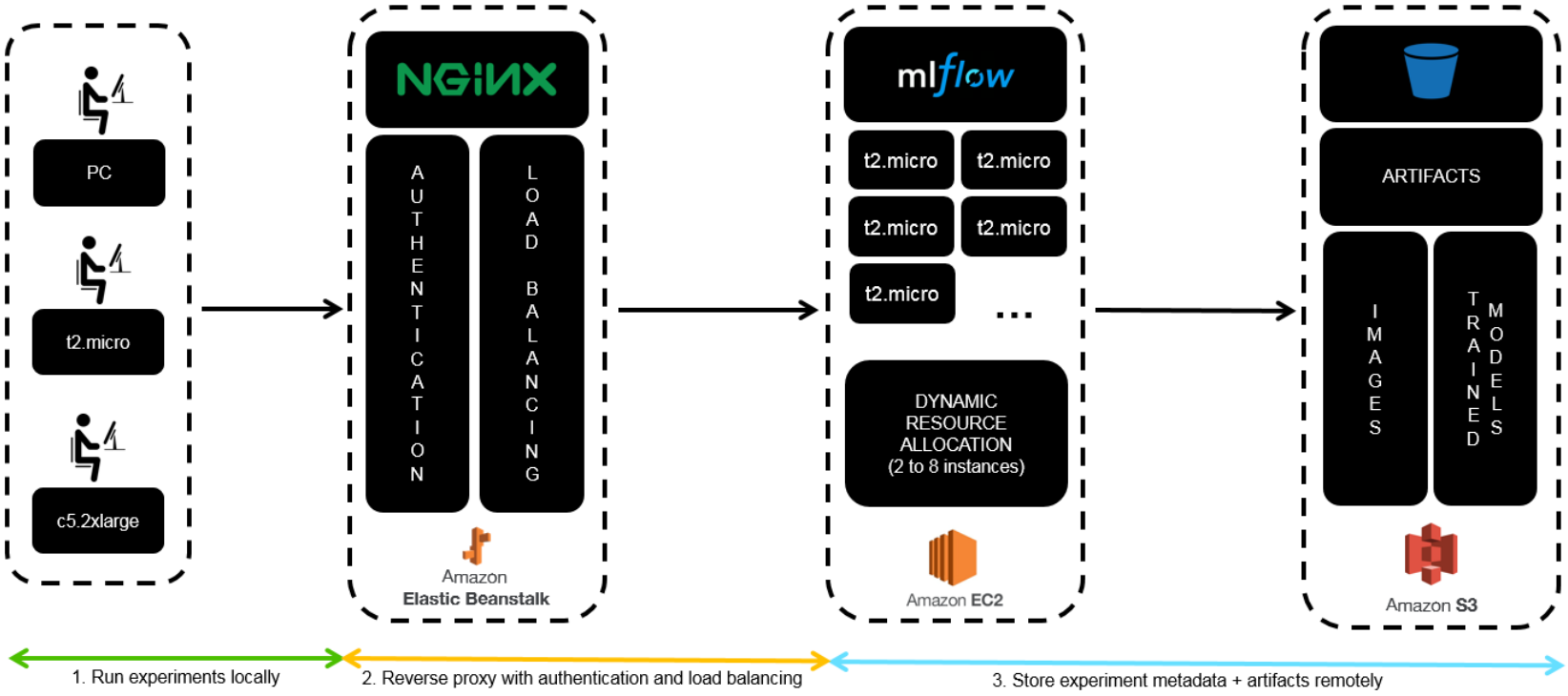

Figura 3.1: Servidor MLflow

3.2. Análise preliminar

3.2.1. Características

O conxunto de datos conta con máis de 700000 observacións (uns 168 *MB*) coas 24 columnas descritas anteriormente. A variable a predicir é **% Silica Concentrate** que toma valores entre 1'12 e 5'53. Non hai valores faltantes nin fóra de rango (ver cadro 3.1), polo que non se estima a necesidade de realizar ningún tipo de imputación. A distribución de valores das variables é moi similar á distribución normal na maioría de casos (ver exemplos na figura 3.2), agás nalgunhas das columnas relacionadas co fluxo de aire que entra nas cámaras. Ademais dos datos de partida, extraemos do campo coa data o día do mes, o día da semana e a hora por se foran feitos relevantes en canto ao rendemento da planta.

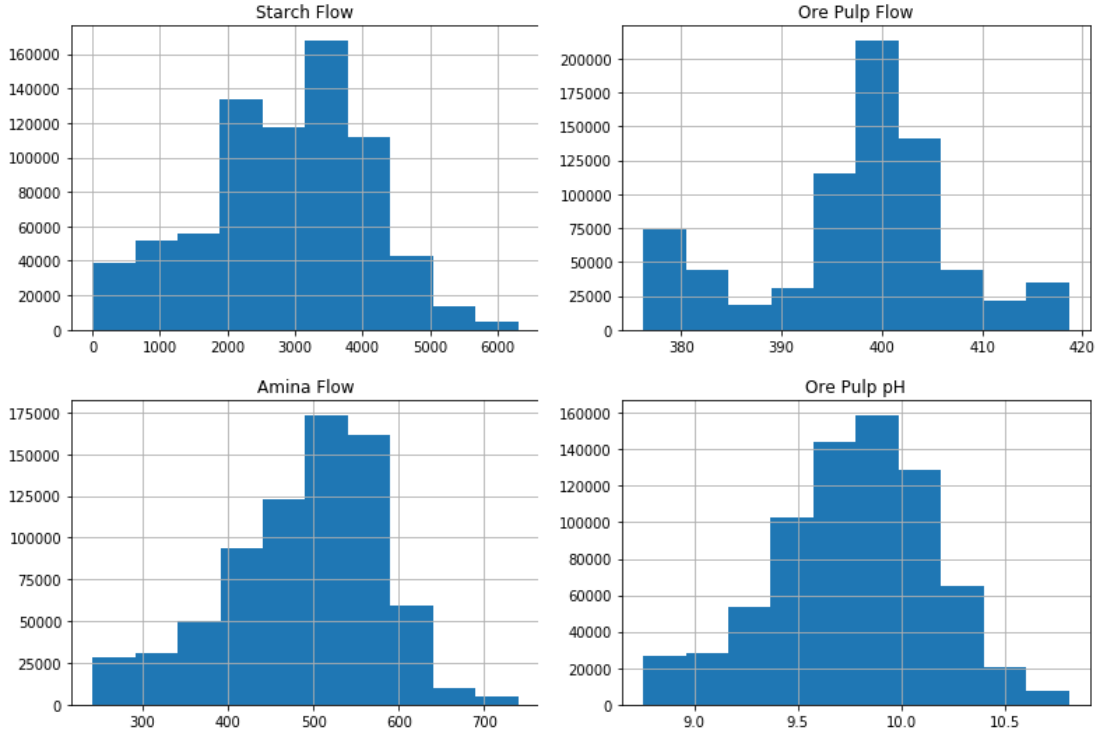

Figura 3.2: Histogramas dalgunhas das variables de entrada

3.2.2. Agrupamento dos datos

Unha das decisións máis relevantes a tomar á hora de traballar con este conxunto de datos foi a de decidir como resolver o problema das medicións de distintas variables a distintos intervalos que se comentou no apartado 2.1. Aínda que no

