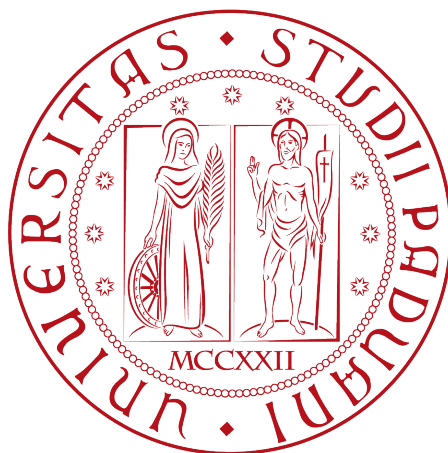


Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

CORSO DI LAUREA IN INFORMATICA



## Studio e Implementazione di un Assistente Virtuale Dotato di Intelligenza Artificiale

*Tesi di laurea triennale*

*Relatore*

Prof. Michele Scquizzato

*Laureando*

Gianluigi Pellè

---

ANNO ACCADEMICO 2019-2020



# Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecento ore, dal laureando Gianluigi Pellè presso l'Azienda ASI s.r.l. nel periodo giugno/luglio 2020.

Lo scopo dello stage era la realizzazione di un assistente virtuale, dotato di intelligenza artificiale, in grado di risolvere in modo automatico i problemi più comuni riscontrati dai clienti durante l'utilizzo del gestionale Plain. L'obiettivo principale era lo studio e l'analisi delle funzionalità offerte dal machine learning, nello specifico dal framework ML.NET, per capire come integrarle all'interno di un prodotto sperimentale che potesse ridurre il numero di richieste in arrivo al supporto clienti dell'azienda.

Il primo capitolo della presente relazione descrive la proposta di stage, esponendone le attività, i requisiti e le tecnologie da approfondire richieste. Il secondo capitolo documenta la fase di formazione personale che ha avuto come oggetti di studio il machine learning, con la descrizione delle varie tipologie e problematiche che è in grado di risolvere, e il funzionamento di una chat bot. Il terzo e il quarto capitolo analizzano la fase di progettazione presentando i prototipi creati e lo sviluppo del prodotto con le relative scelte architetture. In conclusione, il quinto capitolo presenta un resoconto degli obiettivi soddisfatti, una valutazione personale dell'esperienza svolta e delle future potenzialità che potrebbero essere integrate nel prodotto sviluppato.



# Ringraziamenti

*Innanzitutto, desidero ringraziare il Prof. Michele Scquizzato, relatore della mia tesi, per la Sua disponibilità e per l'aiuto fornitomi durante la stesura della tesi.*

*Vorrei inoltre ringraziare Marco e Stefano di ASI s.r.l. per i consigli e il supporto che mi avete fornito durante lo stage.*

*Padova, Settembre 2020*

Gianluigi Pellè



# Indice

<b>1</b>	<b>Presentazione del progetto</b>	<b>1</b>
1.1	Contesto aziendale	1
1.1.1	Profilo aziendale	1
1.1.2	Plain	1
1.2	Proposta di stage	2
1.3	Tecnologie da approfondire	3
1.4	Piano di lavoro	3
1.5	Obiettivi richiesti	3
1.5.1	Requisiti di minima	3
1.5.2	Requisiti medi	4
1.5.3	Requisiti massimi	4
1.6	Analisi dei rischi	4
1.7	Rapporto con il tutor aziendale	5
1.8	Aspettative personali	5
<b>2</b>	<b>Fase di studio individuale</b>	<b>7</b>
2.1	Machine Learning	7
2.1.1	Modalità di apprendimento	7
2.1.2	Compiti risolvibili con il machine learning	8
2.2	Microsoft ML.NET	10
2.3	Documentazione di ML.NET API	12
2.3.1	Preparazione dei dati	12
2.3.2	Valutazione del modello	12
2.4	Microsoft Bot Framework	13
2.5	Requisiti del progetto	14
2.6	Esempio di utilizzo del bot	15
<b>3</b>	<b>Fase di Progettazione</b>	<b>19</b>
3.1	Creazione dei primi prototipi	19
3.2	Creazione del Dataset	20
3.3	Definizione dei moduli	21
3.3.1	Modulo ML	21
3.3.2	Modulo Bot	25
<b>4</b>	<b>Contenuti opzionali</b>	<b>29</b>
4.1	Salvataggio delle nuove domande	29
4.2	Risoluzione dei problemi più frequenti	30
<b>5</b>	<b>Considerazioni finali</b>	<b>33</b>

5.1	Raggiungimento degli obiettivi . . . . .	33
5.2	Possibili miglioramenti futuri . . . . .	33
5.3	Valutazione personale . . . . .	34
<b>Glossary</b>		<b>35</b>
<b>Bibliografia</b>		<b>37</b>



# Elenco delle figure

2.1	Esempio di <i>Multiclass classification</i> per riconoscere la razza di un cane.	8
2.2	Esempio di Regressione per il calcolo del valore di un immobile. . . . .	9
2.3	Esempio di <i>Clustering</i> per la suddivisione di un insieme di dati in sottogruppi. . . . .	10
2.4	Esempio di utilizzo del <i>Model Builder</i> . . . . .	11
2.5	Interfaccia del <i>Microsoft Bot Framework Emulator</i> . . . . .	13
2.6	Esempio di utilizzo dell'assistente virtuale. . . . .	16
3.1	Esempio di utilizzo della <i>ML.NET CLI</i> . . . . .	20
3.2	Esempio del <i>dataset</i> creato manualmente. . . . .	21
3.3	Diagramma del modulo ML. . . . .	21
3.4	Diagramma della classe <i>MLBuilder</i> . . . . .	22
3.5	Sequenza delle operazioni necessarie per creare un modello personalizzato.	23
3.6	Diagramma della classe <i>MLManager</i> . . . . .	25
3.7	Diagramma del modulo Bot. . . . .	25
3.8	Diagramma della classe <i>AsiBot</i> . . . . .	26
3.9	Diagramma della classe <i>AsiCategoryHandler</i> . . . . .	27
3.10	Diagramma della classe <i>ConversationData</i> . . . . .	28
4.1	Diagramma della classe <i>DatasetManager</i> . . . . .	29
4.2	Interfaccia della scheda per raccogliere il <i>feedback</i> dell'utente. . . . .	30
4.3	Diagramma della classe <i>QuickResponseCategoryHandler</i> . . . . .	31

# Elenco delle tabelle

1.1	Piano di lavoro. . . . .	3
1.2	Analisi dei rischi. . . . .	4
2.1	Requisiti funzionali. . . . .	14
2.2	Requisiti non funzionali. . . . .	14
2.3	UC 1 - Risoluzione di un problema riscontrato nel prodotto. . . . .	15
2.4	UC 1.1 - Feedback da parte dell'utente sulla soluzione proposta. . . . .	16
2.5	UC 1.1.1 - L'utente non riesce a risolvere il problema con la soluzione proposta. . . . .	17
2.6	UC 2 - Il sistema è indeciso sulla categoria del problema. . . . .	17
2.7	UC 3 - L'utente non riceve una soluzione al problema. . . . .	17

# Capitolo 1

## Presentazione del progetto

### 1.1 Contesto aziendale

#### 1.1.1 Profilo aziendale

ASI S.r.l.<sup>1</sup> è un'azienda fondata nel 1989 con sede a Padova che si occupa dello sviluppo e vendita di software di tipo industrializzato per imprese che operano in diversi settori. I principali prodotti dell'azienda sono:

- \* soluzioni ERP (Enterprise Resource Planning) per aziende commerciali e manifatturiere
- \* soluzioni per la gestione documentale
- \* soluzioni per la gestione delle risorse umane
- \* portali aziendali
- \* e-Commerce

ASI ha inoltre stretto e consolidato accordi di partnership con alcune tra le più importanti società di ricerca e sviluppo nell'ambito dell'hardware e del software, quali IBM<sup>2</sup> - da oltre 30 anni - e Microsoft<sup>3</sup>. L'azienda punta molto al futuro e all'innovazione collaborando con l'Università di Padova, con offerte di stage destinate alla ricerca e sperimentazione delle ultime tecnologie, e confrontandosi costantemente con altre aziende dello stesso settore all'interno dell'ICTLAB di Confindustria Padova.

