POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

STUDIO E PROGETTAZIONE DI ALGORITMI DI ANALISI DEI DATI IN AMBITO INTERNET OF THINGS

Relatore:
prof. Elena Baralis

Candidato:
Luca Di Maria

Settembre 2018

Ringraziamenti

Vorrei esprimere la mia gratitudine a tutti coloro che mi hanno sostenuto e aiutato durante il periodo di stesura della tesi, per i loro preziosi suggerimenti, critiche e osservazioni.

Ringrazio innanzitutto il mio relatore, la prof.ssa Elena Baralis, e i suoi assistenti, Daniele Apiletti e Luca Venturini, per il loro aiuto e supporto nella stesura di questo lavoro, e per i preziosi consigli e conoscenze che hanno condiviso con me.

Un ringraziamento particolare va ai miei amici e colleghi che mi hanno incoraggiato e con i quali ho condiviso gioie, sacrifici e successi.

Vorrei infine ringraziare la mia famigli.a e la mia fidanzata, che mi hanno sempre sostenuto, sia moralmente che economicamente, e senza i quali non avrei potuto raggiungere questo traguardo, a loro dedico tutto il mio lavoro.

Sommario

Oggigiorno la quantità di dati che circola su internet sta aumentando sempre di più, ed in particolare con l'avvento e la propagazione dell'Internet of Things si sta incrementando in maniera esponenziale. Connettere un oggetto a internet consente di poter monitorare lo stesso, e di comandarlo a distanza. Discorso valido anche per le caldaie, che producono dati grazie ai loro sensori e li inviano ai gateway. Tali dati possono essere analizzati per estrapolare informazioni sul ciclo di vita delle caldaie.
Scopo del lavoro di tesi è definire e progettare le tecniche di analisi dei dati per la creazione di un modello predittivo dello stato di salute della caldaia (predictive maintainance), a partire dai dati raccolti durante il funzionamento della stessa.
Volendo scalare nella gestione di milioni di caldaie, si è affrontata la progettazione di un'architettura Big Data per l'analisi dei dati.
Si è effettuata un'analisi esplorativa, che comprende una fase di pulizia dei dati e verifica dell'effettiva presenza di situazioni anomale negli stessi.
Per raggiungere l'obiettivo si sono svolte anche fasi di raccolta di conoscenza da esperti di dominio, modellazione dei fenomeni di interesse a partire dai dati sperimentali e creazione di dataset sintetici per il test della soluzione proposta.
Per processare i dati si sono applicate tecniche di machine learning basate su classificatori multi-class e one-class. Infine sono state svolte analisi, per il classificatore one-class, sulla probabilità che un campione nel dataset sia segnalato come un'anomalia, e su tutte le anomalie identificate, distinguendole in veri e falsi positivi. Si è infine arrivati ai risultati e ai limiti di questi approcci, e sono state fatte ipotesi sui possibili proseguimenti del lavoro.

Indice

Elenco delle figure

Elenco delle tabelle

Listings

Capitolo 1

Introduzione

Internet of Things, o IoT, è l'espressione utilizzata ormai da diversi anni per definire la rete delle apparecchiature e dei dispositivi, diversi dai computer, connessi a Internet: sensori per il fitness, elettrodomestici, caldaie, automobili, e più in generale qualsiasi dispositivo elettronico equipaggiato con un software che gli permetta di scambiare dati con altri oggetti connessi.
Per essere connesso, un oggetto deve rispettare due caratteristiche: avere un indirizzo che ne consenta l'identificazione sulla rete e avere la capacità di scambiare dati attraverso la rete stessa, senza bisogno dell'intervento umano. Obiettivo degli oggetti connessi è, in generale, quello di semplificare la vita, automatizzando processi o mettendo a nostra disposizione informazioni che prima non avevamo. Gli oggetti connessi alla rete possono quindi permettere di ottimizzare in tempo reale processi produttivi e attività economiche, riducendo inquinamento e consumo di risorse.
Nel caso di caldaie connesse, è possibile ricavare informazioni che ne descrivono il ciclo di vita. Tali informazioni riguardanti, ad esempio, pressione, temperatura dell'acqua, numero di accensioni giornaliere e altre ancora, vengono misurate nella quasi totalità con degli appositi sensori. Tramite degli specifici protocolli di comunicazione, le informazioni vengono poi raccolte per essere mandate a un gateway connesso ad internet ed inoltrate fino ad arrivare ad un database. Qui è utile, per i produttori, avere la possibilità di analizzare questi dati raccolti, in modo da poter monitorare, da remoto, lo stato di salute della caldaia e accorgersi di eventuali comportamenti anomali o guasti verificatisi. Il processo di analisi dei dati, che una volta veniva svolto solo manualmente, oggi, con l'avvento del machine learning, può essere realizzato in maniera automatica con l'uso di appositi algoritmi.
L'obiettivo del lavoro di tesi è definire e progettare le tecniche di analisi dei dati per la creazione di un modello predittivo dello stato di salute della caldaia nel lungo periodo (predictive maintainance), a partire dai dati raccolti durante il funzionamento della stessa. Con l'obiettivo di dover scalare nella gestione di milioni di caldaie, è stata prevista una soluzione basata su un'architettura Big Data. Per raggiungere tale obiettivo sono state affrontate le seguenti attività:

- Raccolta di conoscenza da esperti di dominio.

È stata predisposta una descrizione di guasti comuni e relative informazioni di correlazione con i dati di funzionamento della caldaia, interrogando un esperto del settore.

- Modellazione dei fenomeni di interesse a partire dai dati sperimentali.
  Sono stati raccolti dataset di 10 caldaie reali, durante la loro operatività ordinaria, per un periodo pari a 4 mesi. Le variabili misurate sono pressione, temperatura, contatore di accensioni, potenza del bruciatore e altre ancora, raccolte facendo uso di OpenTherm e ModBus, protocolli di comunicazione con le quali le caldaie scambiano dati con i gateway.

- Creazione di dataset sintetici per il test della soluzione proposta.
  É stato possibile testare gli algoritmi su grandi quantità di dati.

La prima parte del lavoro di tesi ha portato alla progettazione dell'architettura Big Data per l'analisi dei dati, basata principalmente su Core Spark, e sulle librerie di machine learning MLlib e Scikit Learn.
Il passo successivo è stato l'analisi esplorativa, che ha permesso la pulizia del dataset, inclusi l'identificazione dei duplicati, di dati mancanti, ed esigua presenza di guasti reali per la costruzione del modello predittivo degli stessi (classi estremamente sbilanciate). Sono stati quindi creati dataset sintetici, a partire da quelli reali, secondo le caratteristiche descritte dagli esperti di dominio. Tali dati sono stati analizzati e processati con un algoritmo multi-class di Machine Learning in quattro diversi ambienti di sviluppo:

- Spark in ambiente Java e libreria Scikit Learn per il ML;
- Spark in ambiente Java con la libreria MLlib per il ML.
- Spark in ambiente Python e libreria Scikit Learn per il ML;
- Python e libreria Scikit Learn per il ML;

Le prove con Spark in ambiente Java hanno beneficiato dell'uso del cluster messo a disposizione del Centro interdipartimentale SmartData@Polito, che ha permesso di verificare l'efficienza e la scalabilità in presenza di grandi quantità di dati.
Per indirizzare l'esigua presenza della classe di interesse (guasto) nei dataset reali raccolti, è stato applicato un approccio basato su un classificatore "one-class", in ambiente Python con la libreria di machine learning Scikit Learn.
L'applicazione dell'algoritmo one-class sui nuovi dati ha evidenziato la presenza di numerosi falsi positivi, sia nell'intorno temporale del guasto, sia nei periodi in cui non sono presenti guasti etichettati.
Per contrastare questo comportamento indesiderato, è stata quindi svolta un'analisi sulla probabilità che un campione nel dataset sia segnalato come un guasto (anomalia) dal classificatore one-class. Sono state utilizzate le scatter matrix, matrici composte da grafici a dispersione, derivate dalla decision function dell'algoritmo, che prende in esame le variabili a coppie. Da questa analisi è stato osservato che in corrispondenza delle zone a maggiore probabilità, si ha effettivamente riscontro con errori reali, verificatisi per la caldaia in

questione (veri positivi).
Infine, è stata svolta un'analisi su tutte le anomalie identificate dal classificatore one-class, distinguendole in veri e falsi positivi, ricavando le curve ROC (Receiver Operating Characteristics), al variare di due parametri: soglia (threshold) con la quale il classificatore discrimina se un campione è un'anomalia o no numero di giorni per i quali è corretto segnalare un campione come anomalo o no
Quest'analisi ha mostrato l'incremento della precisione dell'algoritmo al crescere del numero di giorni, e il conseguente decrescere del richiamo. Comportamento diverso si ha per l'incremento della soglia, per cui la precisione aumenta fino ad un valore massimo (che si ha in corrispondenza della soglia standard), per poi decrescere. Il richiamo invece, viene incrementato molto rapidamente fino a raggiungere e stabilizzarsi sul valore massimo.
Il lavoro è finalizzato allo studio dei limiti e dei risultati sperimentali di alcuni approcci Big Data basati su tecniche di machine learning per la predictive maintainance, in modo da poter intervenire prima che si verifichi il guasto.
Questo lavoro può essere approfondito con ulteriori sperimentazioni e integrato con altre tipologie di dati al fine di ottenere nuovi modelli e informazioni.

Capitolo 2

Internet of Things

2.1 IoT: Definizione

Internet of Things, o IoT, è l'espressione utilizzata ormai da diversi anni per definire la rete delle apparecchiature e dei dispositivi, diversi dai computer, connessi a Internet ed equipaggiati con un software che gli permetta di scambiare dati con altri oggetti connessi[26].
Per essere connesso, un oggetto deve rispettare due caratteristiche:

- avere un indirizzo IP che ne consenta l'identificazione in maniera univoca sulla rete
- essere in grado di scambiare dati attraverso la rete stessa, senza bisogno dell'intervento umano

Si può pensare ad esempio ad una smart home, ovvero ad una casa nella quale gli oggetti possono essere controllati a distanza facendo uso della rete wireless casalinga. Da qui è facile pensare a telecamere di sorveglianza, o a serrature apribili tramite lo smartphone, o ancora a termostati che regolano la temperatura dell'appartamento in maniera automatica. Solo pensando ad una smart home, e quindi alla domotica, il numero di oggetti collegabili ad internet è smisurato. Ma questi sono solo alcuni possibili esempi. L'Internet of Things ha moltissimi campi d'applicazione: auto, e più in generale i trasporti, healtcare e molti altri.

2.2 Architetture IoT

I dispositivi connessi sono sempre, o quasi sempre, microcontroller con risorse molto limitate. Raccolgono dati tramite sensori e li inviano a unità di memorizzazione, che possono essere locali, oppure remote. Per scambiare o inviare le informazioni raccolte, i dispositivi connessi hanno bisogno di dialogare tra di loro, o più in generale con altri dispositivi. Non esiste però un'architettura standard da seguire nel mondo dell'Internet of Things. Ogni soluzione IoT si può basare su architetture diverse, che dipendono da numerose variabili[38] .

Ma, nonostante ciò, è possibile seguire delle linee guida generali, da specializzare in funzione del proprio progetto o contesto.
Si deve tener conto di:

- oggetti hardware
- protocolli di comunicazione
- applicazioni software
- servizi di memorizzazione

Gli oggetti hardware possono comunicare con altri dispositivi in due modi[13]:

- a corto raggio: sono connessi ad Internet tramite un gateway, al quale sono collegati via cavo o tramite reti wireless o bluetooth.
- a lungo raggio: sono connessi ad Internet tramite SIM, in maniera autonoma.

I protocolli di comunicazione non sono standard, ma cambiano al variare del dispositivo e dell'ambito di riferimento. Caldaie intelligenti non useranno gli stessi protocolli di comunicazione che invece potrebbero usare le telecamere di sorveglianza.
Le possibili applicazioni software possono essere app per smartphone, tablet, computer, persino smartwatch, ma anche più semplicemente i browser.
Infine, i possibili servizi di memorizzazione possono essere drive locali, per esempio la memoria del proprio smartphone, o oppure drive remoti, e quindi servizi cloud, per il quale l'IoT sta fiorendo, date soprattutto le caratteristiche di elasticità, scalabilità.

2.3 IoT nelle caldaie

Gli ambiti applicativi dell'IoT sono molteplici. Nel caso di caldaie connesse, è possibile ricavare informazioni che ne descrivono il ciclo di vita. Tali informazioni riguardanti, ad esempio, pressione, temperatura dell'acqua, numero di accensioni giornaliere e altre ancora, vengono misurate nella quasi totalità con degli appositi sensori. Tramite degli appositi protocolli di comunicazione, le informazioni vengono poi raccolte per essere mandate a un gateway connesso ad internet ed inoltrate fino ad arrivare ad un database o, più in generale, a un centro di raccolta dati. Qui è utile, per i produttori, poter analizzare questi dati raccolti, in modo da poter monitorare da remoto lo stato di salute della caldaia e accorgersi di eventuali comportamenti anomali o guasti verificatisi. Tale processo di analisi dei dati, che una volta veniva svolto solo manualmente, oggi, con l'avvento del machine learning, può essere realizzato in maniera automatica con l'uso di appositi algoritmi.

2.4 Protocolli di comunicazione per le caldaie

Per la comunicazione delle caldaie sono disponibili principalmente 2 protocolli:

- OpenTherm

- ModBus

Entrambi sono utilizzati, ma ModBus consente alcune operazioni aggiuntive. Di seguito vengono analizzati singolarmente.

2.4.1 OpenTherm

OpenTherm è un protocollo standard utilizzato per la comunicazione tra una caldaia a riscaldamento centralizzato e un controller termostatico[50].
Essendo standard, in teoria è indipendente da ogni singolo produttore, per cui un controller prodotto da un costruttore, potrebbe essere usato da un altro. Ma, avendo parametri e funzioni opzionali, alcuni dispositivi potrebbero avere funzionalità non previste da altri, ma specifiche del produttore, che ne possono compromettere la compatibilità.
OpenTherm fa uso di una comunicazione punto-punto bidirezionale tra controller e caldaia. Il comando di base è quello di impostare la temperatura, ma è possibile trasferirne anche altri.
Il protocollo concede la possibilità, dalla versione 3, di inserire un dispositivo aggiuntivo(gateway) tra la caldaia (slave) e il controller (master). Il gateway è collegato da un lato ad una sola caldaia, ma dall'altro può essere collegato a più controller, ed ha la funzione di esaminare i comandi mandati alla caldaia.

2.4.2 ModBus

ModBus, a differenza di OpenTherm, è un protocollo inizialmente creato per mettere in comunicazione controller logici programmabili.
Non è uno standard, ma praticamente è il protocollo più utilizzato in ambito industriale, grazie ad alcune sue caratteristiche cardine:

- è pubblicato apertamente ed è royalty-free;
- semplice da installare e mantenere;
- è stato pensato e progettato per usi industriali;
- non impone troppe restrizioni nel trasferimento di bit.

Modbus consente a dispositivi diversi collegati alla stessa rete di comunicare tra loro. In maniera opposta ad OpenTherm, il protocollo consente la comunicazione tra un solo master e tanti slave. Solo il master può iniziare la comunicazione che è formata da una transazione domanda/risposta con un singolo slave o un messaggio broadcast a tutti gli slave senza risposta[29, 47].

2.5 Architettura IoT di riferimento

Sono possibili moltissime configurazioni per l'architettura IoT.
Nel caso in esame, si fa riferimento all'architettura in figura 2.1.

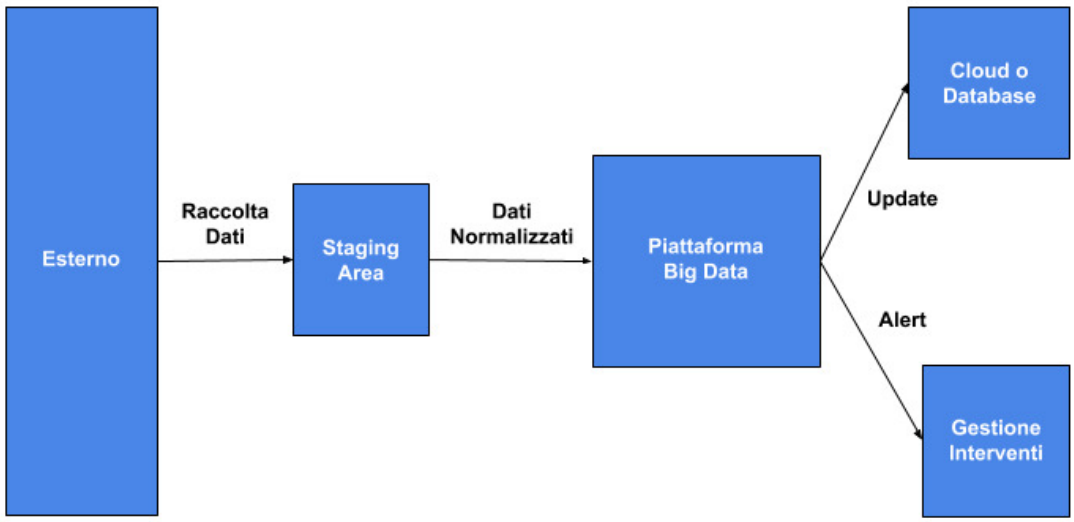


Figura 2.1: Architettura IoT

Tramite dei dispositivi (Gateway) vengono raccolti i dati provenienti dalle caldaie, che comunicano con i protocolli ModBus e/o OpenTherm, e inviati ad una staging area. Essa è un'area intermedia che permette l'elaborazione dei dati durante i processi di estrazione, trasformazione e caricamento (ETL).
Eventualmente, potrebbero essere raccolti dati meteo da correlare con quelli delle caldaie al fine di ottimizzare il consumo energetico.
I dati ora "normalizzati" verranno quindi inviati alla piattaforma big data per essere analizzati.
L'architettura compie statistiche ed analisi per poter individuare i possibili futuri guasti ed eventualmente lanciare un alert per segnalare un intervento da compiere.
Essa si occupa anche di calcolare dei report da inviare ad un database o sul cloud.

Capitolo 3

Big Data

3.1 Introduzione ai Big Data

Al giorno d'oggi è sempre più comune utilizzare social network, applicazioni di messaggistica, fare una semplice ricerca ed in generale utilizzare intenet. Ognuna di queste azioni genera dati. Tali dati, con il passare del tempo, stanno aumentando sempre di più, grazie all'evoluzione tecnologica e al conseguente aumento dei dispositivi connessi ad internet.
L'aumento della mole dei dati ha portato a problemi di gestione degli stessi con i database tradizionali. Per questo motivo si è iniziato a parlare di Big Data, e di nuovi strumenti di analisi, che siano in grado di gestire queste grosse quantità di dati.
Secondo molti analisti, il concetto di Big Data si basa sulle famose tre v[44]:

- Velocità, intesa come la rapidità con la quale vengono prodotti nuovi dati. L'analisi deve essere in grado di tenere il passo con la generazione dei nuovi dati. per questo motivo si parla anche di streaming, cioè di analisi in tempo reale.

- Varietà, intesa come la molteplicità delle sorgenti dalle quali provengono i dati e dei formati con i quali si possono presentare.

- Volume, inteso proprio come la quantità di dati prodotti, che sta aumentando nel tempo in modo esponenziale.

A queste, si sono aggiunte altre 2V che ne estendono la definizione:

- Veridicità, intesa come l'affidabilità dei dati, ovvero la qualità dei dati stessi e dei risultati prodotti da essi.

- Valore, inteso come la capacità di trarre vantaggi e benefici dai dati, in ambito business.

Tutte le aziende si stanno trovando ad affrontare il problema dei Big Data, dato il volume sempre crescente e la prevalenza di dati non strutturati (ad esempio social network), non analizzabili con i classici sistemi per la gestione di database relazionali.

Ciclo di vita dei Big Data

L'estrazione della conoscenza dai Big Data è suddivisa in varie fasi[44]:

- Generazione e acquisizione dei dati.
  In questa fase sono compresi il modo in cui i dati sono generati:
  - da persone, che possono generarli in maniera attiva (es. social network) o passiva (es. transazioni bancarie);
  - in modo automatico (es. caldaie, sensori);

  e anche il modo in cui vengono raccolti e inviati. In questa fase viene fatto anche un pre-processamento locale dei dati.
- Salvataggio dei dati.

  Questa è la fase in cui ci si occupa:
  - della struttura dove salvare i dati, che può essere locale o anche in rete (es. cloud);
  - del modo in cui salvare i dati, a seconda del file system e del tipo di dato.
- Analisi dei dati.
  Nell'ultima fase, in base all'analisi che si vuole affrontare, che può essere:
  - descrittiva, che sintetizza o descrive i dati per renderli più comprensibili all'uomo;
  - predittiva, che permette di predire comportamenti futuri, grazie all'analisi dei dati storici;
  - prescrittiva, che consente di capire i motivi e le cause che hanno portato al verificarsi di un evento;

  i dati raccolti vengono processati per estrarre informazioni. Possono essere usati modelli di programmazione basati su tecniche di data mining, machine learning o di analisi statistica, come clustering, correlazione e regressione.