	count	mean	std	min	25%	50%	75%	max
% Iron Feed	736282.0	56.298307	5.160365	42.740000	52.670000	56.08000	59.72000	65.78000
% Silica Feed	736282.0	14.648984	6.810741	1.310000	8.940000	13.85000	19.60000	33.40000
Starch Flow	736282.0	2869.636615	1216.017896	0.002026	2075.070000	3020.23000	3728.93000	6300.23000
Amina Flow	736282.0	488.165523	91.254428	241.669000	431.835848	504.35250	553.33575	739.53800
Ore Pulp Flow	736282.0	397.570736	9.705444	376.249000	394.248000	399.23800	402.96700	418.64100
Ore Pulp pH	736282.0	9.767315	0.387176	8.753340	9.527050	9.79746	10.03780	10.80810
Ore Pulp Density	736282.0	1.680424	0.069206	1.519820	1.647390	1.69758	1.72838	1.85325
Flotation Column 01 Air Flow	736282.0	280.119813	29.633831	175.510000	250.278000	299.34100	300.14700	373.87100
Flotation Column 02 Air Flow	736282.0	277.121249	30.157126	175.156000	250.448000	296.20200	300.68600	375.99200
Flotation Column 03 Air Flow	736282.0	281.052538	28.571077	176.469000	250.847000	298.69000	300.38500	364.34600
Flotation Column 04 Air Flow	736282.0	299.446217	2.573805	292.195000	298.257000	299.80400	300.63500	305.87100
Flotation Column 05 Air Flow	736282.0	299.914815	3.637020	286.295000	298.069000	299.88600	301.78500	310.27000
Flotation Column 06 Air Flow	736282.0	292.065742	30.241329	189.928000	260.299750	299.48600	303.07700	370.91000
Flotation Column 07 Air Flow	736282.0	290.740507	28.690520	185.962000	256.047750	299.00200	301.90700	371.59300
Flotation Column 01 Level	736282.0	520.168402	131.085819	149.218000	416.902000	491.74900	594.09775	862.27400
Flotation Column 02 Level	736282.0	522.555279	128.216232	210.752000	441.835250	495.85350	595.29300	828.91900
Flotation Column 03 Level	736282.0	531.283790	150.931865	126.255000	411.256000	494.18400	601.29900	886.82200
Flotation Column 04 Level	736282.0	420.169753	91.755819	162.201000	356.627000	411.77050	485.28500	680.35900
Flotation Column 05 Level	736282.0	425.094530	84.479170	166.991000	357.595250	408.65000	484.00800	675.64400
Flotation Column 06 Level	736282.0	429.889517	89.919586	155.841000	358.438000	424.42700	492.77300	698.86100
Flotation Column 07 Level	736282.0	420.910258	84.899167	175.349000	356.705250	410.94000	476.14500	659.90200
% Iron Concentrate	736282.0	65.049096	1.118721	62.050000	64.370000	65.21000	65.86000	68.01000
% Silica Concentrate	736282.0	2.327270	1.125616	0.600000	1.440000	2.00000	3.01000	5.53000

Cadro 3.1: Estatísticos do conxunto de datos orixinal

conxunto de datos as medidas só aparecen etiquetadas a nivel día-hora, en moitos casos hai múltiples observacións por hora; cada vez que chega unha nova medición para algunha das variables xérase unha nova fila que mantén os valores das columnas que non cambiaron. Por exemplo, no caso da variable a predicir só se ten unha medida por hora, a que vén do laboratorio e se engade *a posteriori*, pero hai outras para as que se teñen ata 120 medicións por hora. Ter un número relativamente grande de observacións distintas para o mesmo valor de saída supón un problema importante de cara á aprendizaxe para calquera modelo de regresión, algo que temos comprobado para este caso específico nas probas realizadas.

Neste caso a solución adoptada foi a de xerar unha soa fila por hora (e, polo tanto, unha fila por valor de saída) elixindo a media aritmética como valor representativo para cada columna. Isto reduce enormemente o tamaño do conxunto de datos, que pasa a ter pouco máis de 4000 filas, a cambio de manter só a información verdadeiramente relevante para a realización da predición.

3.2.3. Tratamento da dimensionalidade

En primeiro lugar, trataremos de atopar e eliminar, en caso de que existan, as correlacións entre as variables do conxunto de datos. Polo xeral, as variables correlacionadas non axudan a mellorar a precisión dos algoritmos de aprendizaxe

supervisado e dificultan a inferencia a partir dos resultados. Ademais, tratar de reducir a dimensionalidade do conxunto de datos de adestramento, sen perder información de interese, é unha boa práctica que beneficia ao rendemento dos algoritmos e á súa interpretabilidade. O coeficiente de correlación de Pearson é unha medida da correlación lineal entre dúas variables cuantitativas tipificadas que se calcula como a covarianza que se dá entre elas:

$$\rho = \frac{cov(X, Y)}{\sigma_x \sigma_y}$$

Con este índice, que toma valores entre -1 e 1, podemos calcular unha matriz de correlacións para cada parella de variables do conxunto de datos (ver 3.3).