#### 1.1.2 Plain

Plain ERP è una *suite*<sup>[g]</sup> di *soluzioni software* che permettono di pianificare, gestire e programmare tutte le attività di un'azienda: la produzione, le attività di vendita e acquisto, le operazioni di logistica e magazzino fino alle relazioni con i clienti.

---

<sup>1</sup>ASI: <https://www.plain.it/>.

<sup>2</sup>IBM: <https://www.ibm.com/it-it>.

<sup>3</sup>Microsoft: <https://www.microsoft.com/it-it/>.

### ERP commercio

Plain ERP commercio<sup>4</sup> è il *software gestionale* dedicato alle aziende commerciali che operano in settori merceologici specifici come ad esempio Carta, Cancelleria e prodotti per l'Ufficio, Ricambi automotive, Nautica, Industriale, Prodotti Medicali e Chimici, etc. ERP commercio automatizza il ricevimento della merce, lo stoccaggio, la spedizione e la gestione degli inventari aumentando così la produttività del personale e riducendo i costi e i tempi di evasione degli ordini.

### Altre soluzioni Plain

Tra i servizi proposti da Plain sono presenti dei moduli integrativi destinati a specifiche tipologie di aziende:

**Plain industria** permette di monitorare costantemente la marginalità sulle vendite tenendo in considerazione anche i costi di trasporto e altre voci di spesa

**Plain Logistica** offre una completa gestione del magazzino e dei flussi di materiali per aumentare la produttività e ottimizzare gli spazi

**Plain Documentale** permette di condividere i file inerenti una specifica attività tra tutte le persone che collaborano per lo stesso progetto, facendo risparmiare tempo e migliorando il lavoro di squadra

**Plain HR** raccoglie dati per confrontare le ore prodotte con ore lavorate e ore retribuite e genera report e analisi utilizzando Plain reportistica BI

## 1.2 Proposta di stage

L'offerta di stage proposta dall'azienda ASI srl di Padova consiste nello studio e approfondimento delle potenzialità del *Machine Learning*. Questa nuova tecnologia si basa sull'elaborazione dei dati che gli vengono forniti in input e riesce a sviluppare degli algoritmi che migliorano le proprie prestazioni in modo progressivo, man mano che vengono aggiunte informazioni al sistema.

L'interesse dell'azienda è quello di trovare una concreta applicazione di questa nuova tecnologia in modo da offrire un servizio più completo e innovativo ai propri clienti. In particolare, è stata richiesta la realizzazione di un assistente virtuale - integrato in una *chat bot*<sup>[9]</sup> - per risolvere in modo veloce e automatico alcune delle problematiche più comuni riscontrate durante l'utilizzo dei software acquistati dall'azienda. Le *chat bot* sono un ottimo strumento per offrire un servizio semplice e immediato, in quanto offrono all'utente l'interfaccia di una schermata di conversazione in stile chat dove l'utente può scrivere liberamente le sue problematiche e ottenere una risposta immediata.

L'assistente virtuale dovrebbe essere in grado di categorizzare il tipo di problema e proporre una possibile soluzione rimandando l'utente al manuale d'uso del software utilizzato tramite un link e lasciando a lui l'attività di consultazione per la risoluzione del problema. Nonostante sia richiesta una collaborazione attiva da parte del cliente, l'assistente permetterà una istantanea risoluzione dei problemi più comuni e favorirà la riduzione del numero di richieste in arrivo al team di assistenza tecnica. L'assistente virtuale dovrà anche imparare a partire dai propri errori: quando viene restituito il link a un manuale utente errato, è infatti necessario dare la possibilità all'utente di segnalare l'anomalia in modo che l'algoritmo possa migliorare la sua precisione.

---

<sup>4</sup>Plain ERP commercio: <https://www.plain.it/plain-software-gestione-aziendale-commercio/>.

## 1.3 Tecnologie da approfondire

È stato richiesto lo studio del *framework*<sup>[9]</sup> *ML.NET*<sup>5</sup> per capire le varie funzionalità offerte dalla piattaforma .NET<sup>6</sup> relative al *Machine Learning*, in quanto tutti i prodotti dell'azienda sono sviluppati con l'utilizzo della piattaforma *Microsoft .NET*. Il linguaggio di programmazione consigliato per l'utilizzo del *framework* *ML.NET* è stato il C#<sup>7</sup>. Infine, per lo sviluppo dell'interfaccia offerta dall'assistente virtuale è stato deciso di adottare il *Microsoft Bot Framework*<sup>8</sup> per simulare una conversazione con un dipendente addetto all'assistenza clienti.

Il canale di comunicazione utilizzato per interagire con il *bot* in fase di sviluppo è stato il *Microsoft Bot Emulator*, sistema in grado di creare una finestra di chat dove è possibile inviare dei messaggi che il *bot* può analizzare per produrre una risposta. Era stato richiesto anche uno studio per integrare una finestra di chat personalizzata all'interno del sito dell'azienda, ma si è preferito dedicare il tempo disponibile ad altre funzionalità.

## 1.4 Piano di lavoro

Il Piano di lavoro è stato redatto a seguito di un colloquio con il tutor aziendale ed è stato diviso in 6 fasi di diversa durata, per un ammontare complessivo di 300 ore di lavoro. Di seguito viene riportata la Tabella 1.1 che illustra in dettaglio ciascuna fase:

**Tabella 1.1:** Piano di lavoro.

Fase	Descrizione	Ore stimate
a)	Acquisizione degli standard aziendali	16
b)	Acquisizione delle conoscenze su <i>ML.NET</i>	56
c)	Relazione sulle funzionalità offerte da <i>ML.NET</i>	28
d)	Analisi e progettazione del prototipo funzionale	48
e)	Programmazione	136
f)	Documentazione	16

## 1.5 Obiettivi richiesti

Gli obiettivi di questo progetto sono divisi in obiettivi di minima, obiettivi medi e obiettivi massimi. Gli obiettivi massimi sono stati modificati in corso d'opera dopo la fase di studio iniziale in quanto non era ancora chiaro all'inizio il funzionamento dell'assistente virtuale.

### 1.5.1 Requisiti di minima

- \* Acquisizione degli standard aziendali

<sup>5</sup>ML.NET: <https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet>.

<sup>6</sup>Piattaforma .NET: <https://dotnet.microsoft.com>.

<sup>7</sup>C#: <https://docs.microsoft.com/en-us/dotnet/csharp/>.

<sup>8</sup>Microsoft Bot Framework: <https://dev.botframework.com>.

- \* Acquisizione dei concetti di base del *Machine Learning*
- \* Studio del *framework* *ML.NET*
- \* Realizzazione di un prototipo per mostrare alcune funzionalità del *Machine Learning*

### 1.5.2 Requisiti medi

- \* Realizzazione di una *chat bot* per aiutare l'utente a identificare la soluzione al suo problema all'interno di un manuale

### 1.5.3 Requisiti massimi

- \* Sviluppo di un sistema di raccolta dati per migliorare l'apprendimento automatico
- \* Sviluppo di una funzionalità per la risoluzione dei problemi più comuni che possono verificarsi in un qualsiasi prodotto Plain

## 1.6 Analisi dei rischi

Essendo la mia prima esperienza di lavoro in azienda, durante la fase iniziale dello stage ho individuato una serie di possibili rischi in cui sarei potuto incorrere specificando per ciascuno di essi le contromisure da adottare per minimizzarlo, che vengono riassunte in Tabella 1.2:

**Tabella 1.2:** Analisi dei rischi.

Descrizione rischio	Contromisure	Pericolosità
Non conoscevo il linguaggio di programmazione C# prima dell'inizio dello stage	Trovare il modo di apprendere il nuovo linguaggio il più rapidamente possibile, leggendo la documentazione per una conoscenza approfondita dei concetti di base e approfondendo i temi più avanzati con articoli specializzati	Media
Non avevo mai approfondito il tema dell'intelligenza artificiale	Approfondire il tema del <i>machine learning</i> e delle <i>reti neurali</i> <sup>[g]</sup> con corsi online e approfondimenti sull'argomento	Alta
Non è possibile installare il <i>Microsoft Bot Framework</i> in ambiente macOS	Ho dovuto trovare un computer Windows e configurarlo correttamente con i programmi necessari allo sviluppo software	Bassa
Scegliere adeguatamente i tempi necessari per la fase di formazione	In accordo con il tutor aziendale, abbiamo lasciato un po' di margine nella fase di formazione	Media

## 1.7 Rapporto con il tutor aziendale

A causa della attuale emergenza sanitaria legata al SARS-CoV-2, lo stage è stato svolto esclusivamente da remoto e l'interazione con il tutor aziendale è avvenuta in modalità telematiche per l'intera durata dello stage, utilizzando il servizio Skype<sup>9</sup>. Per risolvere, almeno parzialmente, il problema della presenza in azienda, si è deciso di fissare uno *stand-up meeting* quotidiano all'inizio della giornata lavorativa con lo scopo di mostrare l'avanzamento del progetto e risolvere eventuali problemi. Per tenere traccia delle attività svolte quotidianamente abbiamo utilizzato un file di testo condiviso grazie al servizio offerto dalla piattaforma OneDrive<sup>10</sup>.