3.2 Piattaforma Big Data

Un'architettura Big Data può essere composta da una grande varietà di componenti, in base alle proprie esigenze. Le combinazioni possibili sono tantissime.
Nella figura 3.1 ecco una rappresentazione dei componenti analizzati per il lavoro di tesi.

3.3 Componenti dell'architettura Big Data

Verranno ora presentati e analizzati i componenti che fanno parte dell'architettura.

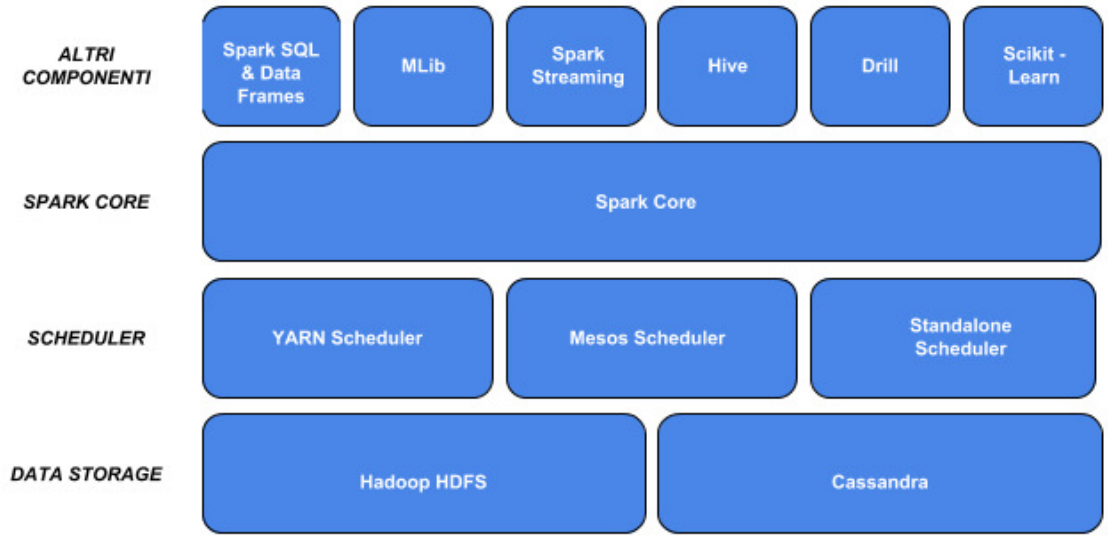


Figura 3.1: Piattaforma Big Data

3.3.1 Data Storage

Data Storage indica l'infrastruttura storage pensata per salvare e gestire grandi quantità di dati. In particolare, per i Big Data, è necessaria che sia anche facilmente accessibile e capace di scalare con l'aumento delle dimensioni dei dati.

Hadoop HDFS

HDFS é un file system distribuito ideato per soddisfare requisiti quali l'affidabilitá e la scalabilitá e per gestire un numero molto elevato di file, anche di dimensioni ragguardevoli (dell'ordine dei gigabyte o terabyte), attraverso la realizzazione di cluster che possono contenere migliaia di nodi. In HDFS i file sono organizzati in una struttura gerarchica di cartelle. Dal punto di vista dell'architettura, un cluster é costituito dai seguenti tipi di nodi:

- NameNode: é l'applicazione che gira sul server principale. Gestisce il file system ed in particolare il namespace, cioé l'elenco dei nomi dei file e dei blocchi (solitamente i file infatti vengono divisi in blocchi da 64/128 MB) e controlla l'accesso ai file, eseguendo le operazioni di apertura, chiusura e modifica dei nomi dei file. Inoltre, determina come i blocchi dati siano distribuiti sui nodi del cluster e la strategia di replica che garantisce l'affidabilitá del sistema. Il NameNode controlla che i singoli nodi siano in esecuzione senza problemi e in caso contrario decide come riallocare i blocchi. Il NameNode distribuisce le informazioni contenute nel namespace su due file: il primo prende il nome fsimage e rappresenta l'ultima immagine del namespace; il secondo é un log dei cambiamenti avvenuti al namespace a partire dall'ultimo aggiornamento del file fsimage.

- DataNode: é una applicazione che gira su altri nodi del cluster, generalmente ve ne é una per nodo, e gestisce fisicamente lo storage di ciascuno di essi. Queste applicazioni eseguono, logicamente, le operazioni di lettura e scrittura richieste dai client e gestiscono fisicamente la creazione, la cancellazione e la replica dei blocchi dati.
- SecondaryNameNode: é un servizio che aiuta il NameNode ad essere piú efficiente.
- BackupNode: é il nodo di failover e consente di avere un nodo simile al SecondaryNameNode sempre sincronizzato con il NameNode.

Come detto in precedenza, i file sono organizzati in blocchi da 64 o 128 MB e sono ridondanti su piú nodi. Sia la dimensione dei blocchi, sia il numero di repliche possono essere configurate da ciascun utente per ogni file. Le repliche sono utilizzate per garantire l'accesso a tutti i dati (anche in presenza di problemi a uno o piú nodi) e per rendere piú efficiente il recupero dei dati. In HDFS le richieste di lettura dati seguono una politica relativamente semplice: esse avvengono scegliendo i nodi piú vicini al client che effettua la lettura e, ovviamente, in presenza di dati ridondanti risulta piú semplice soddisfare questo requisito. Quanto detto fino ad ora, ci permette di asserire che quando vengono trattati file di grandi dimensioni HDFS é molto efficiente. Ma quando vengono trattati file di piccole dimensioni, dove per piccole dimensioni si intendono dimensioni inferiori al blocco, HDFS risulta molto inefficiente perché i file utilizzano spazio all'interno del namespace, cioé l'elenco dei file mantenuti dal NameNode, che ha un limite dato dalla memoria del server. Questo problema viene risolto compattando molti file piccoli in file piú grandi ai quali é possibile accedere dal sistema in parallelo e senza necessitá di espanderli[1, 4, 5, 34].

Cassandra

Cassandra è un database management system non relazionale distribuito con licenza open source e ottimizzato per la gestione di grandi quantità di dati.
Si tratta di un progetto Top-Level, sviluppato da Apache Software Foundation per gestire grandi quantità di dati dislocati in diversi server, fornendo inoltre un servizio orientato alla disponibilità, senza point of failure. Cassandra fornisce una struttura di memorizzazione chiave-valore, con Eventual Consistency. Alle chiavi corrispondono dei valori, raggruppati in famiglie di colonne: una famiglia di colonne è definita quando il database viene creato. Tuttavia le colonne possono essere aggiunte a una famiglia in qualsiasi momento. Inoltre, le colonne sono aggiunte solo specificando le chiavi, così differenti chiavi possono avere differenti numeri di colonne in una data famiglia. I valori di una famiglia di colonne sono memorizzati insieme, in quanto Cassandra adotta un approccio ibrido tra DBMS orientato alle colonne e la memorizzazione orientata alle righe.
L'architettura di Cassandra contribuisce notevolmente alla sua capacità di scalare, eseguire e offrire continuamente disponibilità. Cassandra è stata costruita da zero con la consapevolezza che i guasti hardware e di sistema si verificano. Ciò si traduce in Cassandra con un diverso modo di gestire e proteggere i dati rispetto agli RDBMS tradizionali.
Piuttosto che usare una struttura master-slave o un design accattivante difficile da gestire,

Cassandra ha un'architettura distribuita peer-to-peer che è molto più elegante e facile da configurare e mantenere. In Cassandra, tutti i nodi sono uguali; non c'è il concetto di nodo master, con tutti i nodi che comunicano tra loro tramite un gossip protocol.
Cassandra fornisce la distribuzione automatica dei dati su tutti i nodi che fanno parte di un anello o un cluster di database. Il processo di distribuzione dei dati è automatico, lo sviluppatore non deve fare nulla. I dati vengono partizionati in modo trasparente su tutti i nodi in modo casuale (di default) o ordinato.
Cassandra fornisce inoltre una replica integrata e personalizzabile, che archivia copie di dati ridondanti sui nodi che partecipano ad un anello di Cassandra. Ciò significa che se un nodo in un cluster si arresta, uno o più copie dei dati di quel nodo sono disponibili su altre macchine nel cluster.
Basta indicare semplicemente quante copie di dati si desidera avere, e Cassandra si prende cura di tutto il resto. Sono fornite opzioni di replica che consentono anche la trasmissione automatica dei dati memorizzati in diversi rack fisici, più data center e piattaforme cloud.
Cassandra fornisce una vera architettura "indipendente dalla posizione" quando si tratta di leggere e scrivere dati. Questo significa che qualsiasi nodo in un cluster Cassandra possa essere letto o scritto, il che si traduce in un vero design di lettura / scrittura ovunque.
Quando i dati vengono scritti su Cassandra, questo viene prima scritto in un registro di commit, che garantisce la massima durata e sicurezza dei dati. I dati vengono anche scritti su una struttura in memoria chiamata memtable, che viene infine scaricata su una struttura del disco chiamato sstable (sorted strings table).
Se uno o più nodi responsabili di un determinato insieme di dati crashano, i dati vengono semplicemente scritti su un altro nodo, che detiene temporaneamente i dati. Quando i nodi tornano online, i dati aggiornati vengono riportati automaticamente, prelevandoli dai nodi che li detengono temporaneamente. La lettura dei dati viene eseguita in parallelo attraverso un cluster. Un utente richiede i dati da un qualsiasi nodo (che diventa il nodo coordinatore dell'utente), con la query dell'utente che viene assemblata da uno o più nodi contenenti i dati necessari. Se un particolare nodo ha il richiesto i dati diventa non disponibile, Cassandra richiede semplicemente dati da un altro nodo in possesso di una copia replicata di tali dati.
Non essendo un database transazionale esso offre la parte "AID" di ACID, in quanto i dati scritti sono atomici, isolati e durevoli. La "C" di ACID non si applica a Cassandra, in quanto non vi è alcun concetto di integrità referenziale o chiavi esterne. In un certo senso, si può dire che Cassandra offra supporto per le transazioni Big Data (OLTP).
Cassandra offre una regolabile data consistency su un cluster di database. La consistenza può essere gestita sulla singola operazione, il che significa che uno sviluppatore può decidere quanto sia forte la consistenza finale per le operazioni di SELECT, INSERT, UPDATE e DELETE [15, 30, 3, 46, 5].

3.3.2 Scheduler

Il termine scheduler sta ad indicare i cluster manager, ovvero software eseguiti in uno o in tutti i nodi del cluster. È utilizzato per gestire e configurare servizi, o in generale per la gestione completa del cluster stesso.

Yarn Scheduler

Lo scheduler YARN è il cluster manager di Hadoop, che funge da scheduler generico e distribuito delle applicazioni e delle risorse. Introdotto in Hadoop 2 per superare i limiti della versione 1, YARN risponde alle richieste del client creando un container, ovvero un concetto astratto che rappresenta la collezione di risorse fisiche allocate per una applicazione. YARN monitora l'esecuzione del container, terminandola se necessario.
Apache Hadoop YARN ha un ResourceManager con due parti, uno Scheduler e un ApplicationsManager. Lo Scheduler è un componente collegabile. Vengono fornite due implementazioni, un CapacityScheduler, utile in un cluster condiviso da più di un'organizzazione, e il FairScheduler, che garantisce a tutte le applicazioni, in media, un uguale numero di risorse. Entrambi gli scheduler assegnano le applicazioni a una coda e ciascuna coda ottiene risorse condivise equamente tra loro. All'interno di una coda, le risorse sono condivise tra le applicazioni. ApplicationManager è responsabile di accettare le sottomissioni dei job e di 'avviare l'applicazione ApplicationsMaster specifica. In questo caso, ApplicationsMaster è l'applicazione Spark. Nell'applicazione Spark, le risorse sono specificate nell'oggetto SparkConf dell'applicazione.
YARN supporta il ripristino manuale utilizzando un'utilità della riga di comando e supporta il ripristino automatico tramite un ActiveStandbyElector basato su Zookeeper incorporato nel ResourceManager. Pertanto, a differenza di Mesos e dei gestori autonomi, non è necessario eseguire un controller di failover ZooKeeper separato. ZooKeeper viene utilizzato solo per registrare lo stato dei ResourceManager[25, 28].

Mesos Scheduler

Apache Mesos è un progetto open source per la gestione dei cluster. È stato sviluppato presso l'Università della California, a Berkeley.
Mesos utilizza Cgroup Linux per fornire isolamento per CPU, memoria, I / O e file system. Esso è paragonabile allo scheduler di Google, una piattaforma altamente segreta utilizzata internamente per gestire e distribuire i servizi di Google. Mesos ha HA (high availability) per master e slave, può gestire risorse per l'applicazione e ha il supporto per i contenitori Docker. Può eseguire lavori Spark, Hadoop MapReduce o qualsiasi altra applicazione di servizio. Ha API per Java, Python e C ++. Può funzionare su Linux o Mac OSX.
Apache Mesos ha processi master e slave. Il master fa offerte di risorse per l'applicazione che accetta o meno l'offerta. Pertanto, la richiesta di risorse disponibili e di lavori in corso è determinata dall'applicazione stessa. Apache Mesos consente un controllo fine delle risorse in un sistema come cpu, memoria, dischi e porte. Offre un controllo così fine che Spark può assegnare in anticipo un numero fisso di CPU a ciascun esecutore che non viene rilasciato fino alla chiusura dell'applicazione. Si noti che nello stesso cluster, alcune applicazioni possono essere impostate per utilizzare il controllo fine mentre altre no.
Il gestore cluster Apache Mesos supporta anche il recupero automatico del master utilizzando ZooKeeper Apache. Le attività attualmente in esecuzione continuano a farlo in caso di failover[25, 28].

Standalone Scheduler

Il gestore cluster Spark Standalone è un semplice gestore cluster disponibile come parte della distribuzione Spark. Ha HA per il master, è resiliente ai guasti lavorativi, ha capacità di gestione delle risorse per l'applicazione e può essere eseguito insieme a una distribuzione Hadoop esistente e accedere ai dati HDFS (Hadoop Distributed File System). La distribuzione include script per semplificare la distribuzione locale o nel cloud su Amazon EC2. Può funzionare su Linux, Windows o Mac OSX.
Spark Standalone utilizza un semplice scheduler FIFO per le applicazioni. Per impostazione predefinita, ciascuna applicazione utilizza tutti i nodi disponibili nel cluster. Il numero di nodi può essere limitato per applicazione, per utente o globalmente. Altre risorse, come memoria, cpus, ecc. possono essere controllate tramite l'oggetto SparkConf dell'applicazione. Supporta il ripristino automatico del master utilizzando dei master in standby tramite ZooKeeper. Supporta anche il recupero manuale utilizzando il file system. Il cluster è resiliente ai guasti del nodo in funzione indipendentemente dal fatto che il ripristino del Master sia abilitato[25].

3.3.3 Altri Componenti

Questa sezione raggruppa i rimanenti componenti, tra i quali:

- Core Spark;
- componenti per eseguire query;
- librerie di machine learning;
- componente per lo streaming.

Spark Core

Il componente Spark Core è la base per l'elaborazione parallela e distribuita di grandi set di dati. Il componente Spark Core è responsabile di tutte le funzionalità di I / O di base, pianificazione e monitoraggio dei lavori su cluster spark, dispacciamento delle attività, networking con diversi sistemi di storage, ripristino dei guasti ed efficiente gestione della memoria.
Spark Core utilizza una struttura dati speciale nota come RDD (Resilient Distributed Dataset). La condivisione o il riutilizzo dei dati in sistemi di calcolo distribuiti come Hadoop MapReduce richiede che i dati vengano archiviati in sistemi intermedi come Amazon S3 o HDFS. Ciò rallenta la velocità complessiva di calcolo a causa delle diverse repliche, operazioni di I / O e serializzazioni nella memorizzazione dei dati in questi archivi di dati intermedi stabili. Gli RDD risolvono questo inconveniente di Hadoop MapReduce consentendo calcoli "in memoria" tolleranti ai guasti.
Gli RDD sono una raccolta di record immutabile e partizionata che può essere utilizzata in parallelo. Essi possono contenere qualsiasi tipo di oggetti Python, Scala, Java o anche

oggetti di classe definiti dall'utente. Gli RDD vengono generalmente creati dalla trasformazione di RDD esistenti o caricando un set di dati esterno da una memoria stabile come HDFS o HBase[11].

Spark SQL e DataFrames

Spark SQL è un modulo Spark per l'elaborazione di dataset strutturati. A differenza dell'API Spark RDD di base, le interfacce fornite da Spark SQL forniscono a Spark ulteriori informazioni sulla struttura di entrambi i dati e il calcolo in esecuzione. Internamente, Spark SQL utilizza queste informazioni extra per eseguire ulteriori ottimizzazioni. Esistono diversi modi per interagire con Spark SQL, tra cui SQL e l'API Dataset. Quando si calcola un risultato viene utilizzato lo stesso motore di esecuzione, indipendente da quale API / linguaggio si sta utilizzando per esprimere il calcolo. Questa unificazione significa che gli sviluppatori possono facilmente passare da una API all'altra in base a quella che fornisce il modo più naturale per esprimere una determinata trasformazione.
Un uso di Spark SQL è eseguire query SQL. Spark SQL può anche essere utilizzato per leggere i dati da un'installazione Hive esistente. Quando si esegue SQL da un altro linguaggio di programmazione, i risultati verranno restituiti come Dataset / DataFrame. È anche possibile interagire con l'interfaccia SQL utilizzando la riga di comando o tramite JDBC / ODBC.
Un Dataset è una raccolta distribuita di dati. Essa è una nuova interfaccia aggiunta in Spark 1.6 che offre i vantaggi degli RDD (forte tipizzazione, possibilità di utilizzare potenti funzioni lambda) con i vantaggi del motore di esecuzione ottimizzato di Spark SQL. Un Dataset può essere costruito da oggetti JVM e quindi manipolato utilizzando trasformazioni funzionali (map, flatMap, filter, ecc.). L'API Dataset è disponibile in Scala e Java. Python non ha il supporto per l'API Dataset. Ma a causa della natura dinamica di Python, molti dei vantaggi dell'API Dataset sono già disponibili.
Un DataFrame è un set di dati organizzato in colonne nominate. È concettualmente equivalente a una tabella in un database relazionale o in un frame di dati in R / Python, ma con ottimizzazioni più ricche sotto il cofano. I DataFrames possono essere costruiti da una vasta gamma di fonti come: file di dati strutturati, tabelle Hive, database esterni o RDD esistenti. L'API DataFrame è disponibile in Scala, Java, Python e R[11, 19, 40].

MLlib

MLlib è una libreria di machine learning di basso livello che può essere chiamata dai linguaggi di programmazione Scala, Python e Java. MLlib è semplice da utilizzare, scalabile, compatibile con vari linguaggi di programmazione e può essere facilmente integrato con altri strumenti. MLlib facilita l'implementazione e lo sviluppo di pipeline scalabili per l'apprendimento automatico ed ha implementazioni per vari algoritmi di machine learning comuni[11, 17, 18]:

- Clustering- K-means;
- Classificazione: naïve Bayes, regressione logistica, SVM;

- Decomposizione - Principal Component Analysis (PCA) e Singular Value Decomposition (SVD);
- Regressione - Regressione lineare;
- Filtro collaborativo: alternando i minimi quadrati per le raccomandazioni.

Gli usi più comuni per MLib sono:

- Ottimizzazione e manutenzione della catena di fornitura;
- Ottimizzazione della pubblicità: per scoprire la probabilità che gli utenti facciano clic sugli annunci disponibili;
- Marketing: consigliare prodotti ai clienti per massimizzare le entrate o il coinvolgimento;
- Rilevamento antifrode: per tenere traccia del comportamento anomalo degli utenti.

Spark Streaming

Spark Streaming è un'API leggera che consente agli sviluppatori di eseguire l'elaborazione in batch e lo streaming di dati con facilità, nella stessa applicazione. Gli Stream Discretizzati formano l'astrazione di base in Spark Streaming. Utilizza un flusso continuo di dati di input (Discretized Stream o Streaming di una serie di RDD) per elaborare i dati in tempo reale. Spark Streaming sfrutta la rapida capacità di pianificazione di Apache Spark Core per eseguire analisi di streaming mediante l'importazione di dati in mini-lotti. Le trasformazioni vengono applicate a quei mini batch di dati. I dati di Spark Streaming vengono ingeriti da varie fonti di dati e flussi live come Twitter, Apache Kafka, Akka Actors, IoT Sensors, Amazon Kinesis, Apache Flume, ecc e in applicazioni con drive di eventi, fault-tolerant e type-safe.
Le principali caratteristiche di Spark Streaming sono:

- Elaborazione di flussi di dati in tempo reale semplice, affidabile e veloce;
- Gli sviluppatori Spark possono riutilizzare lo stesso codice per l'elaborazione di flussi e batch e possono anche integrare i dati di streaming con i dati storici;
- Spark Streaming ha esattamente una volta il messaggio che garantisce e aiuta a recuperare il lavoro perso senza dover scrivere alcun codice aggiuntivo o aggiungere ulteriori configurazioni;
- Spark streaming supporta l'inclusione di Spark MLlib per le pipeline di apprendimento automatico nei percorsi dati.