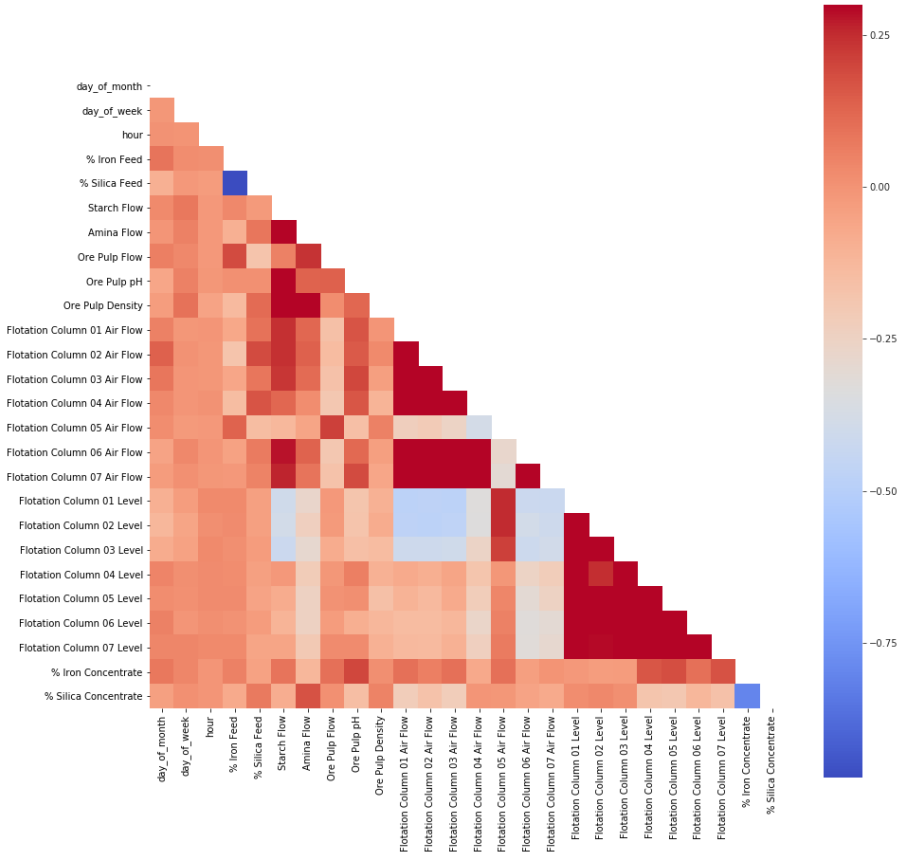

Figura 3.3: Matriz de correlacións inicial

As correlacións máis fortes aparecen nas parellas (‘ % Silica Feed’, ‘ % Iron Feed’) e (‘ % Silica Concentrate’, ‘ % Iron Concentrate’). Estas variables refírense ao grao de pureza dos compostos de entrada e de saída, polo que é normal que exista correlación entre elas (canto máis ferro, menos sílice e de aí a dependencia lineal

negativa que se amosa en azul). No caso da segunda parella, as dúas son medicións de laboratorio que non están dispoñibles en tempo real, polo que eliminamos o segundo elemento (o primeiro é a variable obxectivo). No caso da primeira parella eliminamos a columna que se refire á porcentaxe de sílice. A matriz de correlación recalculada sen estas variables amósase na figura 3.4.

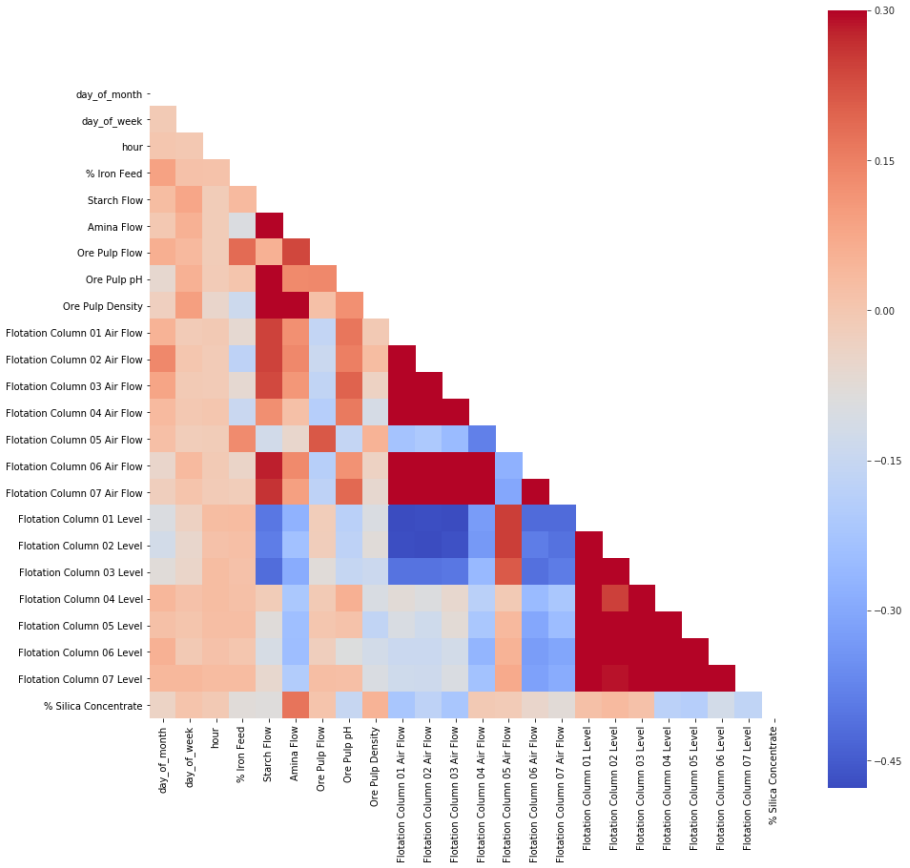

Figura 3.4: Matriz de correlacións final

Nesta nova matriz os valores de correlación son relativamente pequenos, entre -0.3 e 0.3. As variables menos correlacionadas co resto son as que se extraen da variable que contén a data e a hora á que se produce a medición, algo que é de esperar se supoñemos que a planta funciona a un ritmo constante todos os días da semana. En xeral, son variables que non deberían perxudicar ao rendemento dos algoritmos e que poderían aportar certa información en combinación con outras. O resto de valores relativamente altos concéntrase nas variables que se refiren ao estado das distintas cámaras de flotación en canto ao fluxo de aire entrante e ao nivel de escuma na superficie. Parece lóxico pensar que nun instante dado haxa certa correlación entre o estado de todas as cámaras. De novo, estas correlacións non son fortes e non deberían ser perxudiciais para o rendemento nin

a interpretabilidade dos algoritmos, polo que os posibles beneficios de eliminalas non superan o risco de estar a omitir información de relevancia para a predición.

Co obxectivo de simular unha situación máis realista, disporemos do último resultado de laboratorio cun retardo de dúas horas, algo que tamén debería axudar a mellorar a predición. Engadimos ao conxunto de datos ordenado temporalmente a columna 'sc_lag2', co valor de ' % Silica Concentrate' da fila situada dúas posicións máis arriba. Na figura 3.2 inclúense os estatísticos para conxunto de datos definitivo co que entrenaremos os modelos.