## 1.8 Aspettative personali

Durante l'evento STAGE-IT ho partecipato a tutte le presentazioni delle offerte di stage a tema intelligenza artificiale, in quanto volevo cogliere l'occasione di approfondire questa nuova branca dell'informatica per cui nutro molta curiosità e interesse. Ho scelto di approfondire questo tema, a me ancora del tutto sconosciuto, perchè mi appassionava particolarmente e se avessi deciso di continuare i miei studi con un corso di laurea magistrale avrei scelto sicuramente l'indirizzo legato all'intelligenza artificiale e il *Machine Learning*. Dall'altra parte, ero anche molto curioso di affrontare un'esperienza di lavoro in azienda, in quanto ho lavorato come *freelance*<sup>[9]</sup> precedentemente con un contratto di prestazione occasionale, ma mai con un contratto aziendale.

Ho scelto di accettare la proposta di stage offerta da ASI perchè era più incentrata sulla fase di esplorazione e studio delle funzionalità offerte dal *Machine Learning* piuttosto che sulla realizzazione di un prodotto finale. L'offerta di stage era infatti molto vaga sul prodotto finale atteso e lasciava molta libertà in quanto richiedeva la sola creazione di un prototipo come obiettivo minimo.

---

<sup>9</sup>Skype: <https://www.skype.com/it/>.

<sup>10</sup>Microsoft One Drive: <https://www.microsoft.com/it-it/microsoft-365/onedrive/online-cloud-storage>.





## Capitolo 2

# Fase di studio individuale

### 2.1 Machine Learning

Il *machine learning* (noto anche come apprendimento automatico) è un ramo dell'Intelligenza Artificiale che si occupa di trovare relazioni tra i dati al fine di permettere alle macchine di eseguire dei compiti specifici senza che debbano essere programmate per farlo. L'algoritmo, infatti, è capace di elaborare un proprio modello matematico a partire dai dati che gli vengono forniti in *input*, che sono l'ingrediente essenziale di tutti gli approcci di apprendimento automatico.

Da Wikipedia<sup>1</sup>: "Un modello matematico è una rappresentazione quantitativa di un fenomeno naturale. Come tutti gli altri modelli usati nella scienza, il suo scopo è quello di rappresentare il più incisivamente possibile un determinato oggetto, un fenomeno reale o un insieme di fenomeni". La maggior parte dei modelli traduce un fenomeno in un sistema di equazioni differenziali, ordinarie o alle derivate parziali, che è poi possibile utilizzare per fare delle previsioni su nuovi dati di cui il risultato non sia noto. I modelli migliorano in modo iterativo con l'aggiunta di nuovi dati da analizzare, aumentando così le proprie prestazioni e affidabilità nelle previsioni.

Il *machine learning* viene utilizzato spesso in situazioni di vita quotidiana: ad esempio viene integrato nei software di riconoscimento vocale e nei sistemi di guida autonoma delle automobili, che sono diventati sempre più popolari e diffusi.

#### 2.1.1 Modalità di apprendimento

Sono disponibili diverse modalità di apprendimento, che si differenziano per la scelta degli algoritmi utilizzati e per lo scopo per cui vengono create le macchine. In base al tipo di algoritmo utilizzato per elaborare i dati e le informazioni fornite per il *training*, si possono categorizzare tre diversi sistemi di apprendimento automatico:

##### Apprendimento supervisionato

L'apprendimento supervisionato consiste nel fornire alla macchina un insieme di nozioni codificate ed esempi che presentano il risultato esatto a partire da un *input* iniziale. A partire da questo insieme di dati la macchina è in grado di elaborare una regola che permette di ottenere un risultato quanto più preciso da un nuovo input iniziale di dati.

---

<sup>1</sup>Wikipedia: [https://it.wikipedia.org/wiki/Pagina\\_principale](https://it.wikipedia.org/wiki/Pagina_principale).

### Apprendimento non supervisionato (o senza supervisione)

L'apprendimento non supervisionato consiste nel passare alla macchina un insieme di dati privi del proprio risultato finale, e si lascia alla macchina il compito di organizzare le informazioni in modo intelligente per ottenere i risultati migliori in base alle diverse situazioni che si presentano. In questo tipo di apprendimento la macchina ha una maggiore libertà di scelta.

### Apprendimento per rinforzo

L'apprendimento per rinforzo richiede il supporto di strumenti e sensori per comprendere e analizzare l'ambiente circostante al fine di effettuare delle scelte più consone alla situazione in cui si trova la macchina. Gli esempi più classici in cui viene utilizzato questo tipo di apprendimento sono i sistemi di guida autonoma e le intelligenze artificiali dei videogiochi.

## 2.1.2 Compiti risolvibili con il machine learning

I problemi che è possibile risolvere con algoritmi di apprendimento automatico si possono suddividere in tre macro categorie: Classificazione, Regressione e *Clustering*.

### Classificazione

I problemi di classificazione hanno il compito di attribuire la giusta categoria di appartenenza ai dati forniti in *input*. Esistono due tipi di classificazione: la *Binary Classification*, che distingue se un determinato input è positivo o negativo, e la *Multiclass Classification* che attribuisce una categoria di appartenenza precisa a partire da un insieme di categorie prefissato. Questo tipo di *tasks* viene elaborato con un apprendimento supervisionato. Alcuni esempi molto comuni di questa categoria di problemi sono lo *spam filtering*, che, a partire dalle informazioni di una mail, determina se si tratta di una mail di *spam*, e la classificazione delle immagini ad esempio per riconoscere la razza di un cane, come viene mostrato in Figura 2.1.



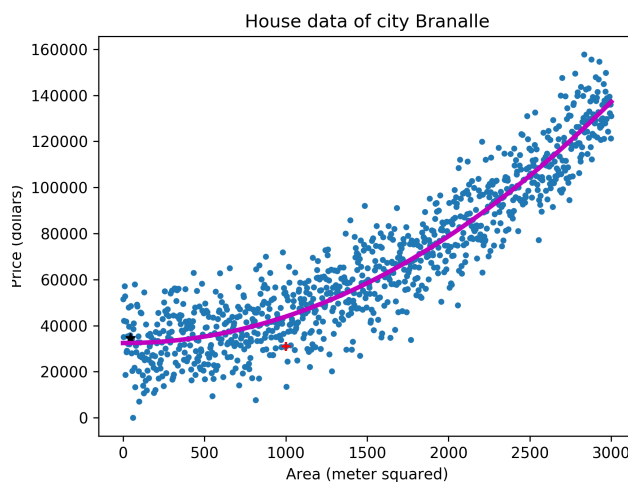
**Figura 2.1:** Esempio di *Multiclass classification* per riconoscere la razza di un cane.

---

<sup>1</sup>Fonte Figura 2.1: <https://medium.com/@ajay.pratap/how-to-build-dog-breed-classifier-with-cnn-model-usi>

## Regressione

Un altro tipo di problemi che vengono risolti con un apprendimento automatico supervisionato sono quelli di regressione, in cui bisogna calcolare il valore di un particolare oggetto o caratteristica a partire da un insieme di informazioni relative allo stesso. L'esempio più comune quando si parla di regressione in ambito di apprendimento automatico è quello del calcolo del valore di un immobile: a partire da un insieme di case di cui si conosce la superficie, il numero di stanze, l'anno di costruzione e altre informazioni rilevanti, la macchina elabora un modello di regressione con cui è possibile calcolare il prezzo di una nuova casa da mettere in vendita. Un esempio del problema di regressione viene illustrato in Figura 2.2.