Spark Streaming viene utilizzato in applicazioni che richiedono statistiche in tempo reale e risposta rapida come allarmi, sensori IoT, diagnostica, sicurezza informatica, ecc. Spark Streaming trova grandi applicazioni nell'elaborazione dei registri, nel rilevamento delle intrusioni e nel rilevamento delle frodi. Esso è molto utile anche per pubblicità online e campagne, finanza, gestione della supply chain, ecc [11, 20].

Hive

Apache Hive è un sistema di data warehousing basato su Apache Hadoop per fornire data summarization, query e analisi dei dati. Hive fornisce un'interfaccia simile a SQL per interrogare i dati memorizzati in vari database e file system che si integrano con Hadoop. Le query SQL tradizionali devono essere implementate nell'API Java di MapReduce per eseguire applicazioni SQL e query su dati distribuiti. Hive fornisce l'astrazione SQL necessaria per integrare query SQL-like (HiveQL) in Java senza la necessità di implementare query nell'API Java di basso livello. Poiché la maggior parte delle applicazioni di data warehousing funzionano con linguaggi di query basati su SQL, Hive supporta la portabilità delle applicazioni basate su SQL su Hadoop. Inizialmente sviluppato da Facebook, Apache Hive è utilizzato e sviluppato da altre società come Netflix e Financial Industry Regulatory Authority (FINRA)[35, 43].
I principali componenti di Hive sono:

- Metastore: memorizza i metadati per ciascuna delle tabelle come il loro schema e la loro posizione. Include anche i metadati della partizione che aiutano il driver a tenere traccia dell'avanzamento di vari set di dati distribuiti sul cluster. I dati sono memorizzati in un formato RDBMS tradizionale. Un server di backup replica regolarmente i dati che possono essere recuperati in caso di perdita;

- Driver: agisce come un controller che riceve le istruzioni HiveQL. Inizia l'esecuzione dell'istruzione creando sessioni e monitora il ciclo di vita e l'avanzamento dell'esecuzione. Memorizza i metadati necessari generati durante l'esecuzione di un'istruzione HiveQL. Il driver funge anche da punto di raccolta dei dati o risultato della query ottenuto dopo l'operazione REDUCE;

- Compilatore: esegue la compilazione della query HiveQL, convertendola in un piano di esecuzione. Questo piano contiene le attività e i passaggi che devono essere eseguiti da Hadoop MapReduce per ottenere l'output come tradotto dalla query. Il compilatore converte la query in un abstract syntax tree (AST). Dopo aver controllato gli errori di compatibilità e di compilazione del tempo, converte l'AST in un grafico aciclico diretto (DAG). Il DAG divide gli operatori in fasi e attività di MapReduce in base alla query e ai dati di input;

- Ottimizzatore: esegue varie trasformazioni nel piano di esecuzione per ottenere un DAG ottimizzato. Le trasformazioni possono essere aggregate insieme, come la conversione di una pipeline di join in un singolo join, per prestazioni migliori. Può inoltre suddividere le attività, ad esempio applicare una trasformazione sui dati prima di un'operazione di riduzione, per fornire prestazioni e scalabilità migliori. Tuttavia, la logica di trasformazione utilizzata per l'ottimizzazione può essere modificata o eseguita in pipeline utilizzando un altro ottimizzatore;

- Esecutore: dopo la compilazione e l'ottimizzazione, l'esecutore esegue le attività. Interagisce con il job tracker di Hadoop per pianificare le attività da eseguire. Si occupa della pipeline delle attività assicurandosi che un'attività con dipendenze venga eseguita solo se vengono eseguiti tutti gli altri prerequisiti;

- CLI, UI e Thrift Server: la common line inteface (CLI) fornisce un'interfaccia utente per un utente esterno per interagire con Hive inviando query, istruzioni e monitorando lo stato del processo. Server Thrift consente ai client esterni di interagire con Hive su una rete, in modo simile ai protocolli JDBC o ODBC.

Apache Drill

Apache Drill è un framework open source che supporta applicazioni distribuite per analisi interattive su grandi insiemi di dati. Drill è la versione open source di Google Dremel disponibile come servizio infrastrutturale chiamato Google BigQuery. Un obiettivo dichiarato è che Drill sia capace di scalare fino a 10.000 server o più e di elaborare petabyte di dati e trilioni di record al secondo. Drill è un progetto di primo livello di Apache.
Drill supporta una vasta gamma di database e file system NoSQL, tra cui HBase, MongoDB, MapR-DB, HDFS, MapR-FS, Amazon S3, Azure Blob Storage, Google Cloud Storage, Swift, NAS e file locali. Una singola query può unire i dati da più datastore. Ad esempio, è possibile unire una raccolta di profili utente in MongoDB con una directory di registri eventi in Hadoop.
L'ottimizzatore di Datastore-aware di Drill ristruttura automaticamente un piano di query per sfruttare le capacità di elaborazione interna del datastore. Inoltre, Drill supporta la localizzazione dei dati, se Drill e il datastore si trovano sugli stessi nodi. Alcune delle principali caratteristiche di Drill sono[16, 32, 42]:

- Modello a documento JSON senza schema simile a MongoDB e Elasticsearch;
- Standard API: ANSI SQL, ODBC / JDBC, API RESTful;
- Facile da usare per l'utente e lo sviluppatore;
- Architettura che consente la connettività a più fonti di dati;
- Supporto a Hadoop (HDFS API 2.3+), MongoDB, Amazon EMR, MapR, CDH, HBase;
- Supporto alle piattaforme cloud: Amazon S3, Google Cloud Storage, Azure Blob Storage, Swift.

Scikit-Learn

Libreria open source per machine learning per il linguaggio Python[52]. Semplice ed efficiente per fare data mining e analisi di dati[8]. Contiene algoritmi di classificazione, regressione, clustering e altro ancora, ed è progettata per operare con le librerie NumPy e SciPy.
Scikit-Learn è caratterizzata da un'API chiara, uniforme e semplificata, nonché da una documentazione online molto utile e completa. Un vantaggio di questa uniformità è che una volta compreso l'uso di base e la sintassi di Scikit-Learn per un tipo di modello, passare a un nuovo modello o algoritmo è molto semplice.

3.3.4 Compatibilità dei componenti

Hadoop HDFS

HDFS ha pienà compatibilità con tutti gli scheduler.

- YARN è il cluster manager di Hadoop, che funge da scheduler generico e distribuito delle applicazioni e delle risorse. È stato introdotto in Hadoop 2 per superare i limiti della versione 1. Essendo uno standard, l'integrazione è davvero buona con la comunità Hadoop, in particolar modo quindi con HDFS.
- Mesos è un cluster manager costruito sui principi del kernel linux, e che può essere eseguito su qualsiasi macchina. Spark è stato sviluppato tenendo conto di Mesos come scheduler, per cui supporta tutto ciò che Spark supporta.
- Essendo disponibile come parte della distrubuzione Spark, lo Standalone Scheduler può processare i dati provenienti da HDFS e altri file system o repository di dati che Spark supporta.

Per quanto riguarda il Core, Spark non ha un ambiente di storage per le distribuzioni locali. Deve appoggiarsi su una piattaforma di archiviazione in lettura-scrittura come il file system distribuito Hadoop (HDFS).
Anche tutti gli ambienti per query SQL risultano pienamente compatibili con HDFS:

- Hive è la soluzione SQL su Hadoop originale, che tenta di emulare il comportamento, la sintassi e le interfacce di MySQL, incluso un client a riga di comando. Nonostante la relativa semplicità e facilità d'uso, Hive è lento e di sola lettura, il che ha provocato una serie di iniziative per migliorarlo.
- Spark SQL si basa su Spark per consentire la scrittura di query SQL sui dati. Dato che riutilizza i pezzi chiave della tecnologia Hive, può essere pensato allo stesso modo di come si presenta Hive per Hadoop.
- Implementazione open source di Google Dremel (ovvero BigQuery), Drill è stato ideato per eseguire query a bassa latenza su più tipi di archivi dati contemporaneamente con interfacce di query diverse (come HDFS e database NoSQL) e per essere altamente scalabile.

Spark Streaming è un'estensione del core Spark che consente l'elaborazione in streaming ad alta velocità e tolleranza agli errori dei flussi di dati in tempo reale. I dati possono essere ingeriti da molte fonti come Kafka, Flume, HDFS, Twitter, ZeroMQ o semplici vecchi socket TCP ed essere elaborati utilizzando algoritmi complessi espressi con funzioni di alto livello come map, reduce, join e window. Infine, i dati elaborati possono essere trasferiti a filesystem (HDFS), database e dashboard live.
MLlib si inserisce nelle API di Spark e interagisce con NumPy in Python e librerie R. Si può utilizzare qualsiasi sorgente di dati Hadoop (ad esempio HDFS, HBase o file locali), semplificando l'inserimento nei flussi di lavoro Hadoop.
Un caso d'uso importante per i big data è l'apprendimento automatico. Specialmente con

Hadoop, scikit-learn è molto importante, poiché è una delle migliori opzioni a disposizione per ottenere un modello di machine learning sui big data. L'apprendimento automatico su larga scala è attualmente uno degli argomenti più caldi, e farlo in un ambiente di big data come Hadoop è ancora più importante. Ora, gli aspetti significativi di machine learning sono un modello su una quantità significativamente grande di dati e ottenere un numero significativo di dati[53, 4].

Cassandra

L'unico progetto esistente in cui sono stati integrati Cassandra e Hive è Brisk, che è divenuto una sottoscrizione di Datastax Enterprise. La soluzione alternativa è fare tutto da zero.
Esisteva una patch da applicare a Cassandra per potervi usare Drill, ma dall'ultima versione di Drill, questa non è più funzionante, per cui deve essere fatta una modifica.
I dati in input per Spark Streaming possono provenire da quasi tutte le fonti, inclusi Amazon Kinesis, Kafka, Cassandra, Twitter e HDFS (così come fonti personalizzate), e possono essere memorizzati in quasi tutti i sistemi di sink, come un file system (S3, HDFS ...) o database (Cassandra, Hbase ...).
Usando lo Spark Cassandra connector (Datastax), i dati di Cassandra vengono presentati come RDD, per cui è possibile usarvi sopra tutte le librerie che lavorano con RDD e quindi anche con Spark[5, 15, 30, 3].

Spark Core

Spark è progettato per scalare in modo efficiente da una a molte migliaia di nodi di calcolo. Per raggiungere questo obiettivo, massimizzando la flessibilità, Spark può essere eseguito su una varietà di gestori di cluster, tra cui Hadoop YARN, Apache Mesos e un semplice gestore cluster incluso in Spark stesso denominato Standalone. Se si sta installando Spark su un insieme di macchine ancora libero, lo scheduler Standalone fornisce un modo semplice per iniziare; se si dispone già di un cluster Hadoop YARN o Mesos, tuttavia, il supporto di Spark per questi gestori cluster consente alle applicazioni di essere eseguite anche su di essi[25].

Spark SQL, Hive, Spark Streaming, MLlib

SparkSQL è il più recente strumento SQL-on-Hadoop. Si integra con Hive e usa di default Hive Metastore per gestire i suoi metadati. Ha un proprio ottimizzatore di query chiamato Catalyst che costruisce un albero di operatori ottimizzando la query per poi generare il codice binario per eseguire il task. A differenza di Hive, non vi è alcuna possibilità di cambiare i motori di esecuzione qui. SparkSQL si integra e funziona perfettamente con altre librerie Spark come Spark streaming, Spark core ecc[53, 19].

MLlib e Scikit-Learn

Scikit-Learn offre prestazioni eccezionali se i dati si inseriscono nella RAM. Python e Scikit-Learn eseguono l'elaborazione in memoria e in modo non distribuito.
Spark's ML Lib è adatto quando si fa ML relativamente semplice su un ampio set di dati. ML Lib non è efficiente dal punto di vista computazionale per i piccoli set di dati, per cui scikit-learn è più indicata per set di dati di piccole e medie dimensioni (megabyte, fino a pochi gigabyte). Per set di dati molto più grandi, meglio usare Spark MLib. Le due librerie possono essere combinate facilmente, per cui è possibile agire in questo modo[37, 36].

3.3.5 Confronto tra i componenti

I due metodi di storage esaminati (tabella 3.1) hanno alcune caratteristiche equivalenti, e nello specifico volume dei dati, scalabilità, richiesta di spazio, licenza[5].
Le caratteristiche per le quali hanno invece una profonda differenza sono:

- l'architettura, che è master-slave in HDFS, mentre è ad anello in Cassandra;
- bilanciamento, che in HDFS è fatto nativamente, mentre in Cassandra si fa uso di tool esterni;
- accesso ai file, poichè HDFS usa le proprie API in Java o browser Http, mentre Cassandra fa uso del proprio linguaggio (CQL, Cassandra Query Language).

Per le rimanenti caratteristiche, HDFS e Cassandra sono simili, ma con piccole differenze.

Tabella 3.1: Storage: HDFS vs CFS

Nome	**HDFS (Hadoop Distributed File System)**	**CFS (Cassandra File System)**
Architettura	Master-Slave, con un master (NameNode) e tanti slave (DataNode)	Distribuita ad anello, con collegamenti peer to peer e nodi paritetici
Licenza	OpenSource	OpenSource
Volume dei dati	Grosse Moli (∼GB - TB)	Grosse Moli (∼GB - TB)
Scalabilità	Altamente scalabile con l'aumento quantitativo dell'hardware	Molto scalabile, aumenta il throughput con l'aumento del numero di nodi nel cluster
Bilanciamento	Il NameNode è responsabile dell'ottimizzazione del bilanciamento del carico	Fa uso di tool per bilanciare i nodi nel cluster
Failure detection e tolleranza ai guasti	C'è un checksum nel contenuto dei file HDFS per poter trovare i dati corrotti e attuare misure correttive	Usa una versione modificata dell'Accrual Failure Detector

Replicazione dei dati	Si, fa uso dei criteri di posizionamento consapevole della replica su rack di Hadoop	Si, fa uso dei criteri di posizionamento consapevole e inconsapevole della replica su rack e datacenter
Accesso ai file	Tramite le API HDFS in Java o browser HTTP	Tramite CQL
Organizzazione dei dati	Ogni file è diviso in blocchi, che sono sparsi per i server. Tipicamente, ogni blocco è di 64-128 MB.	Dati memorizzati nelle famiglie di colonne inode (metadati) e sblock (dati)
Richiesta di spazio	Quando un file viene cancellato dall'utente nell'applicazione, esso non è subito rimosso da HDFS, ma viene spostato temporaneamente nella directory /trash in modo da essere velocemente recuperato in caso di bisogno. Passato un certo intervallo di tempo, viene eliminato.	Quando un file viene cancellato dall'utente nell'applicazione, esso non è subito rimosso, ma gli viene associata una lapide(una sorta di flag), in modo da essere velocemente recuperato in caso di bisogno. Passato un certo intervallo di tempo, viene eliminato.
Sicurezza - Autenticazione e validazione	Di default nessuna sicurezza. Tuttavia. può essere usato il protocollo Kerberos per autenticazione e autorizzazione	Usa Kerberos e SSL per scopi di autenticazione e autorizzazione. Inoltre usa log di commit per garantire che i dati non vengano persi

Per quanto concerne gli scheduler (tabella 3.2), essi mettono a disposizione le stesse tipologie di algoritmi, hanno gestione simile per quanto riguarda la sicurezza e l'availability. Si differenziano specificatamente per[25, 28]:

- capacità dello scheduler, che è diverso in ognuno, e con la particolarità che solo Mesos fornisce un controllo più fine delle risorse;
- sicurezza, per la quale tutti forniscono autenticazione, ma in maniera diversa per ognuno.

Tabella 3.2: Scheduler: Standalone vs Mesos vs Yarn

Nome	**Standalone**	**Mesos**	**YARN**

Capacità dello scheduler	Usa un semplice scheduler FIFO per le applicazioni. Ogni applicazione usa di default tutti i nodi disponibili nel cluster.	Ha processi master e slave. Il master fa una proposta di risorse all'applicazione che accetta/rifiuta. Consente controlli fini delle risorse	Ha un manager delle risorse, composto dallo scheduler e da un manager della applicazione.
High Availability	Supporta recupero automatico del master usando degli standby master tramite ZooKeeper, ma anche il recupero manuale del sistema tramite il file system.	Supporta recupero automatico del master tramite ZooKeeper. I task continuano a fare cose anche in casi di failure.	Supporta recupero automatico del master anche senza uso di ZooKeeper, e manuale tramite una utility a riga di comando.
Sicurezza	Supporta autenticazione tramite segreto condiviso con tutti i cluster manager. L'utente deve configurare il segreto su ogni nodo manualmente. I dati possono essere criptati tramite SSL, SASL e altro.	Fornisce autenticazione (opzionale) per ogni entità che interagisce con il cluster. E' presente una lista del controllo per autorizzare gli accessi. Supporta Cyrus SASL, SSL/TLS, e anche HTTPS.	Fornisce sicurezza per l'autenticazione, per console web e per confidenzialità dei dati. Hadoop usa Kerberos per l'autenticazione. E' presente anche una lista di controllo per autorizzare gli accessi. Supporta SSL e HTTPS.
Monitoraggio	Ha una UI Web per monitorare l'applicazione. Possiede anche una UI Web per controllare statistiche e i log sull'output di ogni job.	Fornisce numerose metriche per i nodi, accessibili via URL. Supporta anche monitoraggio di rete e isolamento per ogni container.	Ha una Web UI per il manager delle risorse (che fornisce metriche per il cluster) e per il manager del nodo(che fornisce informazioni per ogni nodo e i container attivi sul nodo).
Algoritmi disponibili	Classificazione, regressione, alberi di decisione, clustering...	Classificazione, regressione, alberi di decisione, clustering...	Classificazione, regressione, alberi di decisione, clustering...

I componenti per effettuare query SQL sono simili per la maggior parte delle caratteristiche, modelli di database, licenza, uso di SQL, concorrenza, e altre ancora. Ciò per cui si differenziano sostanzialmente è[6, 41]:

- il motivo per cui sono stati creati, ognuno dei 3 componenti è stato creato pensando ad un obiettivo diverso;
- la possibilità di avere modello di database a documenti per Drill;
- i linguaggi di programmazione supportati, differenti tra i vari componenti;
- il supporto a XML, la replicazione e la consistenza dei dati possibili solo per Hive;
- script lato server, non possibili solo in Spark SQL e si negli altri.

Tabella 3.3: Componenti per query SQL: Hive vs Drill vs Spark SQL

Nome	**Drill**	**Hive**	**Spark SQL**
Descrizione	Query engine SQL senza schema per Hadoop, NoSQL e Cloud Storage.	Software di data warehouse per fare query e gestire grandi dataset distribuiti.	Componente che sta su Spark Core per il processamento di dati strutturati
Modello primario di database	Documenti, DBMS Relazionale	DBMS Relazionale	DBMS Relazionale
Modelli aggiuntivi di database	Key-value	Key-value	Key-value
Licenza	Open Source	Open Source	Open Source
Sistemi operativi per il server	Tutti	Tutti OS con una VM Java	Tutti
Schema dei dati	Senza schema	Si	Si
Typing	Si	Si	Si
Supporto XML	No	Si, tramite libreria	No
Indici Secondari	No	Si	No
SQL	Compatibile a SQL 2003	Statements SQL-like e DDL	Statements SQL-like e DDL
API e altri metodi d'accesso	RESTful HTTP API ,JDBC,ODBC	JDBC, ODBC, Thrift	JDBC, ODBC
Linguaggi di programmazione	C++	C++, Java, Python, PHP	Java, Python, R, Scala
Script lato server	Funzioni definite dall'utente	Si	No

Trigger	No	No	No
Metodi di partiziona-mento	Sharding	Sharding	Si, tramite Spark Core
Metodi di replicazione	Nessuno	Fattore di replicazione selezionabile	Nessuno
Consistenza	No	Eventual Consistency	No
Chiavi esterne	No	No	No
Concetto di transazione	No	No	No
Concorrenza	Si	Si	Si
Durability	A seconda dell'origine dati sottostante	Si	Si
Computazione In-memory	A seconda dell'origine dati sottostante	No	No
Concetto di utente	A seconda dell'origine dati sottostante	Accesso per utenti, gruppi e ruoli	No

Infine, per quanto riguarda le librerie di machine learning, possiamo notare dalla tabella 3.4 che le caratteristiche simili sono solo i tipi di algoritmi che mettono a disposizione e il linguaggio di programmazione Python, oltre al tipo di dato double. Per il resto sono abbastanza diverse[37, 36]:

- il layout della memoria è contiguo in Scikit Learn (infatti lavora in RAM) e distribuito in MLlib;
- le performance variano in base alla quantità di dati da processare;
- MLlib offre anche Java e Scala come linguaggi di programmazione;
- Scikit Learn consente anche l'uso di tipi float.

Tabella 3.4: Librerie di Machine Learning: MLlib vs Scikit-Learn

Nome	**MLlib**	**Scikit Learn**
Layout memoria	Distribuita, densa/sparsa	Contigua, densa/sparsa
Performance	Un pò basse in presenza di pochi dati, alte in presenza di molti dati	Molto alte in presenza di pochi dati, più basse in presenza di molti dati
Tipi di dati	double	double, float
Linguaggi di programmazione	Scala, Java, Python	Python
Algoritmi disponibili	Classificazione, regressione, alberi di decisione, clustering...	Classificazione, regressione, alberi di decisione, clustering...