	count	mean	std	min	25%	50%	75%	max
day_of_month	4097.0	15.330730	8.932539	1.000000	8.000000	15.000000	23.000000	31.000000
day_of_week	4097.0	3.014401	1.992304	0.000000	1.000000	3.000000	5.000000	6.000000
hour	4097.0	11.507200	6.925349	0.000000	5.000000	12.000000	18.000000	23.000000
% Iron Feed	4097.0	56.294730	5.158347	42.740000	52.670000	56.080000	59.720000	65.780000
Starch Flow	4097.0	2868.915033	950.895934	54.595483	2168.968993	2908.578878	3528.727412	6270.158798
Amina Flow	4097.0	488.160636	83.694790	242.927477	436.037967	502.454283	549.522256	736.982378
Ore Pulp Flow	4097.0	397.577005	8.370186	376.837604	398.848778	399.841806	400.584861	418.070232
Ore Pulp pH	4097.0	9.767704	0.378065	8.753389	9.540878	9.795850	10.030779	10.807370
Ore Pulp Density	4097.0	1.680399	0.063759	1.519926	1.651352	1.695735	1.721790	1.832430
Flotation Column 01 Air Flow	4097.0	280.151387	29.409908	175.885579	250.089767	299.837839	299.951350	312.295415
Flotation Column 02 Air Flow	4097.0	277.158112	29.421896	178.188430	250.096872	299.526394	299.979900	309.887767
Flotation Column 03 Air Flow	4097.0	281.082141	28.373541	177.202665	250.087672	299.888089	299.946594	302.783000
Flotation Column 04 Air Flow	4097.0	299.447740	2.426897	293.336669	299.732178	299.910922	299.981800	305.596693
Flotation Column 05 Air Flow	4097.0	299.917491	3.332252	287.056447	299.688506	299.923778	300.137694	307.009708
Flotation Column 06 Air Flow	4097.0	292.073489	29.599172	196.512082	268.679419	299.870528	300.095878	354.979117
Flotation Column 07 Air Flow	4097.0	290.754108	27.219859	199.728122	283.155469	299.861356	300.112233	351.268656
Flotation Column 01 Level	4097.0	520.248979	122.178216	181.925623	416.468100	499.619556	599.712939	859.025062
Flotation Column 02 Level	4097.0	522.650507	116.049815	224.909663	449.249644	499.822467	599.330539	827.775874
Flotation Column 03 Level	4097.0	531.356198	138.586669	135.214506	405.366394	499.593706	600.215106	884.840698
Flotation Column 04 Level	4097.0	420.325679	76.590375	165.725058	351.490017	401.272994	496.273322	675.631942
Flotation Column 05 Level	4097.0	425.255386	75.034743	214.735894	350.979578	401.127850	497.775572	674.068176
Flotation Column 06 Level	4097.0	429.939080	75.507132	203.702926	354.128200	407.549394	497.878917	698.505832
Flotation Column 07 Level	4097.0	421.018314	72.556415	185.061388	350.939278	400.991200	462.283886	655.501603
sc_lag2	4097.0	2.326994	1.124727	0.600000	1.440000	2.000000	3.010000	5.530000
% Silica Concentrate	4097.0	2.326754	1.124783	0.600000	1.440000	2.000000	3.010000	5.530000

Cadro 3.2: Estatísticos do conxunto de datos definitivo

3.2.4. Dependencias temporais

Dada a natureza do problema, e xa que decidimos incluír nas medicións datos sobre o rendemento pasado, é de interese estudar se existen dependencias temporais na variable obxectivo. A simple vista (ver parte superior da figura 3.6) non é sinxelo apreciar tendencias ou patróns. Porén, podemos apoiarnos en ferramentas coma os *lag plot* (ver figura 3.5), diagramas de dispersión que enfrontan valores consecutivos dunha serie.

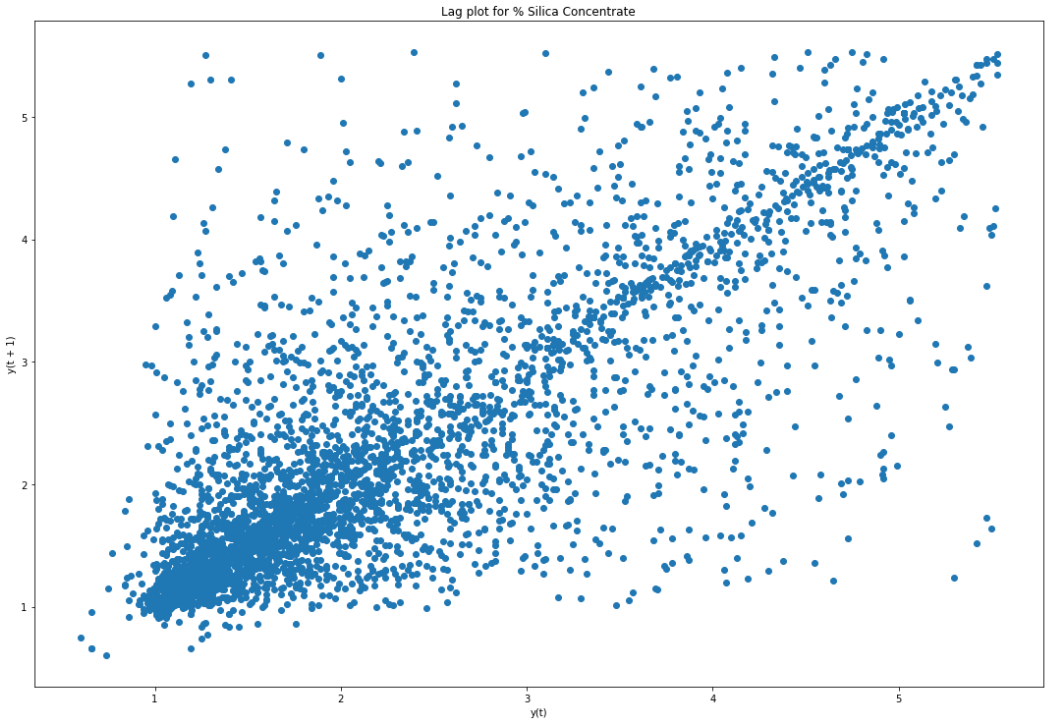

Figura 3.5: Lag plot de ‘ % Silica Concentrate’

Neste tipo de representacións, nas que poderiamos xogar co parámetro de retardo, o comportamento esperado é que os puntos non formen ningún tipo de estrutura se a variable é completamente aleatoria. No caso da variable estudada, o diagrama de dispersión mostra unha concentración de puntos na diagonal que marca a recta $y{=}x$, sinal de que hai certa dependencia entre o valor actual e anterior da variable. Por tanto, é de esperar que a columna engadida (**‘sc_lag2’**), que ten unha relación de dependencia relativamente forte coa variable obxectivo, axude a mellorar a calidade das predicións.

Tamén é de interese estudar o grao de estacionalidade da variable. Dise que unha variable é estacionaria se a súa distribución de probabilidade se mantén estable ao longo do tempo, e polo tanto, a media e a varianza mantéñense constantes. A proba de Dickey-Fuller é un test estatístico baseado nun contraste de hipóteses que serve para determinar se unha serie temporal é estacionaria. A hipótese nula é que existe una raíz unitaria, é dicir, se unha raíz da ecuación característica da función que define a serie é 1. Nese caso pódese dicir que a serie é non estacionaria (aínda que poida haber ou non tendencias). A hipótese alternativa é que a serie non presenta estacionalidade. Na parte superior da figura 3.6 amósase o resultado do test para a serie estudada. Como o *p-valor=0.000*, é inferior que o nivel de significación *alpha=0.05*, debemos rexeitar a hipótese nula e aceptar que a serie é estacionaria.

Nos cadros inferiores da figura 3.6, amósanse os correlogramas ACF (Auto Correlation Function) e PACF (Partial Auto Correlation Function). ACF é unha representación visual do coeficiente de correlación do valor dunha variable respecto aos seus n predecesores. ACF só ten en conta o valor inmediatamente anterior, mentras que PACF mostra a porción de correlación entre dous valores contiguos eliminando correlacións anteriores; habitualmente, son gráficas que serven para atopar patróns. No noso caso, a caída abrupta de valores na gráfica do PACF e que a maioría dos puntos se manteñan na banda do intervalo de confianza é un feito que apoia a afirmación de que a serie é estacionaria.