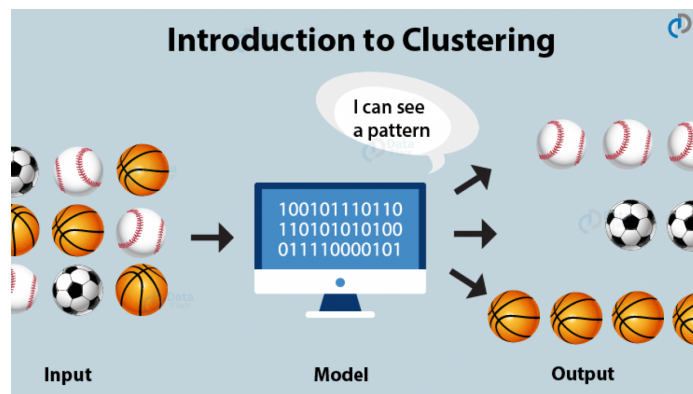
**Figura 2.2:** Esempio di Regressione per il calcolo del valore di un immobile.

## Clustering

I problemi di tipo *clustering* sono generalmente risolti con metodi di apprendimento non supervisionato, in quanto richiedono di individuare dei *pattern*<sup>[9]</sup> all'interno dei dati per poterli suddividere in gruppi, che non sono noti a priori. Alcuni esempi di problemi di *clustering* riguardano la suddivisione di libri per area tematica o contenuti e la segmentazione dei clienti di un *eCommerce*<sup>[9]</sup> in base alle loro abitudini d'acquisto a fini commerciali. Un esempio del problema di clustering viene mostrato in Figura 2.3.

<sup>1</sup>Fonte Figura 2.2: <https://medium.com/@savannahar68/getting-started-with-regression-a39aca03b75f>.

<sup>1</sup>Fonte Figura 2.3: <https://thedailychronicle.in/news/622386/clustering-algorithms-market-2020-28-data-preparat>



**Figura 2.3:** Esempio di *Clustering* per la suddivisione di un insieme di dati in sottogruppi.

## 2.2 Microsoft ML.NET

*ML.NET* è un [framework open source](#)<sup>[9]</sup>, [multi-piattaforma](#)<sup>[8]</sup> e gratuito disponibile per la piattaforma di sviluppo *.NET* e permette di aggiungere funzionalità di *machine learning*, in scenari *online* o *offline*. *ML.NET* consente di allenare, costruire ed esportare modelli di *machine learning* personalizzati utilizzando i linguaggi di programmazione *C#* o *F#*<sup>2</sup> per una vasta gamma di scenari. Il [framework](#) rende anche disponibili dei *tools* come il *Model Builder*<sup>3</sup> per permettere l'integrazione del *machine learning* in applicazioni *.NET* in modo semplice e veloce.

Al centro di *ML.NET* c'è il concetto di modello. Il modello specifica la sequenza di trasformazioni necessarie da applicare a un insieme di dati in *input* per ottenere il risultato di una predizione. Con *ML.NET* è possibile allenare e creare un modello personalizzato specificando un particolare algoritmo oppure importare un modello già pronto.

Come accennato in precedenza, *ML.NET* offre tre alternative per il suo utilizzo: il *Model Builder tool* disponibile solo per Windows; la *ML.NET Command Line Interface*; e la *ML.NET API*<sup>4</sup> che permette di utilizzare anche le funzioni più specifiche.

### Model Builder

Il *Model Builder tool* è un'estensione dell'*IDE Visual Studio*<sup>5</sup> che una volta scaricata e installata permette di costruire e allenare modelli di *machine learning* in modo semplice e intuitivo, come è possibile notare dall'anteprima disponibile in Figura 2.4. Attualmente è disponibile solo per il sistema operativo Windows. Al fine di aumentare il grado di affidabilità e precisione del modello, il *Model Builder* utilizza *AutoML* (automated Machine Learning) per esplorare vari algoritmi di *machine learning* con differenti configurazioni e trovare il migliore che si adatta al proprio scenario. I tipi di problemi che è possibile risolvere con questo strumento sono: *Text classification* per l'analisi di dati in formato testuale, *Value prediction* che comprende tutti i problemi di Regressione e *Image classification* per l'analisi di immagini.

<sup>2</sup>F#: <https://docs.microsoft.com/it-it/dotnet/fsharp/>.

<sup>3</sup>Model Builder Tool: <https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet/model-builder>.

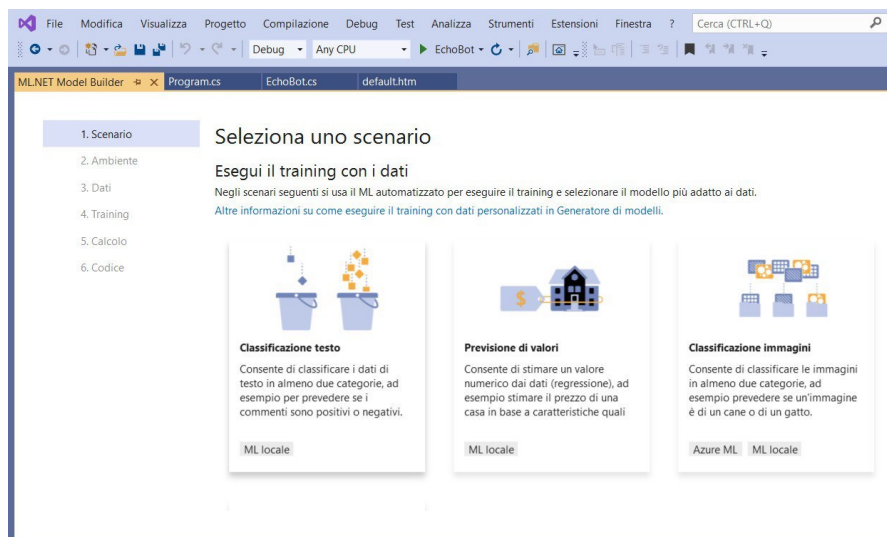
<sup>4</sup>ML.NET API: <https://docs.microsoft.com/it-it/dotnet/machine-learning/>.

<sup>5</sup>Visual Studio: <https://visualstudio.microsoft.com/it/vs/>.

Per creare un modello utilizzando il *Model Builder* è sufficiente seguire i seguenti *steps*:

1. installare l'estensione di *Visual Studio: Model Builder*
2. Selezionare il progetto da utilizzare e scegliere l'opzione Aggiungi — Machine Learning — Scegli scenario (Text classification, Value prediction, Image classification e Recommendation)
3. Selezionare il file contenente i dati da modellare, che può essere in formato CSV, TSV o una tabella di un [database](#)<sup>[9]</sup>
4. Scegliere il tempo massimo di ricerca del modello migliore
5. Controllare la percentuale di accuratezza dei test effettuati sul modello

Al termine di queste operazioni verrà generato un *file zip* contenente il modello e il codice necessario per utilizzarlo in un'altra applicazione *.NET*.



**Figura 2.4:** Esempio di utilizzo del *Model Builder*.

## ML.NET CLI

*ML.NET CLI* è un *.NET Core tool* a riga di comando che svolge le stesse funzioni del *Model builder*, ma è disponibile anche per Mac e Linux. Il comando per la creazione di un modello si chiama *mlnet* e deve essere seguito dal tipo di *task* che si vuole svolgere (*classification*, *regression* o *recommendation*) e un elenco di altri parametri opzionali che vengono riportati di seguito:

- \* **—dataset [path]** (Obbligatorio) percorso del [dataset](#)<sup>[9]</sup> nel *file system*
- \* **—label-col [col]** (Obbligatorio) indice della colonna che contiene il valore di nostro interesse

- \* **—has-header [true/false]** indica se la prima riga del *dataset* contiene l'intestazione del foglio di calcolo
- \* **—ignore-cols [cols]** per ignorare alcune colonne di poco conto o che non servono per la definizione del modello
- \* **—train-time [sec]** tempo massimo (in secondi) per esplorare i modelli con le migliori configurazioni

## ML.NET API

E' infine possibile specificare manualmente l'algoritmo da utilizzare per la creazione del modello e normalizzare i dati forniti in *input* al fine di ottenere un modello personalizzato. La documentazione di *ML.NET* fornisce tutte le informazioni necessarie per avere il pieno controllo sulle tecniche che vengono usate per creare il modello. Nella sezione seguente verranno trattate tutte le fasi necessarie per ottenere un modello a partire dalla preparazione dei dati iniziale.