Capitolo 4

Machine Learning

4.1 Introduzione ai sistemi di Machine Learning

Recentemente, nel campo dell'analisi dei dati si sente spesso parlare di Machine Learning. Tale termine viene usato per indicare la capacità di una macchina di apprendere informazioni a partire da un primo insieme di dati per poi essere in grado di attuare autonomamente soluzioni per ogni nuova situazione.
L'apprendimento automatico, altro termine con cui viene identificato il machine learning, aiuta nell'analisi di grosse quantità di dati (es. Big Data), automatizzandone il processo e riducendo nettamente i tempi di analisi.
Anche se da anni si sono sviluppate tecniche di data mining, quest'ultime non sono tanto efficaci quanto la possibilità di eseguire degli algoritmi sui dati.
Si distinguono principalmente due gruppi di algoritmi di machine learning[22]:

- Supervised Learning, o apprendimento supervisionato
- Unsupervised Learning, o apprendimento non supervisionato

Supervised Learning

Nell'apprendimento supervisionato, il sistema cerca di apprendere conoscenza dagli esempi che vengono forniti in precedenza.
Dal punto di vista matematico, vengono forniti sia i dati di input che i corrispondenti dati di output, e l'algoritmo trova una funzione di mappatura tra i due insiemi.
I problemi di supervised learning possono essere raggruppati in:

- classificazione, se le variabili di output sono un gruppo o una categoria
- regressione, se le variabili di output sono dei valori reali

Unsupervised Learning

Nell'apprendimento non supervisionato, vengono forniti all'algoritmo solo i dati di input senza i corrispondenti dati di output.

L'algoritmo, non avendo i corrispondenti output, deve riuscire a trovarli da solo in base alle informazioni ricavate dai dati.
I problemi di unsupervised learning possono essere raggruppati in:

- associazione, se l'obiettivo è scoprire se esistono regole di associazione che descrivono grosse quantità di dati
- clustering, se l'obiettivo è scoprire se esistono raggruppamenti nei dati

4.2 Classificazione

Un campo di applicazione dell'apprendimento supervisionato è quello della classificazione, che permette di ricavare la classe di appartenenza di una nuova istanza, sulla base della conoscenza acquisita da un set di dati di "allenamento".
Compito della classificazione è quindi trovare una funzione che associa l'insieme di input a quello di output[2]. Le variabili di output sono chiamate etichette o categorie di classe.
Le variabili di input possono essere valori discreti o reali.
I modelli di classificazione possono anche predire valori continui come le probabilità di un dato di input di appartenere alle varie classi di output.
Esistono i seguenti tipi di classificazione[23]:

- classificazione multi-class
- classificazione binaria
- classificazione one-class
- classificazione multi-label

4.2.1 Classificazione multi-class

Si parla di classificazione multi-class quando si ha il problema di classificare un'istanza in un insieme di tre o più classi. Essa usa un approccio mutuamente esclusivo, parte dall'assunzione che ogni istanza può essere assegnata ad una sola classe[48].
Nei prossimi paragrafi saranno esaminati alcuni algoritmi di classificazione multi-class.

Decision Tree

Il Decision Tree è un metodo non parametrico di apprendimento supervisionato usato per classificazione e regressione[7].
In un'analisi decisionale, esso può essere utilizzato per rappresentare in maniera visiva ed esplicita le decisioni e il modo in cui vengono raggiunte[21]. La rappresentazione è basata su alberi, come suggerisce il nome.
La figura 4.1 rappresenta un esempio di albero decisionale. Tale albero viene disegnato dall'alto verso il basso.
Il nodo più in alto viene chiamato radice. I nodi intermedi rappresentano le condizioni in

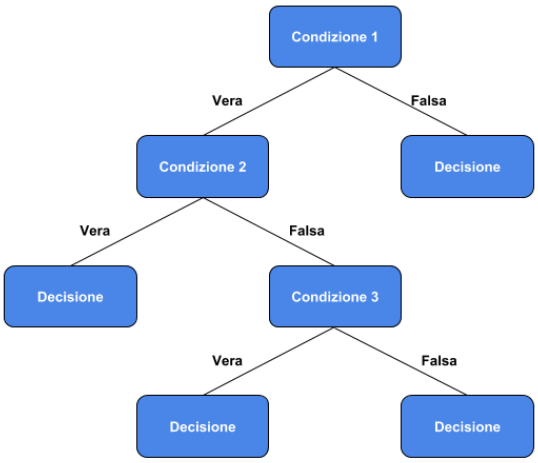


Figura 4.1: Decision Tree Generico

base al quale l'albero si divide in rami. Le foglie infine, ovvero i nodi terminali, contengono la decisione presa.
Considerando un set di dati reale, l'albero rappresentato in figura 4.1 ne sarà solo una piccola parte, data la presenza di una varietà molto più ampia di condizioni possibili.
L'obiettivo è creare un modello che predica il valore della variabile target apprendendo semplici regole di decisione derivate dalle caratteristiche dei dati.
Ecco alcuni dei vantaggi degli alberi di decisione:

- Semplici da capire e interpretare. Gli alberi sono visualizzabili.
- Richiedono una piccola preparazione dei dati.
- Il costo dell'usare l'albero è logaritmico nel numero dei punti usati per allenare l'albero.
- Capace di gestire dati numerici e categorici e anche più output.
- Possibile validare il modello usando test statistici.
- Lavora bene anche se le assunzioni sono violate dal vero modello dal quale i dati sono generati

Al contempo sono presenti anche molti svantaggi:

- L'algoritmo può creare alberi di decisione molto complessi che non generalizzano bene i dati. Questo fenomeno è chiamato overfitting.
- I decision tree possono essere instabili dato che piccole variazioni nei dati posso generare alberi completamente diversi (varianza).

- Alcuni concetti possono essere complicati da apprendere poichè i decision tree non li esprimono in maniera facile.
- Se c'è una classe dominante, l'algoritmo crea alberi di decisione parziali.

K-Nearest Neighbor

Il K-Nearest Neighbor (KNN) è un algoritmo di apprendimento non parametrico[27], cioè non fa ipotesi sulla distribuzione dei dati.
Non ha bisogno di generare un modello, ma tutti i dati di training sono usati nella fase di test. Per questo motivo la fase di training è rapida, ma quella di testing e lenta e costosa in termini di memoria e tempo. Nel caso peggiore, KNN ha bisogno di molto tempo per analizzare tutti i dati, il che richiede maggiore memoria per salvare i dati di training.
Nel K-Nearest Neighbor K rappresenta il numero dei campioni più vicini, e rappresenta il parametro chiave dell'algoritmo. K=1 rappresenta il caso più semplice(il più vicino). Nella figura 4.2 è rappresentato un esempio di applicazione del KNN con 2 classi (blu e

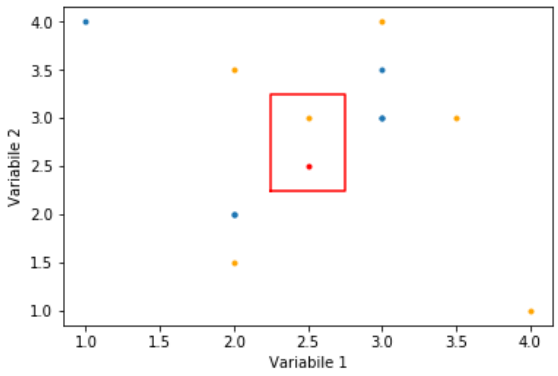


Figura 4.2: Esempio di KNN

arancione). L'algoritmo considera i K campioni (nel grafico K=1) più vicini alla nuova osservazione, che è rappresentata in rosso, e assegna ad essa la classe presente in maggioranza nei K vicini.
La distanza calcolata dall'algoritmo può essere:

- distanza euclidea, distanza nel piano tra 2 punti;
- distanza di Manhattan, per cui la distanza tra due punti è la somma del valore assoluto delle differenze delle loro coordinate;
- distanza di Minkowski, generalizzazione della distanza euclidea e di Manhattan.
- distanza di Hamming, distanza che si calcola tra due stringhe;

Alcuni vantaggi possono essere[33]:

- facilità di implementazione;

- maggiore velocità, dato che non richiede il training
- solo 2 parametri di configurazione;

Gli svantaggi invece sono:

- non è efficiente in presenza di molte dimensioni, dato che deve calcolare le distanze in tutte le dimensioni;
- alto costo computazionale per la predizione di dataset molto larghi;
- feature categoriche possono essere fonte di problemi nel calcolo delle distanze.

Naive Bayes

I classificatori Naive Bayes sono una famiglia di semplici classificatori probabilistici, basati sull'applicazione del teorema di Bayes con forti assunzioni di indipendenza tra le caratteristiche[31]. In altri termini, questi tipi di classificatori assumono che la presenza di una particolare feature (caratteristica) non sia correlata alla presenza delle altre.
Il teorema di Bayes, considerato c una delle classi e x il vettore di valori delle variabili presenti afferma che:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{4.1}$$

dove:

- P(c|x) è la probabilità condizionata della classe c, noto il vettore x;
- P(c) è la probabilità a priori di c(non si tiene conto di x);
- P(x|c) è la probabilità condizionata di x, nota la classe c;
- P(x) è la probabilità a priori di x.

Viene calcolata per ogni osservazione x la probabilità che appartenga a ciascuna delle classi presenti, e le viene assegnata l'etichetta di classe che ha la maggiore probabilità. Alcuni vantaggi di Naive Bayes sono:

- facilità e velocità di applicazione;
- se è rispettata la condizione di indipendenza delle variabili, si comporta meglio di altri algoritmi, anche con meno dati di training;
- funziona bene in caso di variabili categoriche.

Gli svantaggi invece sono:

- se una delle osservazioni appartiene a una categoria non presente in fase di training, le viene assegnata probabilità 0;
- non è precisissimo;
- avere l'ipotesi che le features sono indipendenti, poichè non si verifica quasi mai.

4.2.2 Classificazione binaria

Si parla di classificazione binaria quando si ha il problema di classificare in maniera esclusiva un dato set di dati in un insieme di soli due classi [45](es: bianco/nero).
Alcune applicazioni della classificazione binaria possono essere:

- test medici, per capire se un paziente ha una malattia o no;
- test di superamento (pass/fail), applicabile in vari contesti (es: controllo qualità);

Molti degli algoritmi per la classificazione binaria sono gli stessi della classificazione multiclass, ma applicati a 2 sole classi(Decision Tree, Naive Bayes ecc).

4.2.3 Classificazione one-class

Quando il problema di classificazione riguarda l'identificare una determinata istanza come appartenente ad una classe o no[49], si parla di classificazione one-class. L'effettivo output di questa classificazione è uno score di previsione, per cui discriminare se l'istanza appartiene alla classe positiva oppure no.
Applicazioni comuni per questo tipo di classificazione sono rilevamento di anomalie, di outlier e di novità.
Di seguito ecco una panoramica su alcuni algoritmi di classificazione one-class.

Isolation Forest

Isolation Forest è costruito sulla base degli alberi di decisione. Viene creato un insieme di alberi (foresta), scegliendo in maniera random le variabili e i loro valori.
L'insieme dei campioni in ogni nodo è diviso in due parti[12], in base alla variabile e ai suoi valori. Sia l'attributo che il valore in cui effettuare lo split sono scelti casualmente tra il valore minimo e il massimo per quell'attributo. Comunemente, i campioni anomali sono quelli più facili da isolare poichè ci sono meno condizioni necessarie per distinguerle dai casi normali e quindi avranno anche percorsi più brevi rispetto alle normali osservazioni, risiedendo vicino alla radice dell'albero. Per attenuare il gli effetti della casualità, si calcola la profondità media dell'anomalia nella foresta, ovvero la media delle profondità dei vari alberi per quell'osservazione. Questo valore viene utilizzato come punteggio per l'osservazione. Più il punteggio è basso, maggiore è la probabilità che quel campione sia anomalo[9].

Local Outlier Factor

Local Outlier Factor è un metodo di ricerca degli outlier non supervisionato, che misura la deviazione locale della densità di un'osservazione[10], rispetto ai suoi vicini. É locale (Local) poichè la misura dipende da quanto isolata sia l'osservazione. In particolare, la località è data da un numero fissato di osservazioni più vicine, la cui distanza è usata per stimare la densità locale.
L'algoritmo considera quindi outlier i campioni che hanno una densità sostanzialmente più

bassa rispetto ai vicini.
Un'osservazione è ritenuta normale se il punteggio ottenuto(chiamato Local Outlier Factor) è vicino a 1, mentre viene considerata anomala se è molto maggiore di 1.

OneClass Support Vector Machine

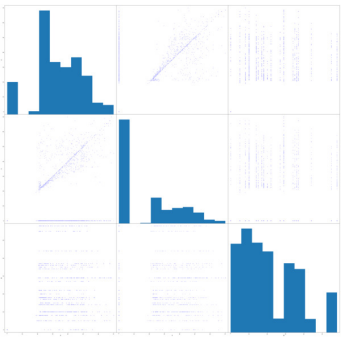
OneClass Support Vector Machine (o OneClass SVM) è un algoritmo non supervisionato che apprende una funzione di decisione per il rilevamento degli outlier. Esso classifica le nuove osservazioni come simili o diversi dal dataset di training.
Di base, nei classificatori SVM, le osservazioni sono rappresentate come punti dello spazio[39], separati da un piano, in base alla classe.
Le nuove osservazioni sono mappate nello stesso spazio e la loro posizione stabilisce a che classe appartengano.
Il classificatore OneClass SVM è un caso particolare di SVM nel quale si cerca di stabilire la posizione del nuovo dato rispetto alle osservazioni normali, ricavate tramite la fase di training, in modo da stabilire se è un outlier o no.

Scatter Matrix e Decision Function

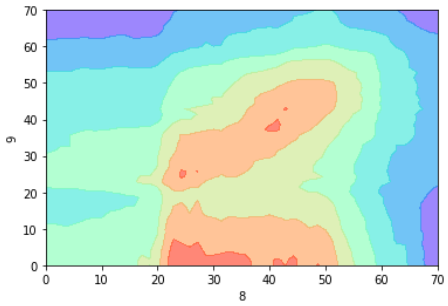
La Scatter Matrix (in figura 4.3a) è una matrice di grafici a dispersione (scatter). Questi ultimi sono tipi di grafici in cui due variabili sono riportate su uno spazio cartesiano. I dati sono riportati come punti, aventi come coordinate i valori delle due variabili.
La quasi totalità degli degli algoritmi consente l'uso di una funzione, la Decision Function, che restituisce in output lo score (punteggio), compreso tra -1 e 1, sull'appartenenza di un campione alla classe. Tale funzione, se calcolata per due variabili, consente di costruire un grafico nel quale sono tracciate le zone a diversa probabilità come mostrato in figura 4.3b.
Combinando la Scatter Matrix con la Decision Function, è possibile ricavare una matrice di grafici a dispersione nei quali sono tracciate le zone a diversa probabilità, come si può vedere in figura 4.3c.
Si traccia un grafico per ogni coppia di variabili. Nella diagonale, nella quale è presente il confronto tra variabili identiche, è tracciato l'istogramma della variabile, ovvero la rappresentazione grafica della distribuzione dei valori per la variabile.

4.2.4 Classificazione multi-label

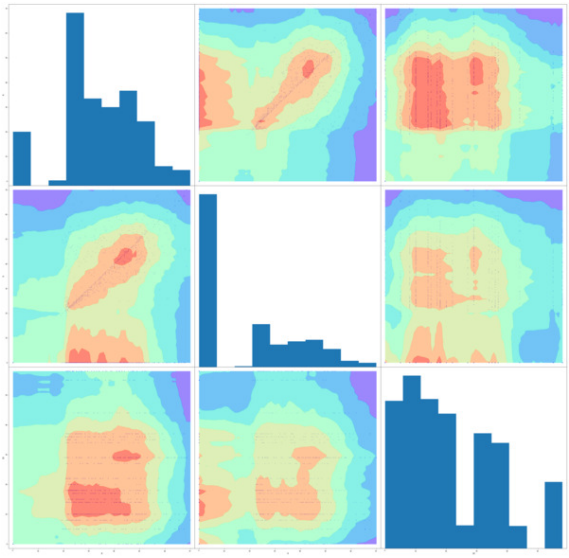
Similmente alla classificazione multi-class, anche la classificazione multi-label si occupa di associare un'istanza a un set di classi. La differenza fondamentale è che l'output può essere più di una classe. Un esempio potrebbe essere quello di associare ad un articolo o un libro delle parole chiave.
Un algoritmo usato per questo genere di classificazione è il MultiLabel K-Nearest Neighbor (Ml-KNN), che è una variante dell'algoritmo KNN per le classificazioni multi-label.



(a) Scatter Matrix Standard



(b) Zone di probabilità



(c) Scatter Matrix derivata

Figura 4.3: Grafici di Scatter Matrix e Decision Function

4.3 Modalità d'uso del calssificatore

Sono possibili 3 modalità d'uso per il classificatore:

- Su tutto il dataset
- Sulle singole caldaie
- Sulle caldaie dello stesso modello

La tabella 4.1 mostra i vantaggi e gli svantaggi dei modi di utilizzo indicati.

Tabella 4.1: Metodi di applicazione dell'algoritmo

Algoritmo	**Pro**	**Contro**
Su tutto il dataset	• analisi di tutto l'insieme dei dati in una singola operazione • possibile evitare alcuni falsi allarmi, se comportamenti simili si sono già verificati	• funzionamento corretto per una caldaia potrebbe non esserlo per un'altra • difficile identificare le caldaie anomale • parametri di configurazione delle caldaie potrebbero essere diversi • tempi di calcolo più lunghi
Sulle singole caldaie	• possibilità di dare priorità a determinate caldaie, rispetto ad altre • distinzione netta tra le caldaie anomale e le altre • tempi di calcolo più brevi	• ogni caldaia da analizzare separatamente • alcuni comportamenti anomali potrebbero rientrare invece nella norma di comportamento
Sulle caldaie dello stesso modello	• mix dei precedenti • analisi di caldaie con parametri uguali	• mix dei precedenti • necessarie altre variabili (id modello)

Capitolo 5

Sperimentazioni E Risultati Ottenuti

5.1 Raccolta di conoscenza da esperti di dominio

Per raggiungere l'obiettivo finale del lavoro di tesi, è stato necessaria una fase di raccolta di conoscenza da esperti di dominio.
È stata predisposta una descrizione di guasti comuni e relative informazioni di correlazione con i dati di funzionamento della caldaia, interrogando un esperto del settore.
Di seguito ne è riportata la lista:

- Pressione dell'acqua di mandata a temperatura definita.
  Indicando la pressione dell'acqua ad una certa temperatura dell'acqua di mandata con $p_{H_2O}(T)$, se
  $$p_{H_2O}(80°) > p_{H_2O}(20°) \times (1 + 200\%) \tag{5.1}$$
  c'è un problema di espansione dovuto a:
  - volume di espansione installato insufficiente
  - membrane dei vasi di espansione bucate
  - precarica dei vasi di espansione eccessiva
- Variazione della pressione nel tempo.
  Indicando con p_{H_2O} la pressione dell'acqua, se
  $$\frac{\partial p_{H_2O}}{\partial t} < 0 \tag{5.2}$$
  l'impianto, in assenza di carico automatico funzionante, può:
  - avere delle perdite
  - non avere perdite, ma avere correnti parassite

- Differenza di temperatura a potenza del bruciatore definita.
  Indicando con F il firing rate, ovvero la potenza istantanea del bruciatore (in percentuale), e con DT la differenza fra temperatura dell'acqua di mandata e temperatura dell'acqua di ritorno e supposto che a caldaia nuova la modulazione del circolatore insegua un DT definito, se

  $$\frac{\partial DT(F = 100\%)}{\partial t} > 0 \tag{5.3}$$

  allora significa che il flusso reale di acqua in caldaia sta diminuendo e si può avere che:

  - nello scambiatore principale si sta accumulando calcare
  - progressivo intasamento dei filtri d'acqua in ingresso caldaia (se presenti)

- Temperatura dei fumi a temperatura dell'acqua di ritorno costante.
  Indicando la funzione temperatura fumi come T_{fumi} e la temperatura dell'acqua di ritorno come T_{rit}, da:

  $$\frac{\partial T_{fumi}(T_{rit} = cost)}{\partial t} > 0 \tag{5.4}$$

  si può inferire quanto lo scambiatore principale si sta progressivamente sporcando e quindi intasando portando a una diminuzione del rendimento medio, diminuzione della portata termica e futura assoluta necessità di pulizia dello scambiatore dal "tartaro".