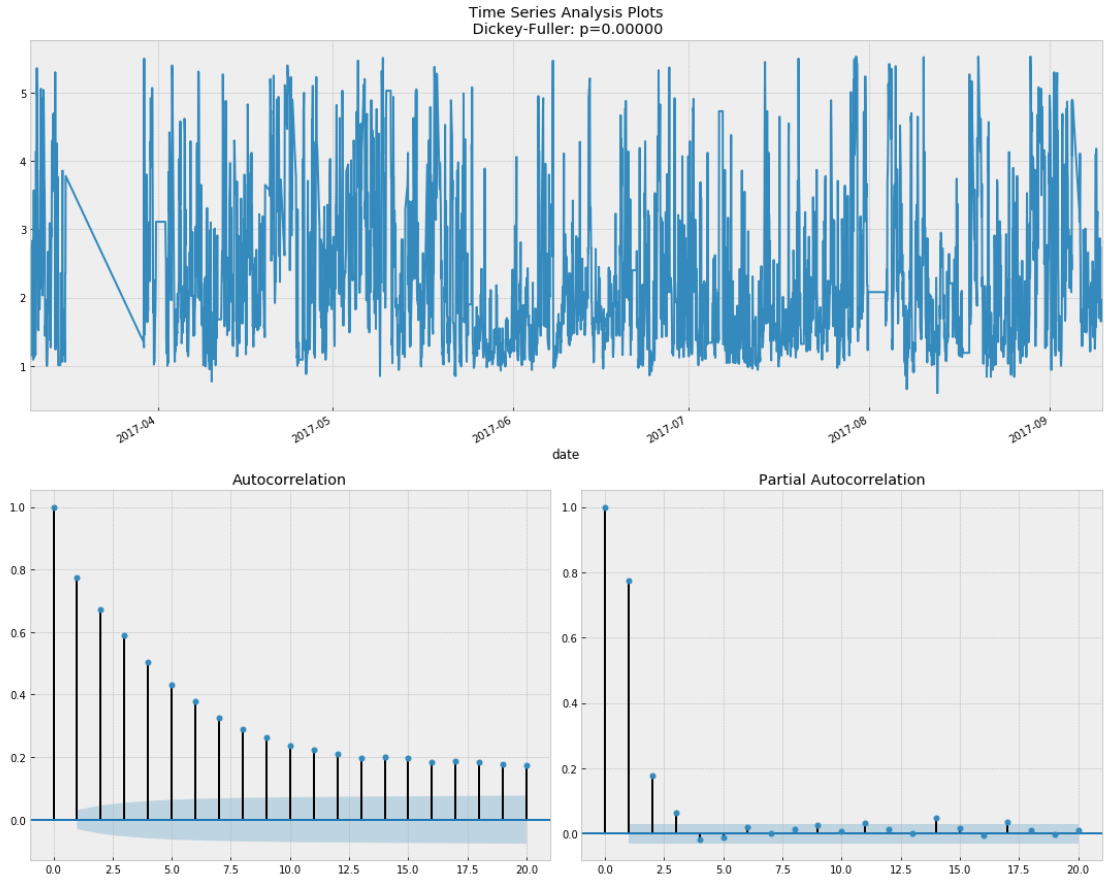

Figura 3.6: Test Dickey-Fuller, ACF e PACF

Cómpre estudar se algún método específico de tratamento de series temporais podería funcionar mellor que a aproximación con algoritmos de aprendizaxe supervisado. Temos demostrado que a serie é estacionaria; non obstante, non temos apreciado ningún tipo de tendencia ou ciclo estable que debera poder ser aproveitado por estes algoritmos. De feito, a distribución recorda en certa medida ao ruído branco. Polo tanto, aínda que sería interesante facer a comparativa, non nos apoiaremos en algoritmos específicos de predición para series temporais.

3.3. Algoritmos avaliados

Co obxectivo de atopar o mellor modelo de regresión para o conxunto de datos dado, temos probado e preparado para a experimentación as implementacións de *scikit-learn* e do paquete *xgboost* de Python dos seguintes algoritmos:

- **CART**: As árbores CART (Classification And Regression Trees) son unha das técnicas de ML máis antigas e simples. O algoritmo baséase na construción de árbores de decisión alternando particións do espazo de entrada mediante estratexias voraces e fases de poda. Aínda que de por si non é o método máis preciso, o seu resultado é moi fácil de interpretar visualmente e é a base doutros métodos que si son moi potentes. Faremos selección de hiperparámetros para *min_samples_split* (número mínimo de mostras nunha rama para considerar unha posible división), *min_samples_leaf* (número mínimo de mostras nunha rama para considerala chea) e *max_depth* (altura máxima permitida).
- **Lasso**: O método Lasso (Least Absolute Shrinkage and Selection Operator) parte do método regresión lineal e engade selección de variables e regularización para tratar de mellorar a predición e a interpretabilidade do resultado. Faremos selección de hiperparámetros para *alpha*, o parámetro que determina o nivel de regularización do modelo.
- **SVR con *kernel* lineal**: As máquinas de vectores de soporte con *kernel* lineal son un conxunto de algoritmos para clasificación e regresión que se basean na división do espazo mostral mediante hiperplanos da dimensión do conxunto de datos. Optimizaremos o parámetro C, que representa a penalización por exemplos mal clasificados dada unha posible división.
- **SVR con *kernel* RBF**: Máquina de soporte vectorial con *kernel* RBF, modelo non paramétrico que, no canto de establecer un hiperplano único que divide o espazo mostral, utiliza unha función de base radial. Estudaremos o efecto de C e *gamma*.
- **Redes neuronais**: Implementación dunha rede neuronal densamente conectada. Estudaremos o efecto do número de capas (*n_layers*) e do número de neuronas por capa (*layer_size*).
- **Random Forest**: Combinación ou *ensemble* de árbores simples que reciben e son adestrados con distintos subconxuntos da entrada (*bagging*). A idea é a de promediar moitos modelos ruidosos pero imparciais, reducindo a variación. O hiperparámetro principal que estudaremos será o número de árbores no *ensemble*, *n_estimators*.
- **Extreme Gradient Boosting**: Os métodos de *boosting* baséanse na construción de modelos de xeito escalonado, neste caso a partir de árbores

de decisión sinxelas, potenciándoas en base á optimización dunha función de perda diferenciable. Neste caso avaliaremos *learning_rate*, *max_depth* e *n_estimators*.

3.4. Partición dos datos

Co obxectivo de demostrar a utilidade do modelo funcionando sobre a arquitectura que deseñemos, e como paso previo ao adestramento, debemos separar unha parte dos datos. Isto é porque na situación real dispoñeriamos dun histórico sobre o que adestrariamos modelos aplicando selección de variables e a agrupación comentada en 3.2.2. Non obstante, á hora despregar este modelo sobre a arquitectura non tería sentido que lle volvésemos a enviar os mesmos exemplos cos que tería adestrado. Para evitar isto, separamos os últimos 10MB do conxunto de datos orixinal ordeados temporalmente e mantémolos fóra da fase de adestramento e validación.