## 2.3 Documentazione di ML.NET API

Come detto in precedenza, L'*API* (Application Program Interface) di *ML.NET* offre la possibilità di aggiungere funzioni di *machine learning* in applicazioni *.NET*, lasciando allo sviluppatore la scelta dei parametri e delle tecniche da utilizzare per il *training* del modello. Una volta creato, sarà sufficiente importarlo nel progetto di destinazione per poterlo utilizzare e fare delle previsioni.

Durante la creazione del modello ci sono due fattori importanti da tenere in considerazione: la preparazione dei dati e la valutazione del modello.

### 2.3.1 Preparazione dei dati

Nella maggior parte dei casi, i dati che si hanno a disposizione non possono essere utilizzati direttamente per creare un modello, ma devono essere pre-processati correttamente: questa fase è anche detta *data preprocessing*. L'operazione più comune è quella di pulizia dei dati corrotti o incompleti, che vengono eliminati o integrati con valori di *default* o valori medi. Un altro rischio da tenere in considerazione è la possibilità che alcune proprietà dei dati potrebbero assumere dei valori molto elevati rispetto alle altre, e ciò porterebbe l'algoritmo utilizzato per il *training* a dare maggior peso a tali proprietà erroneamente. Proprio per questo motivo, è buona prassi normalizzare i dati che si vogliono analizzare.

### 2.3.2 Valutazione del modello

Una fase che non deve essere trascurata è quella della valutazione del modello, che permette di ottenere una stima inerente al grado di precisione delle predizioni del modello stesso. Per farlo, è necessario dividere l'insieme dei dati che si vogliono elaborare (che chiameremo d'ora in poi *dataset*) in 2 gruppi, il *training set* e il *test set* con un rapporto del 70% - 30% rispettivamente. L'insieme dei dati contenuti nel *training set* verrà utilizzato per la creazione del modello, mentre i dati contenuti nel *test set* verranno utilizzati per calcolare delle previsioni con il modello generato e per stabilire l'accuratezza della previsione confrontando il risultato atteso con quello restituito dal modello.

Nel caso in cui il *dataset* non sia molto consistente, è invece preferibile utilizzarlo interamente per la creazione del modello evitando di escludere parte dei dati a fini di test.

## 2.4 Microsoft Bot Framework

Il *Microsoft Bot Framework* rende disponibili i *tools* necessari per creare, testare e gestire *bot* intelligenti. Grazie all'integrazione di servizi di intelligenza artificiale è possibile sviluppare dei *bot* che riescano a comprendere il linguaggio umano e rispondere alle domande.

I *bot* permettono di offrire agli utenti un'interfaccia che simula l'interazione con una persona reale. Ultimamente sono diventati molto popolari e trovano un utilizzo su larga scala nella gestione delle prenotazioni di ristoranti e voli, e nel fornire informazioni sugli ordini effettuati negli *eCommerce*. Gli utenti interagiscono con il *bot* come in una normale conversazione, mandando messaggi di testo o dei messaggi vocali.

E' possibile sviluppare un *bot* in diversi linguaggi di programmazione, come il *C#*, *Javascript*<sup>6</sup> e *Python*<sup>7</sup> e, una volta completato, è disponibile il *Bot Framework Emulator* per testare il bot in locale e vedere se risponde correttamente. Una anteprima del Bot Framework Emulator è disponibile in Figura 2.5.

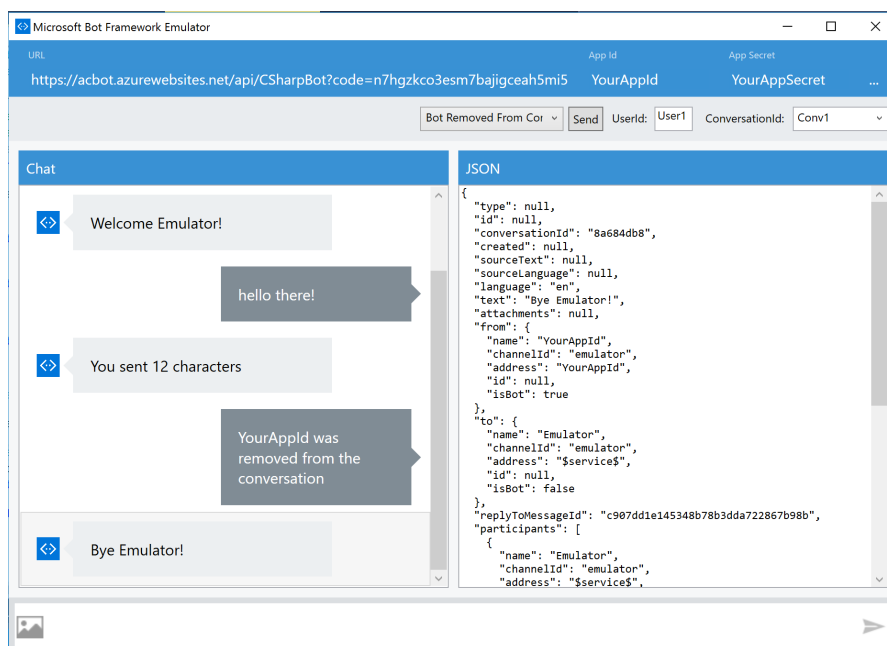


Figura 2.5: Interfaccia del *Microsoft Bot Framework Emulator*.

<sup>6</sup>Javascript: <https://developer.mozilla.org/it/docs/Web/JavaScript>.

<sup>7</sup>Python: <https://www.python.org>.

### Come sviluppare un bot

Un *bot* è un'applicazione che offre agli utenti una interfaccia simile ad una conversazione consentendo l'utilizzo di messaggi di testo, vocali e alcune schede personalizzate per raccogliere *feedback*. Ogni interazione tra il *bot* e l'utente genera un'*activity*, che è il componente principale della comunicazione. Ci sono vari tipi di *activity*, tra cui i *conversation update*, inviati per notificare specifici eventi, e i *message*, messaggi di testo scambiati all'interno della conversazione. I *message* possono inoltre contenere delle *cards* da mostrare all'utente e delle *suggested actions*, ovvero un insieme di "risposte rapide" sotto forma di bottoni. Questa funzione permette di ottenere una specifica risposta da parte dell'utente in modo da poter essere riconosciuta dal *bot* ed elaborata.

Un altro concetto importante è quello di *Turn*, ovvero un turno della conversazione. Un *Turn* consiste nell'*activity* in arrivo dall'utente al *bot* e l'*activity* con cui il *bot* risponde.

## 2.5 Requisiti del progetto

Al termine della fase di studio ho scritto una relazione per illustrare le funzionalità offerte dal *machine learning* e le modalità di interazione disponibili offerte dal *bot*. Abbiamo così definito i requisiti funzionali e non funzionali del prodotto finale, elencati in Tabella 2.1 e Tabella 2.2:

**Tabella 2.1:** Requisiti funzionali.

ID	Requisito
1	L'assistente virtuale dovrà indovinare la categoria di appartenenza della domanda e rispondere coerentemente
1.1	L'assistente virtuale dovrà chiedere conferma all'utente della previsione fatta
1.2	L'assistente virtuale dovrà chiedere all'utente a quale categoria sta facendo riferimento in caso di indecisione tra più categorie
2	L'assistente virtuale dovrà salvare tutte le nuove domande per migliorare il modello

**Tabella 2.2:** Requisiti non funzionali.

ID	Requisito
1	È richiesta la creazione manuale di un <i>dataset</i> di domande categorizzate per il training del modello
2	La costruzione del modello di <i>Machine Learning</i> deve essere affidata al <i>framework open source ML.NET</i>
3	L'assistente virtuale dovrà essere integrata all'interno del <i>sito web</i> del prodotto (opzionale)



## 2.6 Esempio di utilizzo del bot

Il bot cerca di risolvere i problemi riscontrati dai clienti della *suite* di prodotti Plain fornendo il *link* al manuale utente del prodotto specifico a cui fa riferimento l'utente nella richiesta di supporto. L'utente potrà dunque provare a risolvere il problema riscontrato consultando le guide passo passo e le informazioni presenti sull'utilizzo delle varie funzioni del prodotto.