- Mancata accensione della fiamma.
  Indicando il numero di tentativi di accensione del bruciatore con N_{tent} e con N_{acc} il numero di accensioni riuscite, se:

  $$\partial \frac{\Sigma N_{acc}}{\Sigma N_{tent}} / \partial t < 0 \tag{5.5}$$

  vuol dire che il numero medio di tentativi di accensione riusciti sta decrescendo nel tempo, quindi si può avere che:

  - gli elettrodi di accensione sono sporchi e/o consumati
  - gli elettrodi di rilevazione sono sporchi e/o consumati
  - in presenza di elettrodi in buono stato, è molto probabile un'errata taratura di combustione

5.2 Analisi Esplorativa

Passo importante nel lavoro di tesi è stata la modellazione dei fenomeni di interesse a partire dai dati sperimentali.
Sono stati raccolti dataset di 10 caldaie reali, durante la loro operatività ordinaria, per un periodo pari a 4 mesi. Le variabili misurate sono pressione, temperatura, contatore di

accensioni, potenza del bruciatore e altre ancora, raccolte facendo uso di OpenTherm e ModBus, protocolli di comunicazione con le quali le caldaie scambiano dati con i gateway. L'elenco completo è mostrato nella seguente tabella.

Tabella 5.1: Elenco Variabili

ID Variabile	Nome Variabile	Unità Di Misura
07	Pressione Acqua Campionata	Bar
08	Temperatura Acqua Mandata Campionata	Gradi Celsius
09	Temperatura Acqua Ritorno Campionata	Gradi Celsius
10	Tentativi Accensione Campionati	Contatore
11	Tentativi Accensione Riuscite Campionati	Contatore
12	Pressione Acqua A Temperatura Acqua Mandata Massima	Bar
13	Pressione Acqua A Temperatura Acqua Mandata Minima	Bar
14	Temperatura Acqua Mandata A Firing Rate Definito	Gradi Celsius
15	Temperatura Acqua Ritorno A Firing Rate Definito	Gradi Celsius
16	Temperatura Fumi A Temperatura Acqua Ritorno Definita	Gradi Celsius
17	Temperatura Fumi Campionata	Gradi Celsius
18	Firing Rate Campionato	Percentuale

I dati campionati sono stati forniti separatamente per ogni caldaia, e nella seguente forma:

ID Variabile	Valore	Timestamp

È stato inoltre fornito un dataset contenenti i messaggi di errore delle caldaie di cui si sono raccolti i campionamenti.

5.2.1 Pulizia dei dati

La pulizia dei dati (Data Cleaning) è un processo fondamentale nell'analisi dei dati, dato che ha come obiettivo il garantire, con una soglia di affidabilità più alta possibile, la correttezza dei dati[14].
Tale processo è utile per la successiva applicazione degli algoritmi di machine learning, poichè può portare a risultati migliori.

Rimozione Duplicati

Primo passo nel data cleaning è l'identificazione e la rimozione di campioni non utili. Fra questi possono esserci i record duplicati e i campioni irrilevanti per l'algoritmo[14].
Durante l'analisi sono stati trovati valori duplicati, come si evince dalla tabella 5.2:

Tabella 5.2: Dati Duplicati

ID Variabile	Valore	Timestamp
7	1.8	1519879843
8	64.3	1519879843
7	1.8	1519879843
8	64.3	1519879843

Dati Mancanti

Altro passo della pulizia dei dati è la gestione dei dati mancanti. Dato che la maggior parte degli algoritmi di machine learning non accetta la presenza di valori mancanti, è necessario gestirli in qualche modo.
Esistono due modi più comuni di gestire questi valori[14]:

- eliminare le osservazioni con valori mancanti
- interpolare i valori mancanti in base ad altre osservazioni

Eliminare record però potrebbe non essere ottimale, dato che eliminandoli, vengono detratte informazioni.
E anche interpolare i valori mancanti potrebbe essere sub-ottimale, dato che essendo il valore mancante inizialmente, ciò porta lo stesso a una perdita di informazioni.
Preferire un approccio piuttosto che l'altro è influenzato dai motivi per i quali sono presenti i dati mancanti.
Se la motivazione per cui un dato è mancante è attribuita ad un errore di campionamento, oppure a un malfunzionamento di uno dei sistemi coinvolti nel procedimento di raccolta, è più saggio rimuovere l'informazione dall'insieme dei dati poichè potrebbe alterare l'analisi.
Al contrario, se la mancanza di un dato si imputa a tempi diversi di campionamento è opportuno riempire il valore mancante.
Nei dati forniti, visto che i dati sono tutti numerici, e considerato che le funzioni su cui svolgere l'analisi sono tutte in funzione del tempo, la scelta su come riempire i valori mancanti si è basata sul:

- sostituire il valore mancante con l'ultimo valore campionato in precedenza, in maniera da tenere costante il valore;
- sostituire il valore mancante con il primo valore successivo, se non esistono valori campionati in precedenza.

Filtraggio dei dati

Alcune caldaie sono state escluse dall'effettiva analisi, principalmente per i pochi dati raccolti, o per la presenza di grandi vuoti temporali (anche 1 mese a volte).
In particolare, sono stati selezionati i dati di 6 delle 10 caldaie fornite nel dataset, delle quali se ne riportano i codici identificativi:

- 0017A288;
- 0017A299;
- 0017AD16;
- 0017AD1E;
- 0017AD9D;
- 0017AD38.

Seppur i dati delle caldaie menzionate siano migliori di quelli delle caldaie epurate, anche in esse sono presenti valori mancanti. Tale mancanza è dovuta al diverso istante di campionamento di alcune variabili.
I vuoti presenti sono stati colmati con i valori campionati antecedenti a quelli mancanti (forward fill). In caso di assenza di valori antecedenti, il vuoto è stato colmato con il primo valore successivo al dato mancante (backward fill).

5.2.2 Creazione di dataset sintetici

I dati forniti per il test degli algoritmi presentano diversi svantaggi, tra quali:

- molti vuoti temporali nelle misure;
- scarsa presenza delle variabili di interesse;
- quantità esigua di dati forniti.

Tali condizioni possono portare a risultati errati da parte degli algoritmi e all'impossibilità di provare gli stessi algoritmi su grandi quantità di dati.
Per questi motivi, si è deciso di creare un algoritmo per sintetizzare dei dataset, sulla base delle informazioni ottenute dagli esperti di dominio e dei dati forniti.
Sono stati sintetizzati dataset per le sperimentazioni in locale, con diverse quantità di record, e un dataset molto grande da usare per le sperimentazioni su cluster.
Ecco la composizione di una riga del dataset sintetizzato:

Id Caldaia	v_1	v_2	...	v_n	Label	Timestamp

- Id Caldaia rappresenta un numero progressivo con il quale indicare univocamente una caldaia;

- v_1,v_2, v_n rappresentano la lista dei valori assunti da ogni variabile (vedi tabella 5.1);
- Label indica l'etichetta assegnata alla specifica riga, che può essere:
  - green (verde), che indica un campione sano;
  - yellow (giallo), che indica un campione con possibili problemi;
  - red (rosso), che indica un campione che porta sicuramente a un problema;
- Timestamp, che indica il timestamp della riga.

Tutti i dataset sono stati generati con una percetuale del 90% di etichette green, ed un 5% di etichette yellow e red.
Per il test degli algoritmi in locale sono stati sintetizzati 3 file, come è possibile osservare nella tabella 5.3.
Seguendo il modello dei campioni ricevuti, è stata creata 1 entry ogni ora (24 in un giorno). I file generati sono composti da 10 mila caldaie, con i valori di 2 settimane (2w) e di 1 anno(1y), e 100 mila caldaie, con i valori di 2 settimane. Non sono state fatte prove con 1 milione di caldaie e con 100 mila caldaie e 1 anno di entries causa la dimensione esagerata dei file (dai 20GB in su).
Essendo creati ad hoc, i dataset non presentano problemi di duplicati, o letture errate o ancora valori nulli.

Tabella 5.3: Tabella file di test in locale

#Caldaie (entries)	**Dim File**	**Entries**
10k (2w)	198MB	3360000
100k (2w)	2GB	33600000
10k (1y)	5,2GB	87600000

Similmente, anche per il test dell'algoritmo su cluster è stato sintetizzato un file (vedi tabella 5.4).
Il file, della dimensione di 240GB, è composto da un'entry ogni ora per 6 mesi (6m) di campionamenti, e da un numero di caldaie pari a 1 milione.

Tabella 5.4: Tabella file di test su cluster

#Caldaie (entries)	**Dim File**	**Entries**
1M (6m)	240GB	4320M

É stato sintetizzato un ulteriore dataset, utilizzato per fare il training del classificatore. Esso è composto da un numero di caldaie pari a 1000 e 2 settimane di campionamenti(uno ogni ora), come mostrato nella tabella 5.5.

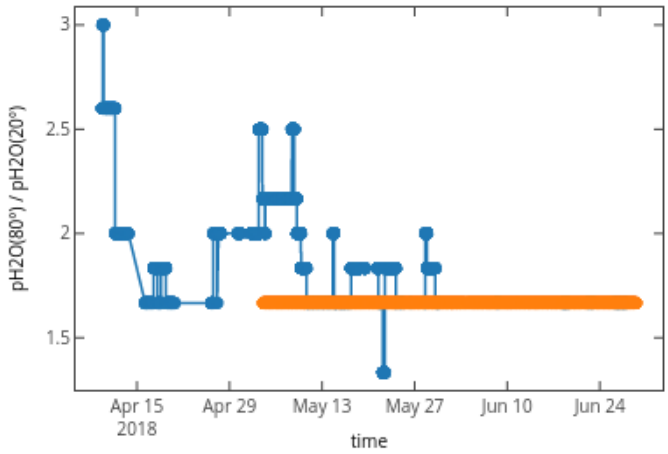
Tabella 5.5: Tabella file per il training del classificatore

#Caldaie (entries)	Dim File	Entries
1k (2w)	19,7MB	336000

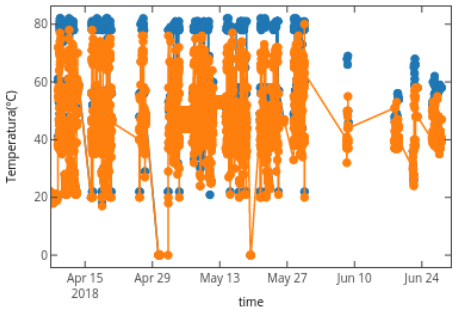
5.2.3 Analisi dei guasti

Dopo aver effettuato il data cleaning dei dati, è stata svolta un'analisi dell'andamento delle variabili nel tempo, per verificare se i comportamenti anomali da indirizzare siano presenti nel dataset fornito.

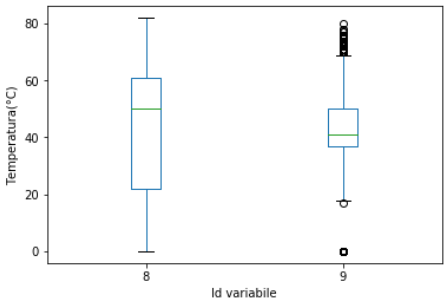
Pressione dell'acqua di mandata a temperatura definita (formula 5.1)



(a) Rapporto tra le pressioni a 80° e 20° nel tempo



(b) Andamento temperature nel tempo



(c) Distribuzione temperature

Figura 5.1: Grafici della pressione e della temperatura dell'acqua

Il grafico 5.1a mostra l'andamendo del rapporto $p_{H_2O}(80°)$ / $p_{H_2O}(20°)$ nel tempo per la caldaia 0017AD9D(in blu) e per la caldaia 0017A299(in arancione). Si può notare che per la prima caldaia, il rapporto si aggira inizialmente intorno al valore limite, per poi assestarsi definitivamente al di sotto, ma senza mai superarlo. Per la seconda invece, il rapporto è definito solo per una porzione del periodo di osservazione, data la presenza delle 2 variabili solo a partire dal 4 maggio in poi.
Il grafico 5.1b invece, mostra l'andamento delle temperature di mandata (in blu, variabile 8) e della temperatura di ritorno (in arancione, variabile 9) in funzione del tempo nel periodo di osservazione del primo grafico per la caldaia 0017AD9D. Da notare che la presenza di valori nel primo grafico è dovuta alla presenza di temperature nel grafico centrale intorno agli 80° per la temperatura di mandata e intorno ai 20° per la temperatura di ritorno.
Infine, il grafico 5.1c mostra la distribuzione di valori delle precedenti temperature per la caldaia 0017AD9D. Si può notare che la temperatura di mandata si assesta prevalentemente tra i 24 e i 62 gradi, ma raggiuge anche valori alti (82°) e bassi (vicini a 0°), mentre la temperatura di ritorno si assesta per lo più tra i 35 e i 50 gradi raggiungendo valori bassi (intorno ai 20°), ma pochissime volte si avvicina agli 80°.
Si deduce dai dati in nostro possesso, che il rapporto non supera mai la soglia stabilita dalla 5.1, e spesso il rapporto non è neanche definito per alcune caldaie o per periodi (anche lunghi) di tempo.
Tuttavia, il caso d'uso può essere coperto per i periodi in cui sono presenti sufficienti campionamenti.

Variazione della pressione nel tempo (formula 5.2)

Il grafico 5.2 mostra la pressione in funzione del tempo per le caldaie 0017AD9D (in blu) e 0017AD1E (in arancione).
Si può notare che in alcune zone la pressione ha un andamento decrescente per entrambe le caldaie, per cui rientra nello use case preso in esame. Per la caldaia 0017AD9D, nel periodo tra il 29 marzo e il 5 aprile la funzione è decrescente, e il 5 aprile viene segnalato un errore di codice 1. Negli altri periodi in cui è decrescente, non sono segnalati errori. Per l'altra caldaia viene segnalato un errore 112 il 5 giugno, ma mai negli altri momenti in cui la pressione decresce.
Da notare anche il valore elevatissimo (tra 4.6 e 4.8) iniziale per la caldaia 0017AD1E, che decresce rapidamente tra il 14 e il 16 marzo. Dai dati in nostro possesso, è visibile che la funzione che rappresenta la pressione è spesso decrescente. A volte, a tale andamento, corrisponde un errore, ma con codice diverso.
Lo use case è al momento parzialmente utilizzabile, dato che potrebbe funzionare bene in caso di campionamenti con cadenza costante, ma in caso di vuoto nelle misure o di valori esageratamente alti potrebbero esserci dei falsi positivi.

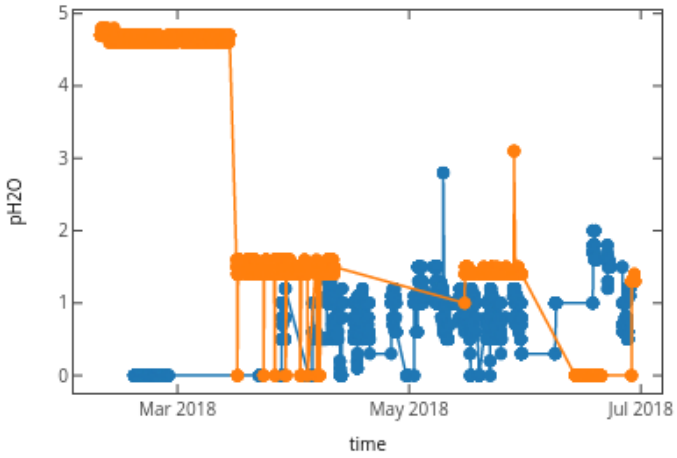
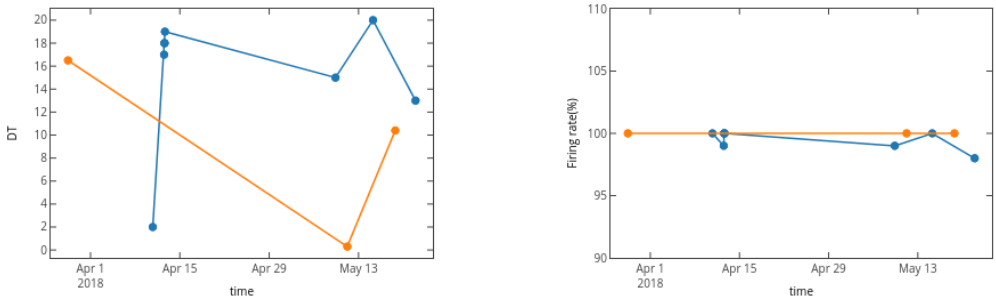


Figura 5.2: Variazione della pressione nel tempo

Differenza di temperatura a potenza del bruciatore definita (formula 5.3)

I grafici 5.3a e 5.3b mostrano l'andamento della Differenza di temperatura tra mandata e ritorno e il firing rate nelle caldaie 0017AD9D (in blu) e 0017A299 (in arancione) in funzione del tempo. Da notare che in 2 punti del grafico per la prima caldaia e in un punto per la seconda, la DT è crescente, per cui rientra nello use case di interesse.
Sono segnalati degli errori (codice 1) per la prima caldaia nel periodo tra il 10 e il 12 aprile e il 10 maggio. Al contrario, per la seconda caldaia non sono segnalati errori nel periodo di interesse.



(a) DT nel tempo

(b) Firing rate nel tempo

Figura 5.3: Grafici della differenza di temperatura e del firing rate

È osservabile nei dati in nostro possesso, che la funzione che rappresenta la differenza di temperatura a F massimo è a volte decrescente. A tale andamento, corrisponde un errore nel caso della prima caldaia.
Lo use case è al momento parzialmente utilizzabile, dato che potrebbe funzionare bene

in caso di campionamenti con cadenza più alta, ma in caso di grandi vuoti nelle misure potrebbero esserci dei falsi positivi (seconda caldaia).

Temperatura fumi a temperatura dell'acqua di ritorno costante (formula 5.4)

Il grafico 5.4 mostra l'andamento della temperatura fumi, a temperatura di ritorno costante, in funzione del tempo. per le caldaie 0017A299(in blu) e 0017AD9D (in arancione). Rientra nel caso di interesse l'andamento crescente della funzione per la prima caldaia,

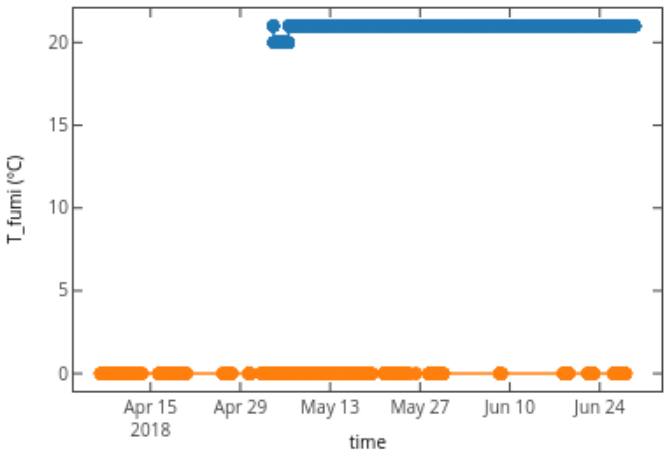


Figura 5.4: Grafico della temperatura fumi a temperatura di ritorno costante

durante il 6 maggio, al quale però, non corrisponde nessun codice di errore. L'andamento nella seconda caldaia è costante dato che l'unico valore assunto è 0. Quest'ultimo comportamento è riscontrato anche in altre caldaie.
Nei dati in nostro possesso, la funzione che rappresenta la temperatura fumi, a temperatura di ritorno costante, è crescente solo 1 volta, nel caso in esame. Ma, a tale andamento, non corrisponde nessun errore.
Lo use case è tuttavia utilizzabile, ma soltanto in caso di campionamenti con cadenza più alta.

Mancata accensione della fiamma (formula 5.5)

Nel grafico 5.5 è mostrato il rapporto tra Tentativi di accensione riusciti e i Tentativi di accensione campionati in funzione del tempo, per le caldaie 0017AD16 (in blu) e 0017A299 (in arancione). Quando le variabili 10 e 11 hanno entrambe valore 0, la funzione non viene tracciata, dato che non vengono svolti tentativi di accensione.
Il primo è l'unico caso in cui il rapporto è minore di 1, dato che sono presenti 2 tentativi di accensione falliti dal 10 al 17 febbraio. Esso non rientra però nel caso di studio, dato che la funzione assume solo quel valore per tutto il periodo, senza crescere/descrescere. Motivo per cui lo use case non si attiva. Nei dati in nostro possesso, la funzione che rappresenta

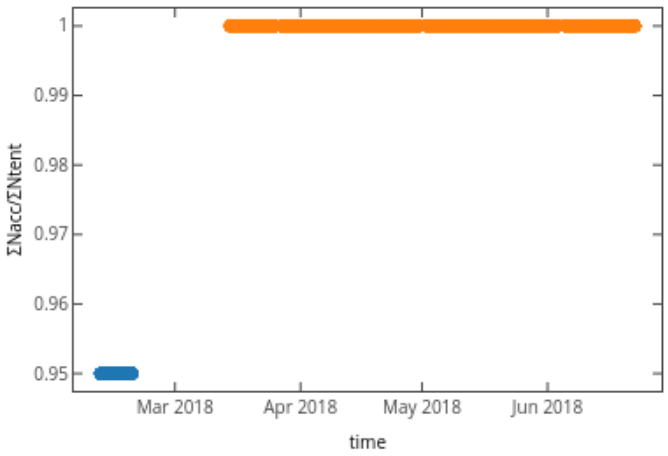


Figura 5.5: Grafico del rapporto tra numero di accensioni riuscite e numero di tentativi

il rapporto tra Tentativi di accensione riusciti e i Tentativi di accensione campionati, non è mai decrescente.
Lo use case è al momento parzialmente utilizzabile. Con i dati attuali infatti, lo use case non si attiva mai.
Sarebbe applicabile solo se le variabili fossero campionate con cadenza costante. I valori non campionati (NULL) non rappresentano un problema.