3.5. Proceso de adestramento

O proceso de adestramento pasa polas seguintes fases independentemente do modelo elixido:

- **División do conxunto de datos**: Realizamos unha división 80-20 (80 % dos datos para adestramento e 20 % para avaliación) con barallado sobre a primeira parte do conxunto de datos xa agrupados e feita a selección de variables (ver 3.2.2 en adiante).
- **Escalado**: O escalado é un paso fundamental para o bo funcionamento de boa parte dos algoritmos que estamos a avaliar, especialmente, todos aqueles que se basean no cálculo de distancias entre observacións que se poden ver moi desviadas en función das unidades e da escala de cada variable. Neste caso utilizamos unha normalización de cada variable en base á media e á desviación típica. É importante ter en conta que a normalización debe axustarse sobre os datos de proba e despois debe aplicarse cos mesmos valores de media e desviación típica sobre o conxunto de test. Neste caso tamén escalamos a variable obxectivo, de novo, axustada en adestramento e reaplicada en test.
- **Criterio**: Adestraremos os modelos para que tenten minimizar o erro cadrático medio (Mean Squared Error, MSE). O MSE mide o promedio dos erros

ao cadrado que se producen entre a predición, $\hat{Y}$, e a variable obxectivo, Y:

$$MSE = \frac{1}{n}\Sigma_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

- **Selección de hiperparámetros**: En canto á selección de hiperparámetors, o código está preparado para facer validación cruzada con k=5. A validación cruzada é unha técnica de avaliación que consiste na división do conxunto de proba en k grupos sobre os que se elixen *k-1* para adestramento e un para validación de xeito alterno. Tómase o promedio dos erros dos k resultados como valor representativo, o que fai máis robusta a medición e permite dar rangos de erro. É unha técnica que funciona especialmente ben cando o conxunto de adestramento non é moi extenso. A combinación de hiperparámetros que ofreza un menor erro de validación será coa que adestraremos o modelo definitivo.
- **Avaliación**: Unha vez realizado o adestramento e achada a mellor combinación de hiperparámetros, volvemos adestrar con cada combinación e probamos cada un destes modelos co conxunto de proba. O obxectivo deste proceso é o de estudar a evolución de ámbolos erros, adestramento e test, a medida que o modelo se vai facendo máis complexo e comprobar se as combinacións que mellor funcionaron durante o adestramento ofrecen un rendemento aceptable sobre o conxunto de test (ver 3.7). As métricas que calcularemos, ademais do MSE, serán o RMSE (Reduced Mean Squared Error) e MAE (Mean Absolute Error), ambas nas unidades da variable de saída diferenciándose en que o RMSE penaliza máis os erros grandes, e o coeficiente de determinación, R^2.
- **Adestramento do modelo final**: Unha vez feita a selección de hiperparámetros co conxunto de adestramento, podemos xuntar de novo todos os datos e adestrar o modelo con esta combinación única de parámetros sobre eles. A lóxica é que os datos de test xa non poden interferir na selección de hiperparámetros porque a decisión xa está tomada e que o poder preditivo e de xeralización do modelo debería mellorar sendo adestrado cunha maior cantidade de datos, e, por tanto, máis variabilidade. Este modelo xa adestrado e cos parámetros de normalización incorporados é sobre o que se poderán facer predicións unha vez incorporado á arquitectura.
- **Representación visual dos resultados**: Incluímos a gráfica de validación cruzada para o parámetro que máis interese teña segundo o modelo (ver 3.7) e o número de casos seleccionado polo usuario. No caso dos métodos baseados en árbores de decisión (CART, Random Forest e Extreme Gradient Boosting) tamén achegaremos gráficas do peso de cada variable no modelo e da árbore de decisión resultante. No caso de CART, a árbore

de decisión e o peso dos preditores veñen dado de xeito natural. Cos métodos de *ensemble* de árbores o que se fai é unha estimación destes valores a partir do resultado final.

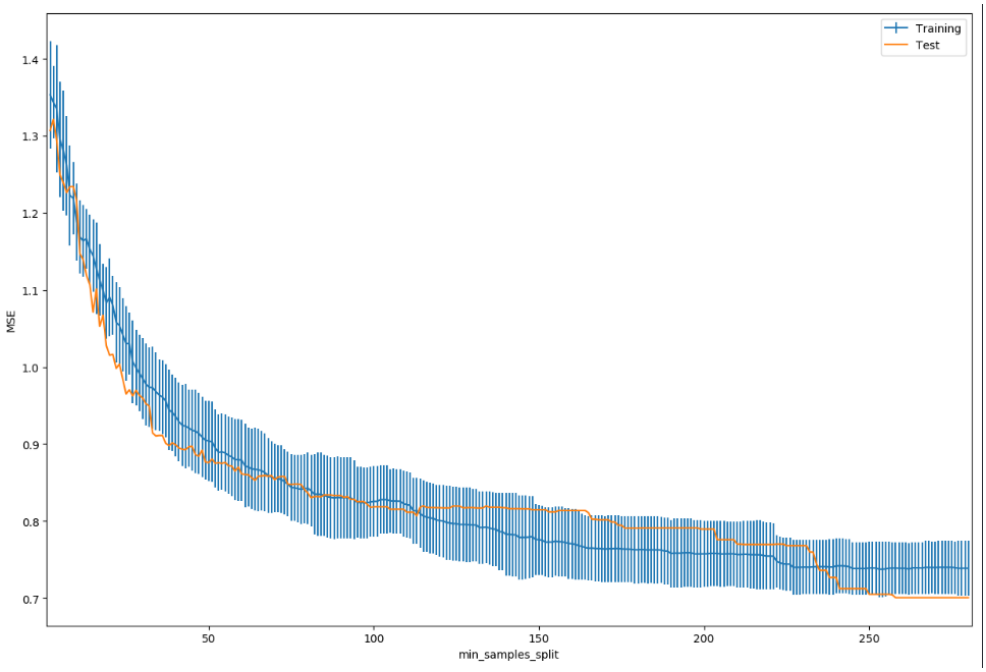

Figura 3.7: Evolución dos erros de adestramento e avaliación para distintos valores de *min_samples_split* en CART

3.6. Comparativa de rendemento dos modelos

Cos modelos parametrizados deste xeito obtivemos os seguintes resultados:

Algoritmo	Mellor combinación de hiperparámetros
cart	*max_depth*: None, *min_samples_leaf*: 1, *min_samples_split*: 299
lasso	*alpha*: 0.01
linear_svr	C: 0.01
rbf_svr	C: 1.0, *gamma*: 0.1
nnet	*n_layers*: 1, *layer_size*: 1
random_forest	*n_estimators*: 104
xgboost	*learning_rate*: 0.05, *max_depth*: 6, *n_estimators*: 115