Lo scenario più comune è quello in cui l'utente digita la domanda che vuole porre al *bot*, il *bot* elabora una previsione cercando di stabilire a quale prodotto della *suite* Plain si sta riferendo l'utente e risponde inviando per messaggio il link al manuale utente relativo al prodotto. L'utente proverà così a risolvere il suo problema in modo attivo consultando il manuale e, se ha successo, dà un *feedback* positivo al *bot* in modo che la domanda formulata originariamente dall'utente venga salvata e possa essere utilizzata per migliorare il modello di previsione del prodotto. In caso contrario, il *bot* invita l'utente a mandare una mail al servizio clienti.

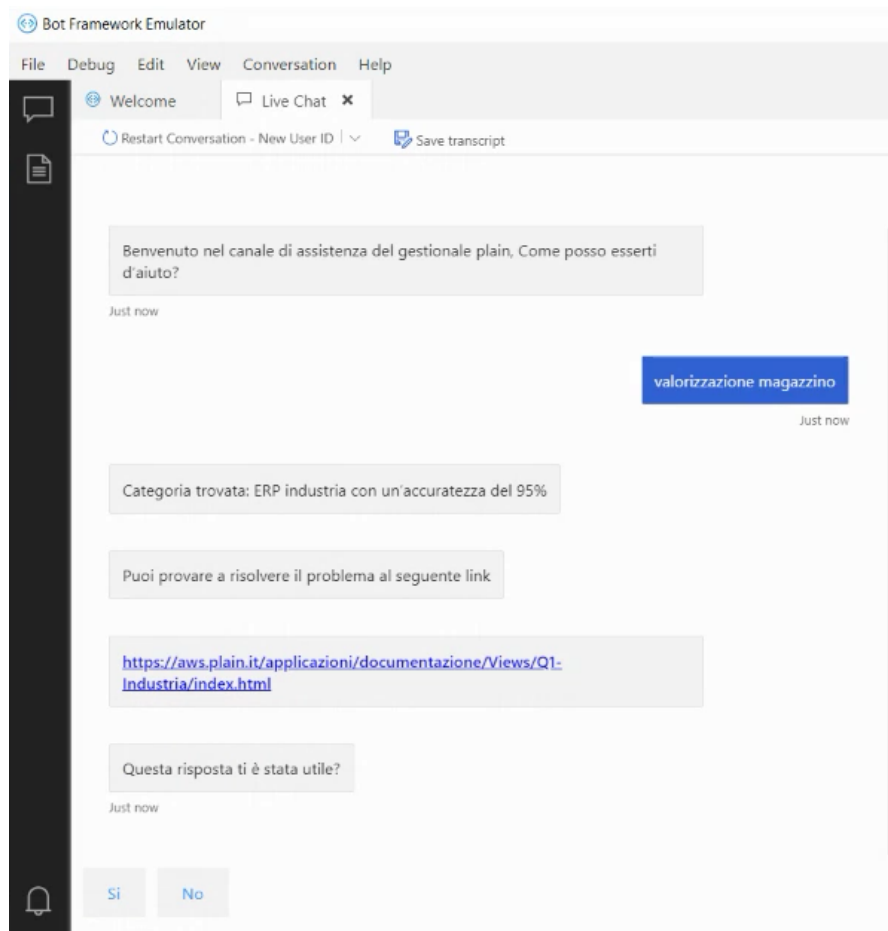
Può però succedere che la domanda formulata dal cliente sia ambigua e il modello restituisca uno o più prodotti con una percentuale di precisione molto simile. In questo caso, viene richiesta all'utente la disambiguazione del prodotto a cui sta facendo riferimento. Una volta ottenuto il prodotto, il *bot* invia il link al relativo manuale utente come di consueto. Anche in questo caso, la domanda viene salvata solo se l'utente manda un *feedback* positivo al termine della conversazione. Un esempio di utilizzo del bot viene mostrato in Figura 3.8.

Per definire in modo formale i comportamenti del prodotto finale, sono stati stilati i casi d'uso per ciascuna possibile situazione di seguito riportati nelle Tabelle 2.3, 2.4, 2.5, 2.6, 2.7:

### Casi d'uso

**Tabella 2.3:** UC 1 - Risoluzione di un problema riscontrato nel prodotto.

Attori	Utente
Descrizione	L'utente vuole risolvere un problema riscontrato con il prodotto
Pre-condizione	L'utente visualizza un messaggio di benvenuto
Post-condizione	L'utente ha ottenuto una possibile soluzione al problema
Scenario principale	L'utente digita una domanda relativa al prodotto L'utente riceve la possibile soluzione al problema
Inclusioni	Feedback da parte dell'utente sulla soluzione proposta (UC 1.1)
Estensioni	Il sistema è indeciso sulla categoria del problema (UC 2) Il sistema non riesce a elaborare una possibile soluzione al problema (UC 3)



**Figura 2.6:** Esempio di utilizzo dell'assistente virtuale.

**Tabella 2.4:** UC 1.1 - Feedback da parte dell'utente sulla soluzione proposta.

Attori	Utente
Descrizione	Il sistema chiede all'utente se la soluzione proposta è stata efficace
Pre-condizione	L'utente riceve una possibile soluzione al problema
Post-condizione	L'utente invia il feedback sulla soluzione proposta
Scenario principale	Il sistema chiede all'utente se la soluzione proposta è stata efficace L'utente invia il feedback scegliendo l'opzione corretta
Inclusioni	-
Estensioni	L'utente non riesce a risolvere il problema con la soluzione proposta (UC 1.1.1)

**Tabella 2.5:** UC 1.1.1 - L'utente non riesce a risolvere il problema con la soluzione proposta.

Attori	Utente
Descrizione	L'utente non è soddisfatto dalla soluzione proposta
Pre-condizione	L'utente riceve una possibile soluzione al problema
Post-condizione	L'utente viene sollecitato a mandare una mail al supporto clienti
Scenario principale	L'utente preme l'opzione a video "No" L'utente riceve a schermo la mail per contattare il supporto clienti
Inclusioni	-
Estensioni	-

**Tabella 2.6:** UC 2 - Il sistema è indeciso sulla categoria del problema.

Attori	Utente
Descrizione	Il sistema è indeciso sulla categoria del problema e richiede maggiori informazioni all'utente
Pre-condizione	L'utente ha inviato la sua richiesta al sistema
Post-condizione	L'utente ottiene la soluzione al problema in base alla categoria che ha fornito al sistema
Scenario principale	Il sistema chiede all'utente di quale categoria fa parte il problema L'utente seleziona la categoria inerente al problema Il sistema fornisce a video una possibile soluzione
Inclusioni	Feedback da parte dell'utente sulla soluzione proposta (UC 1.1)
Estensioni	-

**Tabella 2.7:** UC 3 - L'utente non riceve una soluzione al problema.

Attori	Utente
Descrizione	L'utente non riceve nessuna soluzione possibile e viene sollecitato a contattare il supporto clienti
Pre-condizione	L'utente vuole risolvere un problema riscontrato con il prodotto
Post-condizione	L'utente viene sollecitato a mandare una mail al supporto clienti
Scenario principale	L'utente riceve a schermo la mail per contattare il supporto clienti
Inclusioni	-
Estensioni	-



## Capitolo 3

# Fase di Progettazione

### 3.1 Creazione dei primi prototipi

Una volta definite chiaramente le funzionalità e le modalità di interazione del *bot*, ho iniziato la fase di progettazione. Il tipo di problema che dovevo affrontare era la *Multiclass classification*, ovvero riuscire a riconoscere la categoria di appartenenza di un oggetto a partire da un insieme di categorie note. Nello specifico, infatti, bisogna individuare il tipo di prodotto a cui fa riferimento il cliente nella sua richiesta di supporto.

Per prendere un po' di dimestichezza con il *machine learning*, ho creato un prototipo per risolvere un problema di classificazione utilizzando la *command-line interface* di *ML.NET*. Dopo una ricerca online dei vari *dataset* disponibili, ne ho scelto uno relativo al famoso *servizio web Yahoo Answers*<sup>1</sup>, in cui qualunque utente può porre una domanda e gli altri utenti della piattaforma possono rispondervi ottenendo punti e popolarità. Il *dataset* contiene circa un milione di domande tra quelle presenti sul sito così strutturate: titolo della domanda, descrizione della domanda, risposta, categoria della domanda.