5.3 Sperimentazioni Decision Tree

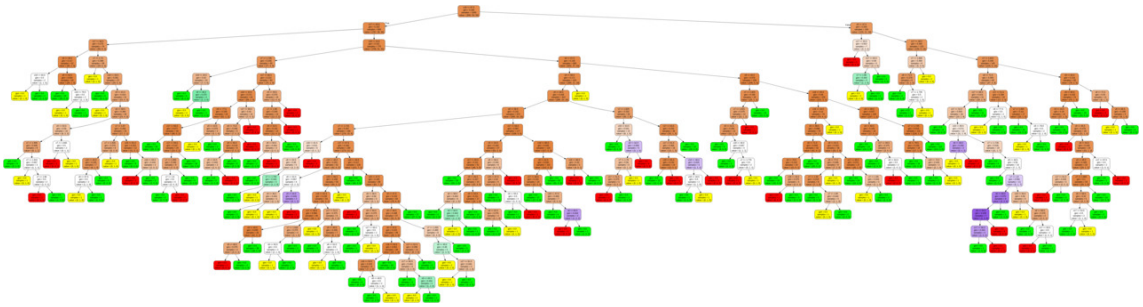
Il passo successivo è stato analizzare i dati (prevalentemente quelli sintetici), con l'algoritmo multi-class Decision Tree.
Questa parte del lavoro di tesi è focalizzata sulle prestazioni, dato che l'analisi ha anche l'obiettivo di scalare al milione di caldaie come citato nel capitolo introduttivo.
Per raggiungere tale fine l'algoritmo è stato quindi testato:

- localmente;
- su cluster, messo a disposizione dal Centro interdipartimentale SmartData@Polito.

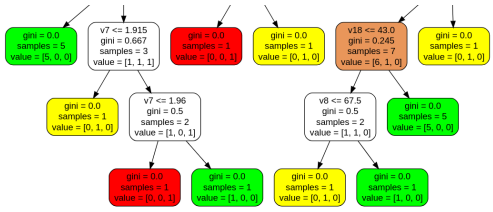
5.3.1 Generazione modello

É stato generato un solo modello di Decision Tree per la libreria Scikit-Learn, sia per le sperimentazioni in locale che quelle su cluster.
Il modello è stato generato fornendo come dataset di training un altro dataset sintetico, con le entries di 1000 caldaie, e 2 settimane di campionamenti (vedi sezione 5.2.2). Dato che il modello creato dall'algoritmo è visualizzabile come albero, ne è stata stampata la rappresentazione (figura 5.11).

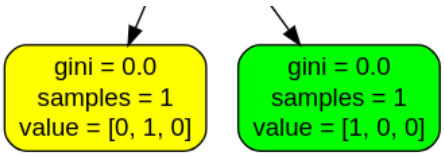
In figura 5.6a è rappresentato l'albero di decisione generato completo.
Si può notare che l'albero è molto largo ed ha una profondità abbastanza alta. Ciò è dovuto al fatto che il numero di variabili fornite è grande, e che il dataset è generato in maniera random, seppur segua delle regole realistiche.
Le foglie dell'albero, colorate in verde, rosso e giallo, rappresentano i punti in cui i campioni hanno una sola etichetta come risultato possibile.
E' naturalmente prevalente la presenza di nodi verdi, dato che il dataset si compone per il 90% di campioni con etichette verdi.
Analizzando più da vicino una parte dell'albero e le foglie stesse (figure 5.6b e 5.6c) si



(a) Albero di decisione completo



(b) Zoom su una parte dell'albero



(c) Zoom su due foglie

Figura 5.6: Decision Tree generato per i test con Scikit Learn

può osservare che la condizione di separazione dell'albero è il valore delle variabili.
In ogni nodo dell'albero è indicato:

- Condizione di separazione (tranne sulle foglie);
- Indice gini, che è una stima della qualità dello split in quel nodo;
- Samples, che rappresenta il numero di campioni che il nodo rappresenta;
- Value, che rappresenta il modo in cui sono suddivisi i campioni tra le etichette (green, yellow, red).

Per la libreria MLlib, si è generato un albero in maniera diversa (tramite codice Java), poichè i formati degli alberi tra la libreria MLlib e Scikit-Learn sono differenti. Per tale albero non si fornisce una rappresentazione grafica, ma è lecito supporre che sia verosimile a quella presente in figura 5.11.

5.3.2 Esperimenti in locale

La sperimentazione dell'algoritmo Decision Tree si è svolta in principio localmente.
Obiettivi di tale sperimentazione è valutare le performance dell'algoritmo sulla propria macchina, esplorando alcuni ambienti di sviluppo disponibili e le varie possibilità di utilizzo.

Condizioni di utilizzo dell'algoritmo

Sono state effettuate sperimentazioni in quattro diversi ambienti di sviluppo:

- Spark in ambiente Java e libreria Scikit Learn per il ML;
- Spark in ambiente Java con la libreria MLlib per il ML.
- Spark in ambiente Python e libreria Scikit Learn per il ML;
- Python e libreria Scikit Learn per il ML.

Gli algoritmi in ambiente Spark sono stati eseguiti con numero di worker e numero di partizioni impostato a 2.
Python in locale è stato eseguito con numero di thread pari a 1.
Inoltre, l'algoritmo è stato testato in 2 casi differenti, e sono state prese le medie su 10 esecuzioni del programma (in caso di troppe entries le esecuzioni sono state minori):

- Su una frazione del dataset composta dagli ultimi campioni presenti (timestamp più recente per ogni caldaia);
- Su tutto il dataset (tutti i timestamp di ogni caldaia).

Risultati

Nelle tabelle 5.6 sono riportati i risultati dell'applicazione dell'algoritmo.

L'esecuzione in ambiente Python è molto rapida in caso di basso numero di entries. Con l'aumentare del numero di entries, però diventa sempre più inefficiente, superando anche le 3 ore di esecuzione per il file più grande. Discorso opposto per Spark con Mllib (ambiente java), che all'aumentare del numero di entries, dimostra nettamente una migliore efficienza paragonato agli altri. I rimanenti 2 algoritmi (Spark su python e su java con utilizzo di scikit) si presentano più efficienti di python all'aumentare del numero di entries analizzate, ma nettamente meno efficienti in caso di basso numero di entries.

5.3.3 Esperimenti su cluster

Le prove con Spark in ambiente Java hanno beneficiato dell'uso del cluster messo a disposizione del Centro interdipartimentale SmartData@Polito, che ha permesso di verificare l'efficienza e la scalabilità in presenza di grandi quantità di dati.
Tale cluster può contare su un totale di 33 nodi, così suddivisi:

Tabella 5.6: Risultati sperimentazioni Decision Tree in locale

(a) Algoritmo su una frazione del dataset composta dagli ultimi campioni presenti (timestamp più recente per ogni caldaia)

#Caldaie (entries)	Spark su Java & Scikit	Python & Scikit	Spark su Python & Scikit	Spark su Java & Mllib
10k (2w)	27,528s	3,125s	20,247	15,787s
100k (2w)	170,618s	31,174s	151,38	59,260s
10k (1y)	374,822s	5014,10s	344,329	127,471s

(b) Algoritmo su tutto il dataset (tutti i timestamp di ogni caldaia)

#Caldaie (entries)	Spark su Java & Scikit	Python & Scikit	Spark su Python & Scikit	Spark su Java & Mllib
10k (2w)	218,134s	6,868s	363,246s	26,552s
100k (2w)*	2104,576s	64,611s	3207,97s	149,104s
10k (1y)*	4983,158s	11457,571s	8173,775s	361,717s

- 3 nodi Master DELL PowerEdge R620;
- 18 nodi Worker DELL PowerEdge R720XD;
- 2 nodi Worker SuperMicro dal cluster DET;
- 10 nodi Worker SuperMicro dal cluster DET.

In ambito software su ogni nodo del cluster è in esecuzione una distribuzione Cloudera su Ubuntu, che è basata sul framework Hadoop per applicazioni Big Data distribuite. Gli ambienti di gestione inclusi sono HDFS e YARN per la gestione dei dati, e Spark per l'accesso agli stessi. Sono supportati e integrati molti altri framework sia per l'accesso ai dati che per altri servizi, quali Hive, Pig, Falcon e molti altri.
Questa fase del lavoro è focalizzata sulla scalabilità. Come accennato in precedenza nel capitolo 1, uno degli obiettivi è che l'algoritmo sia scalabile, in modo da gestire bene un numero elevato di caldaie, e conseguentemente una quantità di dati elevatissima.
Per la sperimentazione dell'algoritmo su cluster, come accennato nella sezione 5.2.2, è stato utilizzato un unico file, con numero di righe pari a 6 mesi di campionamento(1 entry ogni ora) e 1 milione di caldaie.

Condizioni di utilizzo dell'algoritmo

Si sono effettuate sperimentazioni in due diversi ambienti di sviluppo, come detto in precedenza:

- Spark in ambiente Java e libreria Scikit Learn per il ML;

- Spark in ambiente Java con la libreria MLlib per il ML.

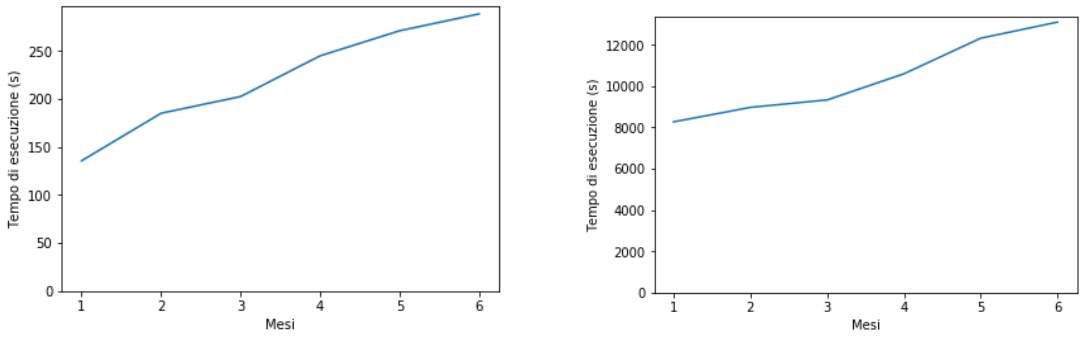
Gli algoritmi sono stati eseguiti al variare:

- del numero di mesi, con:
  - scheduler yarn;
  - numero di core per executor standard (1);
  - numero di executor standard (2);
  - memoria degli executor fissata a 4GB.
- del numero di core, su tutto il dataset con:
  - scheduler yarn;
  - numero di core per executor fissato a 4, a variare è il numero di executor;
  - memoria degli executor fissata a 4GB.

Come tempo di esecuzione viene considerato il tempo che è impiegato per terminare il job, che comprende la lettura e il cleaning dei dati, l'applicazione dell'algoritmo Decision Tree, e infine il salvataggio dei dati.

Risultati dell'applicazione dell'algoritmo

Nella figura 5.7 sono riportati i grafici dei tempi di esecuzione delle librerie MLlib e Scikit riguardanti i tempi di esecuzione in funzione dei mesi di campionamenti. Tali grafici mettono in risalto la scalabilità verticale dei due algoritmi.



(a) Scalabilità verticale MLlib

(b) Scalabilità verticale Scikit-Learn

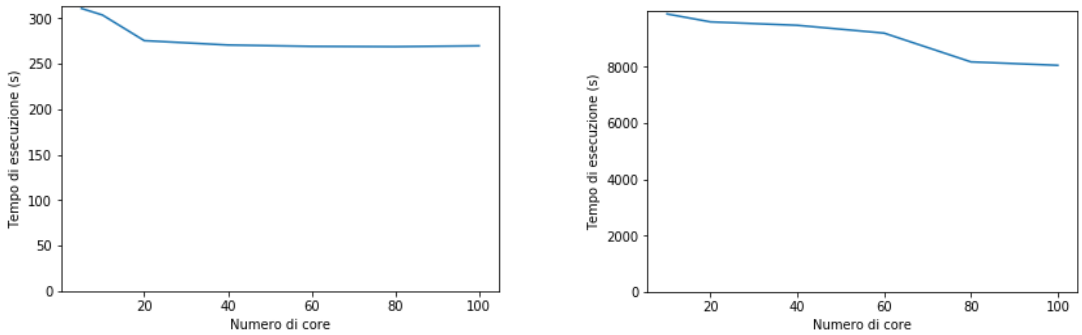
Figura 5.7: Grafici di scalabilità verticale

Si nota subito che MLlib ha tempi di esecuzione molto più bassi rispetto a quelli di Scikit-Learn (quasi 10 volte più piccoli).
Questa disparità è dovuta al processo che esegue il codice Python per la libreria Scikit-Learn. In particolare, sono 2 gli effetti negativi che la creazione di tali processi comporta:

- occupazione della memoria nel nodo, dato che ogni processo ha un suo spazio in memoria;

- Spark deve attendere il completamento di processi esterni per poter proseguire, dato che ne attende l'output.

MLlib si dimostra scalabile verticalmente, soprattutto dopo i 4 mesi di campionamento, poichè l'aumento della quantità di dati non comporta grandi aumenti nei tempi di esecuzione.
Scikit risulta poco efficiente con l'aumentare della quantità dei dati e poco scalabile verticalmente, dato che l'aumentare della quantità di dati genera sempre un aumento non trascurabile del tempo di esecuzione.
In figura 5.8 sono mostrati i grafici per MLlib e Scikit Learn dei tempi di esecuzione in funzione dei core impiegati nel calcolo. I grafici mostrano la scalabilità orizzontale dei 2 algoritmi.



(a) Scalabilità orizzontale MLlib

(b) Scalabilità orizzontale Scikit-Learn

Figura 5.8: Grafici di scalabilità orizzontale

Nel caso in questione MLlib si dimostra scalabile orizzontalmente solo inizialmente, poichè l'aumento della quantità di executor è efficace fino a raggiungere i 20 core, dopodiché ulteriori aumenti non comportano cospicue diminuzioni dei tempi di esecuzione.
Scikit, differentemente da prima, si dimostra scalabile orizzontalmente, con diminuzioni evidenti dei tempi di esecuzione.

5.4 Sperimentazioni Isolation Forest

Il lavoro di tesi si focalizza ora sulla sperimentazione dell'algoritmo Isolation Forest.
Gli esperimenti vengono svolti solo localmente, principalmente per 2 motivi:

- non si possiedono grandi quantità di dati da poter sottoporre all'algoritmo;

- non è possibile sintetizzare dataset pressochè verosimili a quelli reali.

I dati forniti sono relativi a 4 mesi di campionamento, e di sole 10 caldaie. Inoltre, come detto nella sezione 5.2, non è plausibile utilizzare tutte le caldaie a causa dei vari problemi che hanno alcune di esse.
La creazione di dataset sintetici non è presa in considerazione a causa della randomicità dei dati generati che causerebbe problemi al classificatore nella corretta individuazione di anomalie.

5.4.1 Esperimenti in locale

Per il test dell'algoritmo, sono quindi stati utilizzati i dati delle 6 caldaie reali, selezionate durante la fase di filtraggio, svolgendo test:

- separatamente per ogni caldaia;
- su tutto il dataset.

Condizioni di utilizzo dell'algoritmo

L'algoritmo è stato eseguito in un solo ambiente di sviluppo:

- Python con libreria Scikit-Learn per il Machine Learning.

Il numero di thread impiegati nella computazione è pari a 1.

Risultati

L'applicazione dell'algoritmo one-class sui nuovi dati ha evidenziato la presenza di numerosi falsi positivi:

- sia nell'intorno temporale dell'errore (segnalazione);
- sia nei periodi in cui non sono presenti errori etichettati.

5.4.2 Scatter Matrix

Per contrastare il comportamento indesiderato generato dal classificatore one class, è stata quindi svolta un'analisi sulla probabilità che un campione nel dataset sia segnalato come un errore (anomalia) dall'algoritmo, utilizzando le scatter matrix derivanti dalla decision function dell'algoritmo, che prende in esame le variabili a coppie.
L'analisi è possibile poichè la decision function dell'algoritmo one-class fornisce come output le probabilità dei campioni di essere etichettati come anomalie.
La figura 5.9 mostra un esempio di scatter matrix, composta dai grafici tra le variabili presenti nel dataset. Nella diagonale, dove si ha la corrispondenza della stessa variabile, sono presenti gli istogrammi delle variabili, cioè il grafico di come sono distribuiti i valori per la suddetta variabile. Nella figura 5.10 è mostrato un esempio di possibile grafico presente nella scatter matrix. In particolare, è mostrato il grafico tra la pressione(bar, in ascissa) e la temperatura dell'acqua di mandata (°C, in ordinata). La zona in cui il cam-

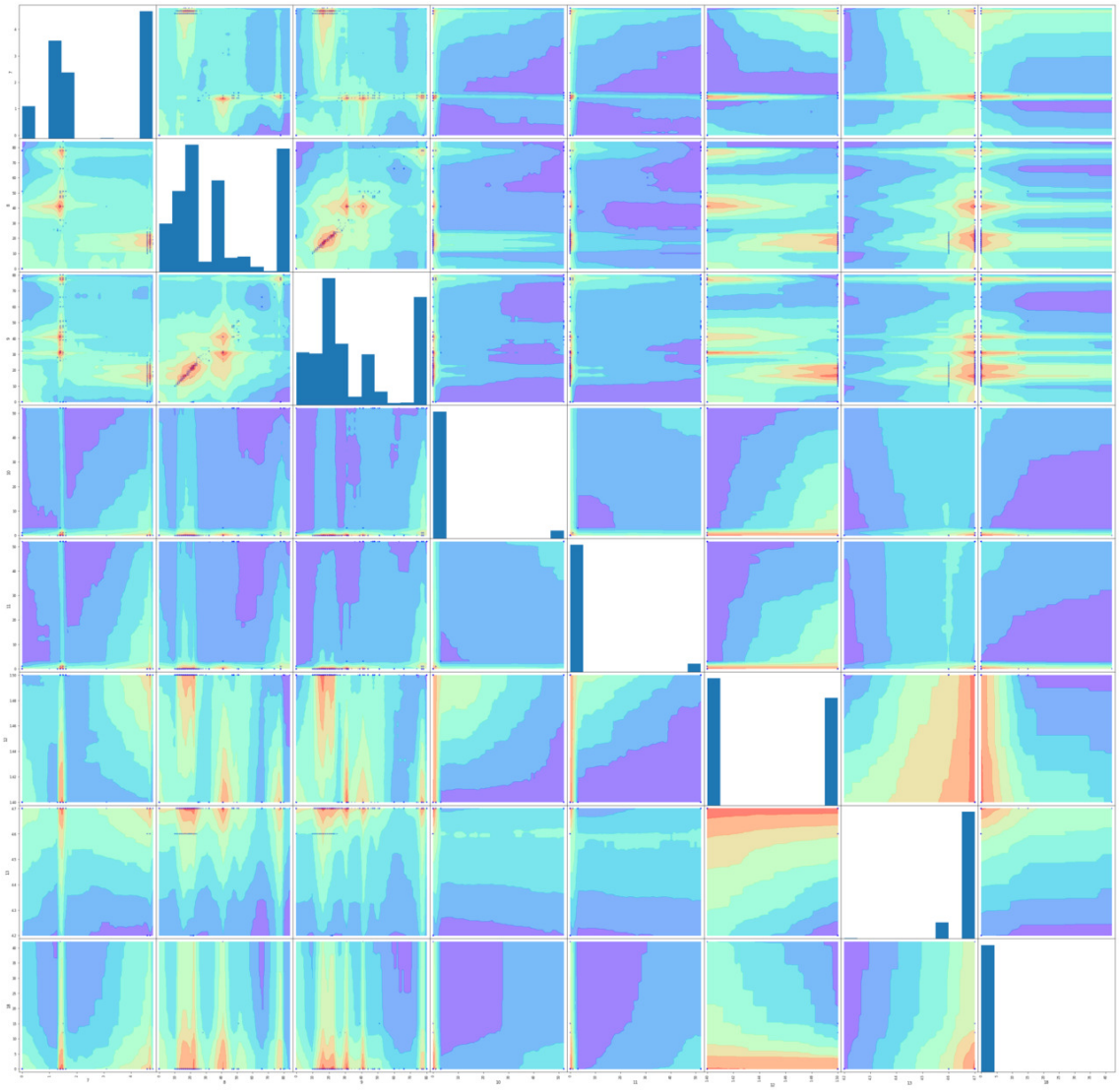


Figura 5.9: Esempio di Scatter Matrix

pione ha maggiore probabilità di essere etichettato come buono è quella di colore rosso. Al contrario, il campione ha maggiori probabilità di essere considerato un'anomalia nella zona di colore viola. Nelle zone che si estendono tra le due nominate in precedenza sono invece situati i campioni con probabilità intermedie.
Tali grafici permettono di valutare la probabilità che un campione sia un'anomalia e di comportarsi di conseguenza.
Da questa analisi è stato osservato che in corrispondenza delle zone a maggiore probabilità, si ha effettivamente riscontro con errori reali, verificatisi per la caldaia in questione (veri positivi).