Cadro 3.3: Combinación de hiperparámetros co mellor resultado por modelo

Algoritmo	Training MSE	MAE	RMSE	R^2
cart	0.732	0.609	0.830	0.468
lasso	0.687	0.591	0.815	0.488
linear_svr	0.719	0.570	0.833	0.465
rbf_svr	0.661	0.556	0.804	0.500
nnet	0.689	0.593	0.816	0.486
random_forest	0.671	0.586	0.800	0.506
xgboost	0.669	0.589	0.808	0.496

Cadro 3.4: Métricas resultado por modelo

Os tres métodos que obteñen os valores de R^2 máis altos, próximo a 0.5, son *rbf_svr*, *random_forest* e *xgboost*; a priori, os algoritmos máis potentes dos probados. Xa que o rendemento é moi semellante entre os tres, para a simulación teriamos a opción de lanzar a predición contra os tres modelos e devolver a media ou a mediana como resultado. A lóxica detrás desta decisión sería que uns algoritmos adáptanse mellor ca outros a certas casuísticas, e combinando as predicións poderiamos lograr que os erros se compensasen.

Capítulo 4

Operacionalización

O proceso de operacionalización dun modelo de Machine Learning inclúe todas aquelas tarefas necesarias para que o modelo xa entrenado poida ser despregado nalgún tipo de contedor que permita que poida ser utilizado polas aplicacións de negocio no contexto dun problema de regresión ou clasificación. O ideal é que a infraestrutura de adestramento e de operacionalización sexan independentes. Isto é porque non serven aos mesmos propósitos: os clústers de adestramento deberían dimensionarse en función da cantidade de datos e da complexidade dos modelos que se estean a probar mentres que as máquinas encargadas de servir os modelos deberían ser dimensionadas en función do número de peticións concorrentes e dos tempos de resposta esixidos. A arquitectura global deseñada e presentada neste capítulo trata de reflictir este feito.

Consideramos tamén importante amosar como encaixaría a parte do modelado e produtivización de modelos dentro dunha arquitectura de fluxo de información realista como a deseñada con Kafka. Polo tanto, neste capítulo explicaremos tanto o proceso para adestrar e operacionalizar modelos como os pasos seguidos para conseguir integrar estes dous sistemas dentro do *pipeline*.

4.1. Plataforma completa

En 4.1 inclúese un diagrama da arquitectura global do sistema. Na situación plantexada, hai un ou varios equipos de científicos de datos traballando en paralelo no modelado do problema. Estes usuarios, ben dende os seus equipos ou utilizando servizos de computación na nube con máquinas adicadas para os adestramentos máis computacionalmente esixentes, lanzan experimentos de xeito concorrente contra o servidor MLflow en Elastic Beanstalk. Este servizo xestiona de xeito transparente os mecanismos de autenticación, a instanciación dinámica de nodos e a persistencia. O proceso de *setup* para os usuarios consiste tan só en descargar unha copia do repositorio do proxecto nunha máquina Linux e configurar unha serie de variables do *shell* coas súas credenciais de acceso ao servidor. Utilizando a interface de liña de comandos, deben seleccionar o modelo e os hiperámetros a probar de xeito individual ou en conxuntos. O sistema encárgase de descargar as dependencias necesarias, de rexistrar metainformación sobre o experimento e de xestionar os resultados, sempre dando *feedback* visual ao usuario.

Unha vez se obteñen un ou varios modelos cun rendemento aceptable, elíxese unha máquina adecuada para despregalos. O proceso é semellante ao do caso anterior: descargar o repositorio e utilizar a interface de línea de comandos para seleccionar o modelo a produtivizar (a través do seu *Run ID*) e para especificar a configuración do servidor. O modelo pasará a ser accesible a través dunha API REST aloxada nun servidor Flask. Esta aproximación permítenos ter en escoita varios modelo dentro incluso da mesma máquina por se quixésemos facer unha predición híbrida sen empregar máis recursos.

No clúster Kafka (ver 2.2) , os datos irán chegando aos *topic* de entrada a medida que se vaian rexistrando novas lecturas. Como paso previo, temos engadido ás funcións de KSQL unha UDF escrita en Java que se encarga de recoller as mensaxes coa última medición de laboratorio engadida (*topic C* na figura 2.1), serializalas a JSON coa estrutura adecuada e envialas mediante unha petición HTTP ao servizo en escoita que contén o modelo. Tamén temos definido as transformacións que ocorren entre as colas mediante consultas KSQL facendo uso da UDF comentada. Deste xeito, obteremos unha predición por cada novo evento.

En canto á saída, calquera das colas pode ser consumida nun instante dado. Actualmente Kafka conta con múltiples conectores que permiten a integración cómoda coas bases de datos e ferramentas de visualización máis comúns. No caso da visualización en tempo real, parécenos interesante a integración con sistemas de *dashboarding* en tempo real como Grafana, Prometheus ou Netdata utilizando software intermedio como Graphite, influxDB ou Elasticsearch.