La *ML.NET CLI* permette di elaborare automaticamente un modello di *machine learning* a partire da un *dataset* e uno scenario di applicazione - nel nostro caso la *multiclass classification*. Per farlo, utilizza la *engine AutoML* (automated Machine Learning) che, in base allo scenario del problema da risolvere, confronta un insieme di algoritmi di apprendimento per poi scegliere quello che dà il risultato migliore. Per creare il modello eseguiamo il comando

```
mlnet classification -dataset yahoo_train_set.csv -label-col 0
                        -has-header false -train-time 150
```

Una volta terminato, il comando stampa a schermo gli algoritmi di *training* che ha testato, salva il modello generato in un *file zip* e crea un nuovo progetto in *C#* con un esempio di codice preconfezionato che può essere utilizzato per caricare il modello elaborato ed effettuare delle previsioni. Le metriche più significative da tenere in considerazione sono la *micro accuracy* e la *macro accuracy*, che indicano rispettivamente la frazione di istanze classificate correttamente e la media delle probabilità che una previsione sia corretta per una specifica categoria. Un esempio di utilizzo del comando *mlnet* viene fornito in Figura 3.1.

Il livello di accuratezza delle metriche mostrate a schermo era molto deludente, con appena un 40% di precisione, ma, utilizzando il codice generato automaticamente per

---

<sup>1</sup>Yahoo answers: <https://it.answers.yahoo.com>.

fare delle previsioni con alcune domande scritte a mano, il modello è sempre riuscito a indovinare la giusta categoria delle domande con un indice di precisione di più dell'85% in media.

```

gianluigipelle@Gianluigi-MacBook-Pro ~/Desktop/ml-net cli
$ mlnet classification --dataset yahoo_train_set.csv --label-col 0 --has-header false --train-time 150
Start Training
|  Trainer          MicroAccuracy  MacroAccuracy  Duration #Iteration  |
|1  AveragedPerceptronOva      0.4062      0.3911      11.9      1
| dcaMaximumEntropyMulti      0.4160      0.3205      56.5      2
|-----|-----|-----|-----|
=====Experiment Results=====
|-----|-----|-----|-----|
| Summary |
|-----|-----|-----|-----|
|ML Task: multiclass-classification
|Dataset: /Users/gianluigipelle/Desktop/ml-net cli/yahoo_train_set.csv
|Label : col0
|Total experiment time : 68.4089382 Secs
|Total number of models explored: 2
|-----|-----|-----|-----|
| Top 2 models explored |
|-----|-----|-----|-----|
|  Trainer          MicroAccuracy  MacroAccuracy  Duration #Iteration  |
|1  SdcaMaximumEntropyMulti      0.4160      0.3205      56.5      1
|2  AveragedPerceptronOva      0.4062      0.3911      11.9      2
|-----|-----|-----|-----|
Code Generated
Generated C# code for model consumption: /Users/gianluigipelle/Desktop/ml-net cli/SampleClassification/SampleClassification.ConsoleApp
Check out log file for more information: /Users/gianluigipelle/.mlnet/log.txt
Exiting ...
gianluigipelle@Gianluigi-MacBook-Pro ~/Desktop/ml-net cli

```

**Figura 3.1:** Esempio di utilizzo della *ML.NET CLI*.

Dopo aver elaborato qualche previsione con il nuovo modello, ho iniziato a sviluppare il prototipo di un *bot* in grado di inviare un messaggio di benvenuto ai nuovi utenti, elaborare il testo dei messaggi ricevuti e rispondere a questi in modo appropriato. Ho riutilizzato il codice che era stato generato insieme al *training* del modello per elaborare una previsione a partire dal testo contenuto nel messaggio dell'utente. Ad ogni messaggio in arrivo, infatti, il *bot* carica il modello, elabora una previsione con il testo ricevuto e risponde all'utente mostrando la categoria di appartenenza della domanda formulata dall'utente con la relativa percentuale di accuratezza.

## 3.2 Creazione del Dataset

Dopo aver acquisito esperienza nella creazione dei modelli grazie allo sviluppo dei prototipi, sono passato allo sviluppo del prodotto finale. Come detto in precedenza, il componente essenziale per l'elaborazione di un modello di *machine learning* sono i dati, che permettono alla macchina di apprendere in modo automatico. Sfortunatamente, però, l'azienda non disponeva di alcuna informazione relativa ai *ticket* di assistenza del supporto clienti, in quanto non venivano salvati in modo permanente. Per sopperire a questa mancanza, ho dovuto creare a mano il *dataset* dedicandoci un'intera settimana della fase di progettazione. Abbiamo scelto i 5 prodotti più utilizzati della *suite* Plain (ERP industria, ERP commercio, ERP A.F.C., ERP E-fattura e la piattaforma che gestisce i flussi delle fatture elettroniche) e ho scritto a mano circa 150 domande verosimili per ciascun prodotto. Ogni riga del foglio di calcolo utilizzato per salvare il *dataset* contiene la categoria di appartenenza e una possibile domanda che è stata posta al supporto tecnico. Un esempio del *dataset* creato è illustrato di seguito in Figura 3.2.

AsiBot dataset	
category	Question
ERP industria	Non riesco a capire come funziona la procedura di gestione dell'inventario, potete aiutarmi?
ERP industria	Non riesco a trovare il Menu magazzini dal gestionale, mi date una mano?
ERP industria	Ho eseguito la funzione di Stampa rilevazioni inventario ma non funziona, che fare?
ERP industria	Ho eseguito la funzione di Ricalcolo inventario contabile alla data ma non funziona, che faccio?
ERP industria	Stampa rilevazioni inventario mi mostra un elenco di giacenze errato, dovrebbero essere molte meno, perché?
ERP industria	Ho già effettuato dei movimenti di magazzino per l'anno nuovo, c'è ancora qualche modo di usare la funzionalità Stampa rilevazioni inventario?
ERP industria	Stavo eseguendo il Ricalcolo inventario contabile alla data ma il programma non risponde e la rotella di caricamento continua a girare, che devo fare?
ERP industria	Abbiamo effettuato l'inventario fisico ma non riusciamo a caricare i dati nel gestionale, aiuto
ERP industria	Il pulsante Caricare i dati di inventario fisico non va, e ora?
ERP industria	Non riesco a creare un movimento di rettifica attraverso il programma, come faccio?
ERP industria	Ho aperto il gestionale ed è scomparso il Menu Magazzini, devo aggiornare il programma?
ERP industria	Dopo aver aggiunto un movimento di rettifica il programma si è chiuso da solo bruscamente, e ora?
ERP industria	Ho inserito un nuovo movimento di rettifica ma non è stata associata la corretta causale, perché?
ERP industria	Siamo a fine anno e non riesco a capire come funziona la procedura di valorizzazione di magazzino, aiuto!
ERP industria	Come si calcolano i dati per il LIFO?
ERP industria	Come si calcolano i costi medi annuali dei materiali di acquisto?
ERP industria	Come si calcolano i costi medi dei semilavorati e dei prodotti finiti?
ERP industria	Il bottone elaborazioni periodiche è presente ma non è possibile premerlo perché?
ERP industria	Ho effettuato una modifica a un movimento di magazzino nell'anno dopo aver eseguito la valorizzazione di magazzino, che devo fare adesso?
ERP industria	Sto avendo problemi con la procedura di Stampe di magazzino valorizzate, cosa devo fare?
ERP industria	La funzione di valorizzazione a LIFO non mi calcola i valori, nonostante tutte le causali dei di magazzino siano di tipo C+1 e C+2, non capisco perché
ERP industria	Come faccio a controllare il risultato della valorizzazione LIFO prima di confermare l'operazione e stampare il documento?
ERP industria	Non riesco a stampare il documento LIFO per articolo, è disponibile solo l'opzione P per la stampare provvisoria
ERP industria	Non riesco a eseguire una nuova valorizzazione per il nuovo anno, mamma mia
ERP industria	Sto calcolando la valorizzazione a costo medio ma non tutti i movimento vengono presi in considerazione nel risultato, perché?
ERP industria	Nei movimenti del nostro magazzino abbiamo alcuni movimento con tipo causale Scarico ma che non sono Resi, come possiamo aggiungerli ai fini del calcolo della valorizzazione? Grazie

Figura 3.2: Esempio del *dataset* creato manualmente.