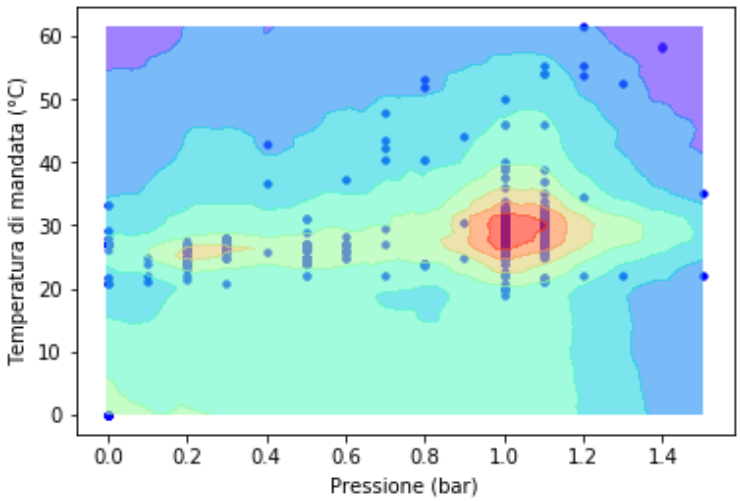


Figura 5.10: Esempio di grafico della Scatter Matrix

5.4.3 Isolation Forest: esempio d'uso del classificatore

Il processo di utilizzo del classificatore è articolato in varie fasi:

- Manipolazione dei dati.
  I dati vengono organizzati in modo da poter essere processati dall'algoritmo.
  Nella tabella 5.7a è rappresentato il modo in cui vengono inizialmente forniti i dati.
  In seguito, i dati vengono messi nella forma mostrata nella tabella 5.7b, in modo da essere pronti per essere processati dal classificatore.

- Scatter Matrix.
  Dai dati viene ricavata la scatter matrix, per tutte le variabili presenti (5.11a).

- Analisi dei campioni.
  Vengono individuati i campioni che hanno maggiore probabilità di essere etichettati come anomalie. Analizzando ogni grafico (5.11b) della scatter matrix, si individuano quelli con campioni situati nella zona a maggiore probabilità (viola).

- Corrispondenze nel dataset.
  Vengono trovate le corrispondenze nel dataset delle possibili anomalie trovate durante la fase precedente (5.8).

- Anomalia.
  Le corrispondenze trovate corrispondono a dei possibili futuri errori e/o guasti, per cui le caldaie in cui si sono verificate vengono segnalate come "a rischio guasto".

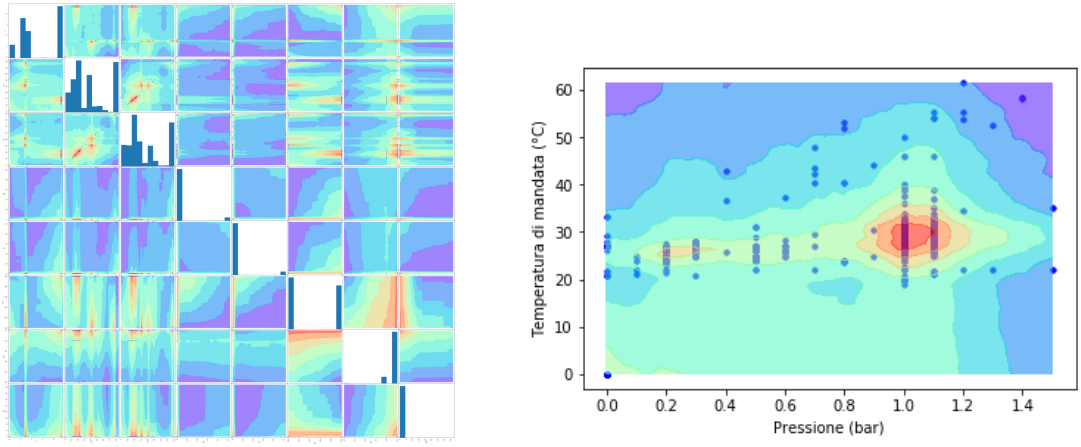
Tabella 5.7: Forme delle tabelle utilizzate

(a) Tabella estratta dal dataset

Id Variabile	Valore	Timestamp
7	1.5	1525349360
8	35.0	1525349360
9	25.2	1525349360
10	5.0	1525349360

(b) Tabella da passare al classificatore

Timestamp	V_7	V_8	V_9	V_{10}	V_{11}	V_{13}	V_{17}	V_{18}
04-02-2018 5:11:12	1.5	35.0	25.2	1.0	1.0	14.0	32.0	80.0



(a) Scatter Matrix generata

(b) Individuazione Campione Anomalo

Figura 5.11: Esempio di uso del classificatore

5.5 Analisi Veri E Falsi Positivi

Infine, è stata svolta un'analisi su tutte le anomalie identificate dal classificatore one-class e dall'analisi esplorativa dei guasti, distinguendole in veri e falsi positivi, per valutare la bontà del classificatore.
Essendo stato fornito un dataset aggiuntivo composto dagli errori verificatisi per le caldaie, è stato possibile capire quando comportamenti o campioni anomali possono condurre in errore la caldaia e quindi portare ad un possibile futuro guasto.
Non bisogna però confondere i due concetti:

- un errore è un comportamento anomalo riscontrato dalla caldaia stessa e segnalato tramite apposito messaggio;
- un guasto è un evento dannoso che compromette il regolare funzionamento del sistema (la caldaia).

Tabella 5.8: Tabella individuazione anomalie

Timestamp	V_7	V_8	V_9	V_{10}	V_{11}	V_{13}	V_{17}	V_{18}
07-06-2018 15:31:22	1.4	58.6	55.2	0.0	0.0	1.0	0.0	0.0
07-06-2018 15:32:20	1.4	48.6	23.6	1.0	1.0	1.0	20.0	53.0

Nel nostro caso, il dataset fornito è associato ai messaggi di errore delle caldaie, che non è detto portino a successivi guasti.
É bene quindi capire cosa si intende nell'analisi per veri e falsi positivi:

- vero positivo indica che un campione segnalato come anomalo risiede nell'intervallo temporale antecedente al giorno dell'errore (giorno dell'errore compreso);
- falso positivo indica che un campione segnalato come anomalo risiede nelle zone al di fuori dell'intervallo di tempo antecedente al giorno dell'errore (giorno dell'errore compreso).

Per questa analisi, sono state utilizzate le curve ROC.

Curve ROC

Le curve ROC (Receiver Operating Characteristic) sono degli schemi grafici per un classificatore binario[51], che studiano i rapporti tra veri e falsi allarmi.
Tali curve vengono costruite tracciando la percentuale di veri positivi rispetto alla percentuale di falsi positivi al variare dei parametri.
Indicando con:

- VP i veri positivi;
- FP i falsi positivi;
- FN i falsi negativi;
- TN i veri negativi;

possiamo calcolare la percentuale di veri positivi (%VP, o TPR) come:

$$\%VP = TPR = \frac{TP}{TP + FN} \tag{5.6}$$

e la percentuale di falsi positivi (%FP o FPR) come:

$$\%FP = FPR = \frac{FP}{FP + TN} \tag{5.7}$$

Le curve ROC vengono tracciate in grafici che si estendono tra 0 e 1 come è possibile vedere nella figura 5.12. La diagonale rappresenta l'insieme dei punti per i quali la percentuale di veri positivi coincide con quella dei falsi positivi (random guess). Più la curva si trova distante dalla diagonale nella zona superiore, migliore sono le performance dell'algoritmo. Nel punto [0,1] si ha classificazione perfetta.

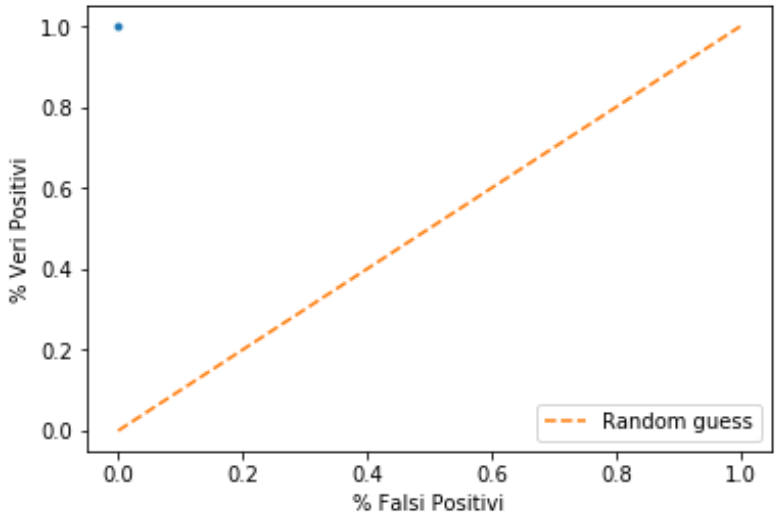


Figura 5.12: Spazio per tracciare le ROC

ROC per analisi esplorativa dei guasti al variare del numero di giorni di osservazione di un errore

Per la curva in esame sono state fatte le seguenti assunzioni:

- Sono prese in considerazione solo le caldaie migliori:
  - Mancanza di grandi vuoti nelle misure
  - Campionamenti constanti
- Per disegnare la curva ROC, sono stati calcolate le percentuali di veri positivi e falsi positivi al variare dell'intervallo di giorni di osservazione per cui è corretto segnalare un'anomalia.
  - Intervallo di giorni definito in maniera induttiva. Dato che nelle analisi operative delle caldaie, gli use case iniziano ad attivarsi tra i 4 e i 7 giorni prima, ho scelto di scegliere come intervallo di osservazione [0, 10] giorni.

Il grafico 5.13 mostra la ROC (coppie %Veri Positivi, %Falsi Positivi) al variare del numero di giorni scelti come intervallo di tempo in cui segnalare l'anomalia. Il valore iniziale [0,0] rappresenta il momento in cui si ha 0 come lunghezza dell'intervallo e il punto [1,1] al contrario si ha quando la lunghezza dell'intervallo è massima(10).
Da notare che la curva rimane sopra la diagonale per numero di giorni tra 2 e 5, e per 9 giorni. Ciò comporta che la percentuale di Veri Positivi è maggiore di quella di Falsi Positivi solo per i valori riportati in precedenza. Il campione con maggiore distanza dalla diagonale si ha per numero di giorni uguale a 2.

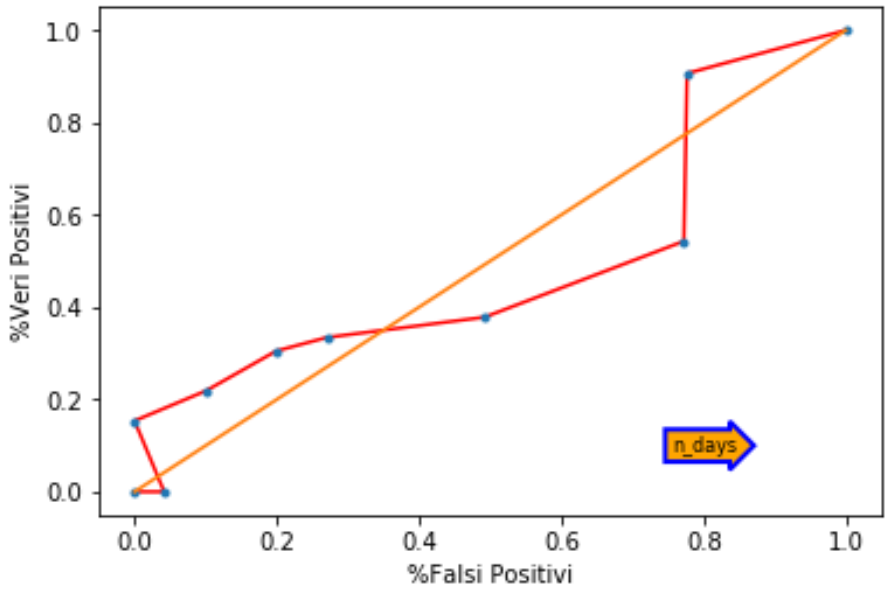


Figura 5.13: ROC sull'analisi esplorativa dei guasti

ROC per OneClass al variare della threshold e del numero di giorni di osservazione di un errore

Per la curva in esame sono state fatte le seguenti assunzioni:

- Sono prese in considerazione solo le caldaie migliori:
  - Mancanza di grandi vuoti nelle misure
  - Campionamenti constanti
- Per disegnare la curva ROC, sono stati calcolate le percentuali di veri positivi e falsi positivi al variare:
  - della threshold dell'algoritmo OneClass. L'intervallo di variazione della threshold è stato scelto in maniera induttiva. L'intervallo di riferimento è [-0.25, 0.2], dato che oltre questi valori(valori minori di -0.25 e maggiori di 0.2), il comportamento è lo stesso per l'algoritmo.
  - dell'intervallo di giorni per cui è corretto segnalare un'anomalia, che è stato definito in maniera induttiva. Dato che nelle analisi operative delle caldaie, gli use case iniziano ad attivarsi tra i 4 e i 7 giorni prima, ho scelto di scegliere come intervallo di osservazione [0, 10] giorni.

Il grafico 5.14 nella pagina seguente mostra la ROC (coppie %Veri Positivi, %Falsi Positivi) al variare della threshold dell'algoritmo OneClass e del numero di giorni di osservazione. Il valore iniziale [0,0] rappresenta il momento in cui la threshold è minima (-0.25)

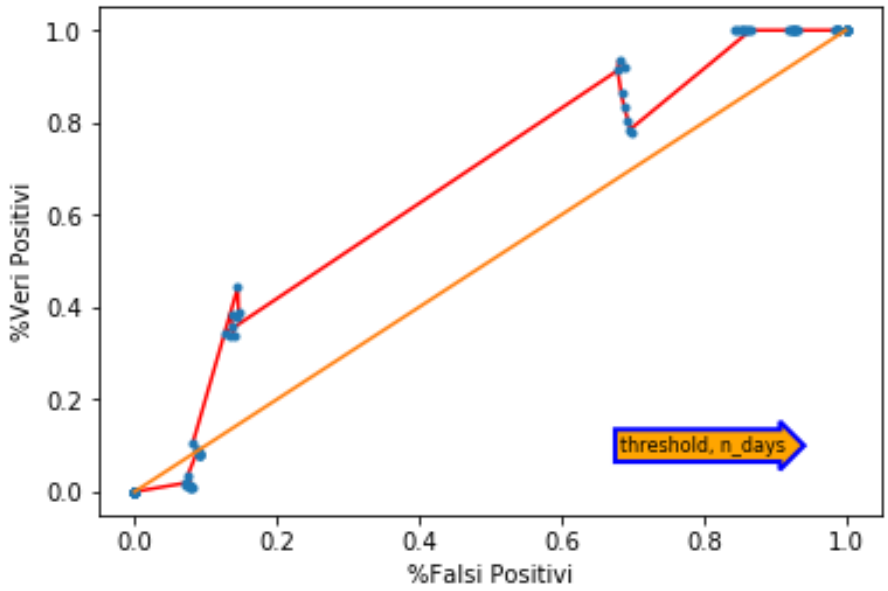


Figura 5.14: ROC dell'algoritmo One-Class

e il numero di giorni è minimo (0) e il punto [1,1] al contrario si ha quando la threshold e il numero di giorni sono massimi (0.2 e 10).
Da notare che i valori per una determinata threshold, al variare del numero di giorni sono racchiusi in piccole aree del grafico. Ciò vuol dire che la variazione della threshold ha un impatto maggiore rispetto alla variazione dei giorni. Inizialmente la curva sta sotto la diagonale o sulla stessa. Ciò comporta che per valori troppo bassi della threshold, il grafico ha una percentuale più alta di Falsi Positivi rispetto a quella dei Veri Positivi. Aumentando il valore della threshold, la curva si sposta e si assesta sempre sopra la diagonale.
Il punto in cui la curva è più lontana dalla diagonale si verifica quando la threshold è 0.0 (quella standard) e il numero di giorni è 4, che corrisponde al punto in cui il rapporto fra le percentuali è la più alta (a favore dei Veri Positivi).

Risultati

L'analisi sui Veri Positivi e Falsi Positivi porta alle seguenti osservazioni:

- Per l'algoritmo OneClass la threshold ha un impatto molto più alto rispetto al numero di giorni per cui un campione può essere considerato o no un outlier. Le percentuali di Veri Positivi e Falsi Positivi crescono in maniera proporzionale al crescere della threshold. Dall'analisi svolta, il valore 0.0 per la threshold e 4 per il numero di giorni si è scoperto essere il migliore per rapporto tra Veri Positivi e Falsi Positivi.
  In base alla probabilità di veri positivi che si vuole avere, si deve tenere in conto una probabilità di falsi positivi che è sempre minore della precedente, tranne nella prima parte della curva, per percentuali basse di veri positivi. In particolare il punto più

distante dalla diagonale sia ha per una probabilità di veri positivi intorno al 40% alla quale va sacrificato un 15% di probabilità di falsi positivi.

- Per l'analisi dei guasti, al variare dell'intervallo di tempo in cui un campione è considerato outlier, il rapporto tra la percentuale di veri positivi e falsi positivi è per certi intervalli ([2-4],9) a favore dei veri positivi. In alcuni casi però, è a favore dei falsi positivi.
  La curva mostra che il rapporto è massimo quando il numero di giorni è 2.
  In base alla percentuale di veri positivi che si desidera avere, la percentuale di falsi positivi è quasi sempre di poco inferiore, e per parti del grafico è anche superiore a quella dei veri positivi. Nel punto più distante dalla diagonale si ha una percentuale di veri positivi del 15% alla quale va sacrificata una percentuale di falsi positivi dello 0.05%.

- Entrambi gli algoritmi si comportano bene in caso di mancanza di errori, dato che non generano falsi positivi.

Capitolo 6

Conclusioni

L'Internet of Things ha permesso di connettere alla rete le caldaie negli edifici.
Riuscire a monitorare le informazioni prodotte da quest'ultime, anche se in grandi quantità, permette di salvaguardare la salute delle caldaie e di renderle più efficienti. In questo modo inoltre, i distributori di servizi per le caldaie possono conoscere in anticipo quali utenti necessitano di manutenzionare le proprie caldaie e gestire in maniera più semplice ed efficiente gli interventi da eseguire.
Questo lavoro di tesi ha permesso di studiare i limiti e i risultati sperimentali di alcuni approcci, anche Big Data, basati su tecniche di machine learning per la *predictive maintainance*, in modo da poter intervenire prima che si verifichi un guasto.
L'utilizzo del classificatore multi-class, Decision Tree, consente di affermare che localmente e in presenza di pochi dati, l'utilizzo di Python e di Scikit-Learn come libreria di machine learning è il metodo più efficiente. Incrementando la quantità di dati però, l'efficienza si abbassa repentinamente. Discorso opposto per Spark (testato su ambiente Java con MLlib e Scikit-Learn e su ambiente Python con Scikit-Learn), che mostra un notevole aumento dell'efficienza con l'aumento della quantità di dati.
Gli esperimenti su cluster, effettuati con Spark su ambiente Java ed MLlib o Scikit-Learn come librerie di Machine Learning, denotano che l'algoritmo permette di scalare bene orizzontalmente (capacità di migliorare i tempi di esecuzione aumentando il numero dei core) in entrambe le configurazioni, e in particolar modo con l'uso di Scikit-Learn. Discorso opposto per la scalabilità verticale (capacità di aumentare la quantità di dati da processare senza peggiorare significativamente i tempi di esecuzione), dato che MLlib scala bene al crescere della quantità di dati, ma al contrario Scikit-Learn genera sempre un aumento non trascurabile del tempo di esecuzione.
Spark su ambiente Java e MLlib come libreria di Machine Learning si è dimostrato essere la combinazione migliore in termini di efficienza su larga scala, con tempi 10 volte inferiori rispetto alla configurazione alternativa.
Le sperimentazioni del classificatore one-class, Isolation Forest, hanno invece permesso di individuare comportamenti anomali dai dati reali, forniti da esperti di dominio, e di estrapolare delle statistiche sulla correttezza del risultato, grazie anche all'utilizzo combinato di alcuni strumenti (Scatter Matrix e Decision Function dell'algoritmo one-class).

Il limitato numero di dataset a disposizione non ha permesso analisi esaustive ed ha portato ad ottenere una precisione dell'algoritmo non molto alta, dato che per ottenere una buona percentuale di veri positivi deve essere sacrificata una percentuale non irrisoria di falsi positivi. Nonostante ciò, l'algoritmo è pronto per essere testato in presenza di molti dataset, in modo da migliorarne la precisione e poterne valutare le vere potenzialità. Precisione che può essere migliorata anche cambiando opportunamente il modo e i tempi di campionamento delle variabili.