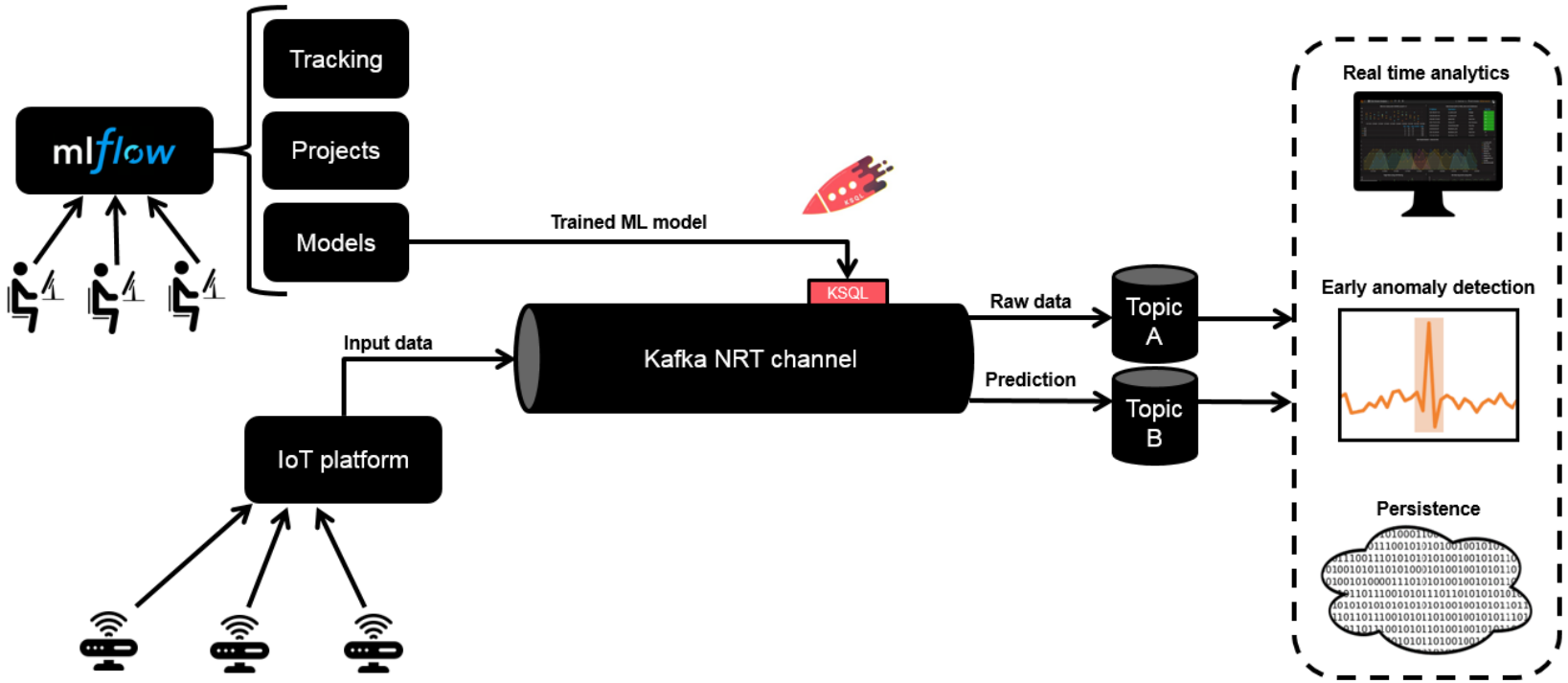

Figura 4.1: Visión global da arquitectura

Capítulo 5

Conclusións

Durante o transcurso do proxecto temos desenvolvido unha plataforma para o modelado colaborativo e áxil sobre MLflow, analizado e proposto unha solución para o problema da predición de calidade en tempo real durante o proceso de flotación de ferro proposto e elaborado unha simulación do *pipeline* dunha planta mineira que xestiona o fluxo de información e as transformacións en tempo real mediante Kafka e KSQL. Ademais, temos comprobado a flexibilidade e as facilidades que nos proporcionan as tecnoloxías de computación na nube, en concreto Amazon AWS, para despregar este tipo de solucións.

Un dos obxectivos do proxecto era avaliar a madurez das tecnoloxías seleccionadas. Aínda estando en fase de proba durante o desenvolvemento do proxecto (a versión utilizada foi a 0.9.1), cremos que MLflow é unha ferramenta con moito potencial. A día de hoxe ofrece aos equipos de *data scientists* capacidades como a reproducibilidade e a centralización dos resultados das experimentacións cun proceso de adaptación moi simple, o cal permite a integración con calquera das linguaxes e *frameworks* habituais no campo. Dende o punto de vista operativo, permite ter catalogados os experimentos xunto coa documentación e cos modelos resultantes e permite ademais despregar estes modelos de xeito áxil, facilitando as predicións en tempo real e o mantemento e a verificación da calidade dos modelos (tests A/B, comprobacións de degradación do rendemento...). Á espera dos cambios que se introduzan con versións máis estables (autenticación, soporte para *cross-validation*...), ten cumprido de sobra coas nosas expectativas.

Doutra banda, Kafka é unha tecnoloxía cada vez máis consolidada que ofrece resultados moi notorios en casos de usos diversos: ETLs, arquitecturas de microservizos, *pipelines* NRT... Con KSQL, Confluent permítenos axilizar enormemente o proceso de desenvolvemento, facilitando a creación de topoloxías complexas. É un ferramenta sinxela de configurar, flexible e potente que ten resultado moi satisfactoria para o desenvolvemento do proxecto.

5.1. Posibles melloras e ampliacións

Temos identificado as seguintes melloras e posibles ampliacións futuras:

- **Melloras na plataforma de modelado**: O código que temos implementado permite que os usuarios poidan lanzar adestramentos para facer validación cruzada de hiperparámetros sobre modelos de regresión de xeito áxil. Non obstante, require que se traballe en modo texto por medio dunha interface de liña de comandos. Ademais, tanto a execución dos adestramentos, en máquinas locais ou na nube, coma o proceso de despregar e servir os modelos, require dunha intervención manual mínima (instanciar a máquina, descargar o repositorio e utilizar a CLI). Unha versión máis avanzada da plataforma debería contar cun servizo que permitise pasar da proposta actual, próxima ao IaaS, a unha solución máis próxima ao PaaS. Isto é, deberiamos facer transparente os detalles da instanciación e configuración das máquinas de adestramento e produtivización. No caso ideal, os usuarios utilizarían unha aplicación que lles permitiría elixir de xeito visual o conxunto de datos (selección de variables, transformacións...), os modelos e hiperparámetros a probar e o tipo e cantidade de máquinas sobre as que executar o adestramento. O software encargaríase de reservar e instanciar os recursos necesarios, instalar e configurar as dependencias e executar a instrución dada informando en todo momento ao usuario das súas accións e de posibles erros.
- **Modelos de clasificación**: Dado a natureza do problema estudado durante o proxecto, a nosa prioridade foi a construción de código que permitise o adestramento con modelos de regresión. En versións futuras o obxectivo non debería ser só ampliar o soporte para outros modelos de regresión máis avanzados senón construír paquetes de código análogos aos que temos xerado para permitir adestrar modelos de clasificación.
- **Monitorización avanzada**: En canto á plataforma Kafka, sería interesante implementar fluxos de datos que nos permitisen unha monitorización en tempo real do estado da planta, non só en canto á predición de calidade; por exemplo, controlar en tempo real as diferenzas entre as cámaras podería servir para detectar posibles avarías. Con ideas coma esta poderiamos enriquecer os cadros de mando e xerar alertas cunha lóxica máis complexa.
- **Verificación de calidade dos modelos**: Un problema que se podería dar é o da degradación progresiva do rendemento do modelo en produción co paso do tempo. Isto é un problema habitual debido á natureza complexa e cambiante das situacións nas que se aplican os modelos. Sería desexable implementar algún mecanismo de control da precisión que se execute regularmente sobre o modelo e que poida comportar un readestramento.

Bibliografía

[1] NARKHEDE, Neha, SHAPIRA, Gwen, PALINO, Todd. *Kafka The Definitive Guide. RealTime Data and Stream Processing at Scale*. O'Reilly, USA 2017. ISBN 978-1-491-99065-0.

[2] BYZEK, Yeva. *Stream Processing Made Easy with Confluent Cloud and KSQL. A deployment guide: design, configuration and management*. Confluent, 2018.

[3] FREYTAG J. et al. *Streams and Tables: Two Sides of the Same Coin*. BIRTE '18, Rio de Janeiro, Brazil. Agosto de 2018. Dispoñible en `https://www.confluent.io/blog/streams-tables-two-sides-same-coin`

[4] FRIEDMAN, Jerome, HASTIE, Trevor, TIBSHIRANI, Robert. *The Elements of Statistical Learning. Data Mining, Inference and Prediction*, 2ed. Springer, 2009. ISBN 978-0-387- 84858-7.