Sviluppi Futuri Alcuni tra i possibili sviluppi futuri di questo lavoro potrebbero essere ulteriori sperimentazioni, soprattutto su dataset reali per valutarne l'efficienza, ma anche l'integrazione con le correlazioni da dati meteo per garantire la massima efficienza di funzionamento della caldaia (energy saving models).
In presenza di grandi dataset reali, sarebbe possibile anche effettuare dei test su cluster del classificatore one-class, per studiarne le performance.

Appendice A

Codice utilizzato

Sono riportati di seguito alcune parti di codice che hanno aiutato nello svolgimento del lavoro di tesi.

A.1 Generazione Scatter Matrix

Una delle funzioni essenziali per il lavoro di tesi è quella che permette di disegnare la scatter matrix. Al fine di plottare la matrice di scatter, sono state necessarie le librerie riportate nel frammento di codice A.1.

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
```

Listing A.1: Librerie necessarie al plottaggio della Scatter Matrix

La funzione di plottaggio(A.2) prende 3 parametri, dei quali:

- *data_sm* corrisponde al dataset. Esso deve essere strutturato in modo che l'algoritmo possa processarlo.
- *model* è il modello creato dell'algoritmo di machine learning. Il modello non è specifico di un algoritmo, ma può essere un qualsiasi modello di un algoritmo one-class, che sia dotato di decision_function.
- *fig_inches* rappresenta la dimensione in inches(pollici) della matrice.

```
1 def draw_scatter_matrix(data_sm, model, fig_inchs):
```

Listing A.2: Definizione funzione per il plottaggio della Scatter Matrix

Nel frammento di codice A.3 viene plottata la Scatter Matrix standard, fornita dalla libreria *pandas*:

- prima si estrapolano le variabili dal dataset;

- viene quindi creata la nuova figura (*figure*);
- infine si esegue il plot della Scatter Matrix standard, che ha dimensioni *fig_inches*, e contiene nella diagonale gli istogrammi(*hist*) delle variabili.

```
1     varList = data_sm.columns.values
2     plt.figure()
3     scm = pd.plotting.scatter_matrix(data_sm[varList],
          figsize=(fig_inchs,fig_inchs), diagonal='hist',
4                                      marker=".", color = "blue")
```

Listing A.3: Codice per plottaggio della Scatter Matrix Standard

Nella Scatter Matrix standard, ad ogni coppia di variabili corrisponde un grafico a dispersione. Vanno ora inserite le zone a diversa probabilità in ogni grafico.
I passi successivi sono:

- allenamento del modello sui valori delle 2 variabili; le variabili sono selezionate a coppie, per le quali si allena di volta in volta il modello.

```
1  model.fit(data_sm[[v1,v2]].values)
```

Listing A.4: Training del modello

- calcolo della *decision function*; dal modello viene poi calcolata la decision function dell'algoritmo, in modo da poter effettuarne il plot.

```
1  Z = model.decision_function(np.c_[X.ravel(), Y.ravel()])
```

Listing A.5: Decision function dell'algoritmo

- plottaggio delle zone a diversa probabilità. Infine viene effettuato il plot delle varie zone, grazie alla funzione *contourf*. Gli indici j e i indicano la posizione nella Scatter Matrix dello specifico grafico per le due variabili considerate.

```
1  scm[j,i].contourf(x, y, Z, alpha = 0.6,cmap=plt.cm.rainbow);
```

Listing A.6: Plottaggio zone a diversa probabilità

Per completezza, si riporta il codice integrale della funzione (A.7).

```
1  def draw_scatter_matrix(data_sm, model, fig_inchs):
2      varList = data_sm.columns.values
3      plt.figure()
4      scm = pd.plotting.scatter_matrix(data_sm[varList],
           figsize=(fig_inchs,fig_inchs), diagonal='hist',
5                                      marker=".", color = "blue")
6      i=0 #indice for esterno
7      j=0 #indice for interno
8      for v1 in varList:
```

```
9           for v2 in varList:
10              model.fit(data_sm[[v1,v2]].values)
11              max_v1 = max(data_sm[v1])
12              max_v2 = max(data_sm[v2])
13              min_v1 = min(data_sm[v1])
14              min_v2 = min(data_sm[v2])
15              y = np.linspace(min_v2, max_v2)
16              x = np.linspace(min_v1, max_v1)
17              X, Y = np.meshgrid(x, y)
18              Z = model.decision_function(np.c_[X.ravel(),
                    Y.ravel()])
19              Z = Z.reshape(X.shape)
20              if i!=j:  #al di fuori della diagonale
21                  scm[j,i].contourf(x, y, Z, alpha =
                        0.6,cmap=plt.cm.rainbow);
22              j+=1
23          j=0
24          i+=1
25      return
```

Listing A.7: Codice integrale Scatter Matrix

A.2 Uso di Spark su cluster

Il codice utilizzato per Spark su cluster sfrutta lo scheduler yarn. Viene lanciato da linea di comando (vedi A.8).

```
1 spark2-submit --class MAIN_CLASS --master yarn --deploy-mode
    cluster [--executor-memory DIM] [--executor-cores CORES]
    [--num-executors NUM] [--files FILE_LIST] PATH_JAR INPUT_PATH
    OUTPUT_PATH LIB PATH_MODEL [NUM_MESI]
```

Listing A.8: Comando completo per l'esecuzione su cluster di spark

I parametri che devono essere settati sono:

- MAIN_CLASS è la classe main del progetto, compreso il relativo package (ad esempio: it.example.MainClass).
- DIM è la dimensione della memoria del singolo executor. Se non inserita corrisponde a 1G.
- CORES è il numero di core per ogni executor. Di default è 1.
- NUM permette di indicare il numero di executor coinvolti nel calcolo, se non si vuole usare quello standard (2);
- FILE_LIST consente di passare i file, necessari alla computazione tramite scikit-learn (il file python da eseguire per la effettuare predizione e il modello allenato del Decision Tree in formato pkl);

- PATH_JAR è il percorso al jar del progetto, all'interno della struttura del cluster(HDFS).
- INPUT_PATH è l'argomento che rappresenta il percorso di input per i dati;
- OUTPUT_PATH è il percorso per l'output;
- LIB è un argomento che consente di scegliere quale libreria di Machine Learning usare. I possibili valori sono "mllib" e "scikit".
- PATH_MODEL è il percorso al modello del Decision Tree allenato con MLlib/Scikit-Learn;
- NUM_MESI è un parametro opzionale che consente di scegliere a quanti mesi di campionamenti deve essere ridotto il calcolo.

Modello degli algoritmi Il modello per Scikit-Learn è stato allenato in locale (è stato preso il file di estensione pkl già usato per le sperimentazioni in locale). Nel frammento di codice A.9 è mostrata la procedura per la creazione del modello.

```
1  import pandas as pd
2  from sklearn.tree import DecisionTreeClassifier
3  from sklearn.externals import joblib
4
5  data = pd.read_csv(input_path, sep=",")
6  X = data.drop(["id","timestamp","label"],axis=1)
7  y = data["label"]
8  tree_model = DecisionTreeClassifier()
9  tree_model.fit(X, y)
10 joblib.dump(tree_model, outputPathPkl)
```

Listing A.9: Training Decision Tree per Scikit-Learn

Dopo aver letto il file, vengono separate le colonne. In seguito il decision tree viene creato e allenato. Infine, tramite la libreria *joblib* viene salvato il modello in formato pkl (pickle). Il modello per MLlib è stato allenato nel cluster stesso, tramite il frammento di codice A.10.

```
1  JavaRDD<String> data = javaSparkContext.textFile(inputPath);
2  data = data.sample(true, 0.1);
3  JavaRDD<LabeledPoint> train_data = data.map(new
       MapLabeledPoints());
4  int numClasses = 3;
5  Map<Integer, Integer> categoricalFeaturesInfo = new HashMap<>();
6  String impurity = "gini";
7  int maxDepth = 5;
8  int maxBins = 32;
9
10 DecisionTreeModel mllib_model =
       DecisionTree.trainClassifier(train_data, numClasses,
```

```
11 categoricalFeaturesInfo, impurity, maxDepth, maxBins);
12
13 mllib_model.save(javaSparkContext.sc(),outputPath);
```

Listing A.10: Training Decision Tree per MLlib

Viene selezionato il 10% del dataset, dopodiché i campioni sono mappati in *LabeledPoint*, ed infine viene creato il modello con la funzione *trainClassifier* e salvato nel percorso di output.

Uso di Scikit-Learn Per poter usare Scikit-Learn è necessario avviare un processo che esegua lo script Python passato come argomento con l'opzione *–files*.
Il frammento di codice A.11 mostra come sia stato effettuato e come ricevere l'output dello script Python.

```
1 Process p = Runtime.getRuntime().exec("python␣"+ pythonScript +
     argString);
2 BufferedReader in = new BufferedReader(new InputStreamReader
     (p.getInputStream()));
3 String res = in.readLine();
```

Listing A.11: Uso di Scikit su cluster

La funzione *exec* della classe *Runtime* consente di avviare il nuovo processo, eseguendo la linea di comando passata come argomento.

Bibliografia

[1] Dhruba Borthakur. *HDFS Architecture Guide*. A cura di The Apache Software Foundation. [Online; Documentazione Ufficiale]. Lug. 2018. URL: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html#NameNode+and+DataNodes.

[2] Jason Brownlee. *Difference Between Classification and Regression in Machine Learning*. A cura di Machine Learning Mastery. [Online; postato 11-Dicembre-2017]. Set. 2018. URL: https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/.

[3] Giulio Cantali. *Il database NoSQL Cassandra*. A cura di Database Master. [Online; Postato 26-Settembre-2017]. Lug. 2018. URL: http://databasemaster.it/database-nosql-cassandra/.

[4] Arcadia Data. *How Apache Spark Integrates with Apache Hadoop*. A cura di Arcadia Data. [Online; postato 21-Marzo-2017]. Lug. 2018. URL: https://www.arcadiadata.com/resources/knowledge-base/article/apache-spark/.

[5] DataStax. *Comparing the Hadoop File System (HDFS) with the Cassandra File System (CFS)*. A cura di DataStax. [Online;] lug. 2018. URL: https://www.datastax.com/resources/whitepapers/hdfs-vs-cfs.

[6] DB-Engines. *System Properties Comparison Apache Drill vs. Hive vs. Spark SQL*. A cura di DB-Engines. [Online;] lug. 2018. URL: https://db-engines.com/en/system/Apache+Drill%3BHive%3BSpark+SQL.

[7] Scikit learn developers. *Decision Trees*. A cura di scikit learn.org. [Documentazione Ufficiale]. Lug. 2018. URL: http://scikit-learn.org/stable/modules/tree.html.

[8] Scikit learn developers. *scikit-learn*. A cura di scikit learn.org. [Sito Ufficiale]. Lug. 2018. URL: http://scikit-learn.org/stable/index.html.

[9] Scikit learn developers. *sklearn.ensemble.IsolationForest*. A cura di scikit learn.org. [Documentazione Ufficiale]. Lug. 2018. URL: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html.

[10] Scikit Learn Developers. *sklearn.neighbors.LocalOutlierFactor*. A cura di scikit learn.org. [Documentazione Ufficiale]. Set. 2018. URL: http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html#sklearn.neighbors.LocalOutlierFactor.

[11] DeZyre. *Apache Spark Ecosystem and Spark Components*. A cura di DeZyre. [Online; Postato 26-Febbraio-2016]. Lug. 2018. URL: https://www.dezyre.com/article/apache-spark-ecosystem-and-spark-components/219.

[12] Zhiguo Ding e Minrui Fei. *An Anomaly Detection Approach Based on Isolation Forest Algorithm for Streaming Data using Sliding Window*. A cura di ScienceDirect. [Online; postato nel 2013]. Set. 2018. DOI: 10.3182/20130902-3-CN-3020.00044. URL: https://www.sciencedirect.com/science/article/pii/S1474667016314999.

[13] Maurizio Dècina. *Internet of things, ecco gli standard di comunicazione*. A cura di corrierecomunicazioni.it. [Online; postato 16-Settembre-2016]. Lug. 2018. URL: https://www.corrierecomunicazioni.it/digital-economy/internet-of-things-ecco-gli-standard-di-comunicazione/.

[14] EliteDataScience. *Data Cleaning*. A cura di elitedatascience.com. Lug. 2018. URL: https://elitedatascience.com/data-cleaning.

[15] Apache Software Foundation. *Apache Cassandra*. A cura di The Apache Software Foundation. [Online; Documentazione Ufficiale]. Lug. 2018. URL: http://cassandra.apache.org/.

[16] Apache Software Foundation. *Apache Drill*. A cura di Apache Software Foundation. [Online; Sito Ufficiale]. Lug. 2018. URL: https://drill.apache.org/.

[17] Apache Software Foundation. *Machine Learning Library (MLlib) Guide*. A cura di The Apache Software Foundation. [Online; Documentazione Ufficiale]. Lug. 2018. URL: https://spark.apache.org/docs/latest/ml-guide.html.

[18] Apache Software Foundation. *MLlib*. A cura di The Apache Software Foundation. [Online; Documentazione Ufficiale]. Lug. 2018. URL: https://spark.apache.org/mllib/.

[19] Apache Software Foundation. *Spark SQL, DataFrames and Datasets Guide*. A cura di The Apache Software Foundation. [Online; Documentazione Ufficiale]. Lug. 2018. URL: https://spark.apache.org/docs/latest/sql-programming-guide.html.

[20] Apache Software Foundation. *Spark Streaming Programming Guide*. A cura di The Apache Software Foundation. [Online; Documentazione Ufficiale]. Lug. 2018. URL: https://spark.apache.org/docs/latest/streaming-programming-guide.html.

[21] Prashant Gupta. *Decision Tree in Machine Learning*. A cura di Towards Data Science. [Online; postato 17-Maggio-2017]. Set. 2018. URL: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052.

[22] Vihar Kurama. *Introduction To Machine Learning*. A cura di Towards Data Science. [Online; postato 15-Luglio-2017]. Lug. 2018. URL: https://towardsdatascience.com/introduction-to-machine-learning-db7c668822c4.

[23] Vincenzo La Spesa. *Evoluzione genetica applicata agli alberi di classificazione*. A cura di thedarshan.com. [Online;] lug. 2018. URL: http://www.thedarshan.com/tesi/tesi/Machine_learning_e_classificazione_supervisionata.html.

[24] Eryk Lewinson. *Outlier Detection with Isolation Forest*. A cura di towardsdatascience.com. [Online; postato 2-Luglio-2018]. Lug. 2018. URL: https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e.

[25] Dan Lynn. *APACHE SPARK CLUSTER MANAGERS: YARN, MESOS, OR STANDALONE?* A cura di AGILDATA. [Online; Postato 15-Marzo-2016]. Lug. 2018. URL: http://www.agildata.com/apache-spark-cluster-managers-yarn-mesos-or-standalone/.

[26] Rebecca Mantovani. *L'internet delle cose in 8 domande e risposte*. A cura di focus.it. [Online; postato 30-Ottobre-2015]. Lug. 2018. URL: https://www.focus.it/tecnologia/innovazione/tutto-quello-che-ce-da-sapere-sullinternet-of-things-in-x-domande-e-risposte.

[27] Avinash Navlani. *KNN Classification using Scikit-learn*. A cura di DataCamp. [Online; postato il 2-Agosto-2018]. Set. 2018. URL: https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn.

[28] Michele Nemschoff. *YARN vs MESOS: Can't We All Just Get Along?* A cura di mapr.com. [Online; Postato 22-Giugno-2015]. Lug. 2018. URL: https://mapr.com/blog/yarn-vs-mesos-cant-we-all-just-get-along/.

[29] Modbus Organization. *Modbus*. A cura di http://www.modbus.org. Lug. 2018. URL: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf.

[30] Sandro Paganotti. *Introduzione a Apache Cassandra*. A cura di HTML.it. [Online;] lug. 2018. URL: http://www.html.it/articoli/introduzione-a-apache-cassandra-1/.

[31] Sunil Ray. *6 Easy Steps to Learn Naive Bayes Algorithm (with codes in Python and R)*. A cura di Analytics Vidhya. [Online; postato il 11-Settembre-2017]. Set. 2018. URL: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/.

[32] Neeraja Rentachintala. *Big Data SQL: Overview of Apache Drill Query Execution Capabilities – Whiteboard Walkthrough*. A cura di mapr.com. [Online; postato 3-Agosto-2016]. Lug. 2018. URL: https://mapr.com/blog/big-data-sql-overview-apache-drill-query-execution-capabilities-whiteboard-walkthrough/.

[33] Scott Robinson. *K-Nearest Neighbors Algorithm in Pythonand Scikit-Learn*. A cura di Stack Abuse. [Online; postato il 15-Febbraio-2018]. Set. 2018. URL: https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/.

[34] Margaret Rouse, Jack Vaughan e Emma Preslar. *Hadoop Distributed File System (HDFS)*. A cura di TechTarget SearchDataManagement. Lug. 2018. URL: https://searchdatamanagement.techtarget.com/definition/Hadoop-Distributed-File-System-HDFS.

[35] Margaret Rouse, Jack Vaughan e Delaney Rebernik. *Apache Hive*. A cura di TechTarget SearchDataManagement. Lug. 2018. URL: https://searchdatamanagement.techtarget.com/definition/Apache-Hive.

[36] Rajesh Sampathkumar. *How is scikit-learn compared with Apache Spark's MLlib?* A cura di Quora. [Online; postato 17-Settembre-2016]. Lug. 2018. URL: https://www.quora.com/How-is-scikit-learn-compared-with-Apache-Sparks-MLlib.

[37] Rajesh Sampathkumar. *Which is the best tool for machine learning: MlLib (Apache-Spark) or scikit-learn?* A cura di Quora. [Online; postato 17-Settembre-2016]. Lug. 2018. URL: https://www.quora.com/Which-is-the-best-tool-for-machine-learning-MlLib-Apache-Spark-or-scikit-learn.

[38] sterlinux. *Le architetture nell'Internet of Things*. A cura di www.iotsolutions.it. [Online; postato 25-Novembre-2016]. Lug. 2018. URL: http://www.iotsolutions.it/le-architetture-nellinternet-of-things/.

[39] James Stradling. *Unsupervised Machine Learning with One-class Support Vector Machines*. A cura di ThisData. [Online; postato il 21-Ottobre-2016]. Set. 2018. URL: https://thisdata.com/blog/unsupervised-machine-learning-with-one-class-support-vector-machines/.

[40] tutorialspoint. *Spark SQL - DataFrames*. A cura di tutorialspoint.com. [Online;] lug. 2018. URL: https://www.tutorialspoint.com/spark_sql/spark_sql_dataframes.htm.

[41] Sumit Vyas. *How is scikit-learn compared with Apache Spark's MLlib?* A cura di Linkedin. [Online; postato 31-Gennaio-2017]. Lug. 2018. URL: https://www.linkedin.com/pulse/hive-spark-vs-sparksql-sumit-vyas/.

[42] Wikipedia. *Apache Drill*. A cura di it.wikipedia.org. Lug. 2018. URL: https://en.wikipedia.org/wiki/Apache_Drill.

[43] Wikipedia. *Apache Hive*. A cura di it.wikipedia.org. Lug. 2018. URL: https://en.wikipedia.org/wiki/Apache_Hive.

[44] Wikipedia. *Big data*. A cura di it.wikipedia.org. Lug. 2018. URL: https://it.wikipedia.org/wiki/Big_data.

[45] Wikipedia. *Binary classification*. A cura di it.wikipedia.org. Set. 2018. URL: https://en.wikipedia.org/wiki/Binary_classification.

[46] Wikipedia. *Cassandra (database)*. A cura di it.wikipedia.org. Lug. 2018. URL: https://it.wikipedia.org/wiki/Cassandra_(database).

[47] Wikipedia. *Modbus*. A cura di it.wikipedia.org. Lug. 2018. URL: https://it.wikipedia.org/wiki/Modbus.

[48] Wikipedia. *Multiclass classification*. A cura di it.wikipedia.org. Lug. 2018. URL: https://en.wikipedia.org/wiki/Multiclass_classification.

[49] Wikipedia. *One-class classification*. A cura di it.wikipedia.org. Lug. 2018. URL: https://en.wikipedia.org/wiki/One-class_classification.

[50] Wikipedia. *OpenTherm*. A cura di it.wikipedia.org. Lug. 2018. URL: https://en.wikipedia.org/wiki/OpenTherm.

[51] Wikipedia. *Receiver operating characteristic*. A cura di it.wikipedia.org. Lug. 2018. URL: https://en.wikipedia.org/wiki/Receiver_operating_characteristic.

[52] Wikipedia. *Scikit-learn*. A cura di it.wikipedia.org. Lug. 2018. URL: https://it.wikipedia.org/wiki/Scikit-learn.

[53] Serdar Yegulalp. *10 ways to query Hadoop with SQL*. A cura di InfoWorld. [Online; postato 16-Settembre-2014]. Lug. 2018. URL: https://www.infoworld.com/article/2683729/hadoop/10-ways-to-query-hadoop-with-sql.html.

[54] Matei Zaharia et al. *Learning Spark*. A cura di www.safaribooksonline.com. [Online; lug. 2018. URL: https://www.safaribooksonline.com/library/view/learning-spark/9781449359034/ch01.html.