### 3.3 Definizione dei moduli

Per organizzare in modo più modulare il codice, ho deciso di creare due progetti distinti: uno per gestire le funzionalità del *machine learning* e l'altro per lo sviluppo del *bot*.

#### 3.3.1 Modulo ML

Il modulo ML, illustrato in Figura 3.3, si compone di 2 classi:

**MLBuilder** si occupa della creazione di un modello personalizzato, della valutazione della sua accuratezza e del salvataggio dello stesso in un *file zip*

**MLManager** permette di effettuare delle previsioni specificando il modello da utilizzare e ha delle funzioni di comodo per determinare il nome della categoria di appartenenza e se la previsione sia affidabile

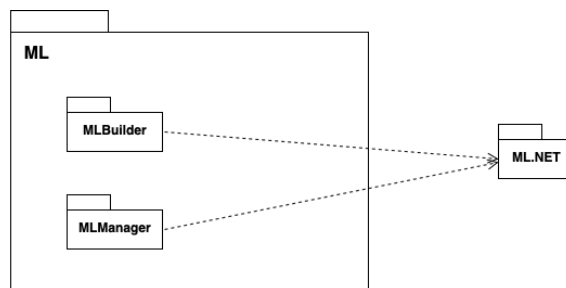


Figura 3.3: Diagramma del modulo ML.

## Classe MLBuilder

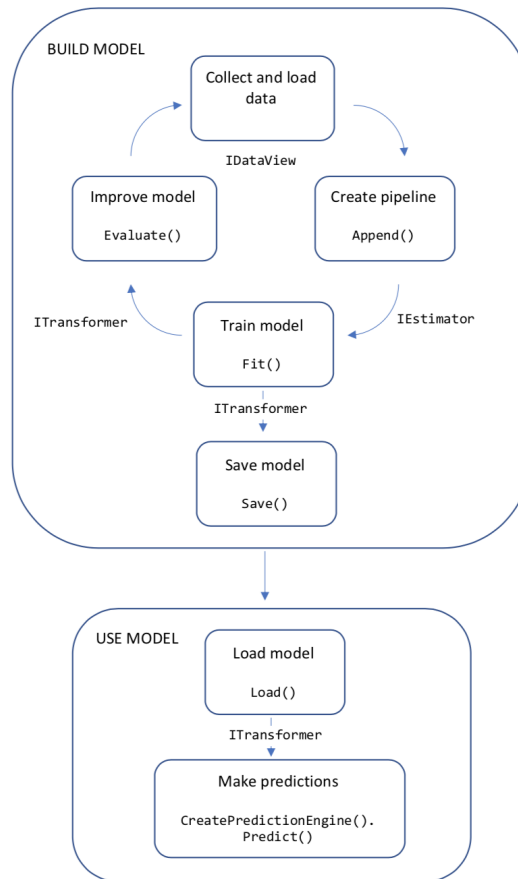
MLBuilder
- trainDataPath: string - testDataPath: string - modelPath: string - mlContext: MLContext
+ CreateModel(): void + CreateModelPipeline(): IEstimator<ITransformer> + TrainAndSaveModel(IDataView, IEstimator<ITransformer>): void + EvaluateModel(): void + Evaluate(ITransformer): void + MakePrediction(): void + ConvertScoresArrayIntoDictionary(DataViewSchema, string, float[]): Dictionary<string, float>

Figura 3.4: Diagramma della classe MLBuilder.

*MLBuilder* si occupa della creazione di un modello di *machine learning* personalizzato e della sua valutazione, come illustrato dal diagramma della classe presente in Figura 3.4. Il *training* di un modello richiede una sequenza di operazioni da eseguire in modo iterativo, come illustrato di seguito in Figura 3.5:

- \* Raccogliere e caricare i dati del *train set* in un oggetto di tipo *IDataView*
- \* Specificare una sequenza di operazioni per estrarre le *Features* e applicare un algoritmo di apprendimento automatico
- \* Eseguire il training di un modello chiamando il metodo *Fit()* sulla [pipeline](#)<sup>[g]</sup>
- \* Valutare il modello ed eseguire una nuova iterazione per provare a migliorarlo
- \* Salvare il modello in formato binario per poterlo utilizzare in un altro progetto
- \* Caricare il modello in un oggetto di tipo *ITransformer*
- \* Fare delle previsioni chiamando il metodo *CreatePredictionEngine.Predict()*





**Figura 3.5:** Sequenza delle operazioni necessarie per creare un modello personalizzato.

Per ciascuna di queste operazioni è stato creato un metodo all'interno della classe *MLBuilder*, in modo da poter modificare singolarmente una specifica fase senza doversi preoccupare delle altre. L'utilizzo della classe è molto semplice e permette di creare un modello, valutare un modello esistente e fare delle previsioni. Per ciascuna di queste operazioni è sufficiente impostare alcuni parametri correttamente e chiamare i metodi necessari all'interno della funzione *Main()* della classe, come viene spiegato di seguito:

### Creare un nuovo modello

Per generare un nuovo modello personalizzato è sufficiente seguire queste istruzioni:

1. Importare il *file CSV* contenente i dati del [dataset](#) nella cartella Data del progetto
2. Impostare il percorso del file appena importato nella variabile statica *trainData-Path*

<sup>1</sup>Fonte figura 3.5: <https://docs.microsoft.com/it-it/dotnet/machine-learning/how-does-mldotnet-work>.

3. Impostare il percorso dove si vuole salvare il modello che verrà generato, nella variabile statica *modelPath*
4. Assegnare al campo dati *mlContext* un nuovo oggetto *MLContext()* all'interno della funzione *Main()*
5. Chiamare il metodo *CreateModel()* all'interno della funzione *Main()*
6. Scegliere la *pipeline* di operazioni che deve essere restituita dal metodo *CreateModelPipeline()* tra quelle presenti, oppure specificarne una nuova

Al termine di tutte queste operazioni sarà sufficiente eseguire il progetto e verrà creato un nuovo modello nel percorso specificato nello step 3.

### Valutare un modello

Per valutare un modello già esistente o appena creato è sufficiente eseguire queste istruzioni:

1. Impostare il percorso del modello da valutare nella variabile statica *modelPath*
2. Impostare il percorso del file contenete i dati del *test set* nella variabile statica *testDataPath*
3. Chiamare il metodo *EvaluateModel()* all'interno della funzione *Main()*

Al termine di tutte queste operazioni sarà sufficiente eseguire il progetto e verranno mostrate a schermo le principali metriche dei test effettuati sull'accuratezza del modello.

### Calcolare una previsione

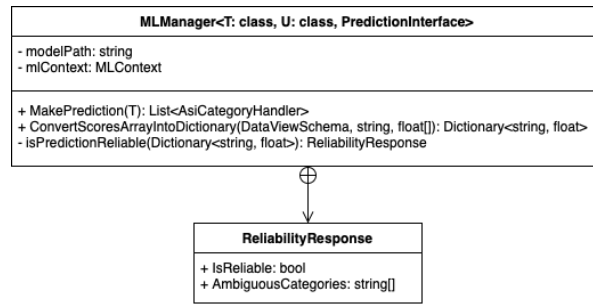
Per calcolare una previsione su un modello esistente è sufficiente seguire queste istruzioni:

1. Impostare il percorso del modello da utilizzare nella variabile statica *modelPath*
2. Creare un oggetto da mandare in *input* al modello all'interno del metodo *MakePrediction*
3. Chiamare il metodo *MakePrediction()* all'interno della funzione *Main()*

Al termine di tutte queste operazioni sarà sufficiente eseguire il progetto e verrà mostrata a schermo la previsione del modello insieme alla relativa percentuale di accuratezza.

### Classe MLManager

La classe *MLManager* è una classe generica che semplifica notevolmente l'utilizzo di un modello di *machine learning* per il calcolo di una previsione, come si può vedere dal diagramma della classe presente in Figura 3.6. Per istanziare un oggetto concreto della classe è necessario specificare una classe T per modellare l'oggetto da fornire in *input* al modello, e una classe U che aderisca all'interfaccia *PredictionInterface* per modellare l'oggetto fornito in *output* dal modello e che conterrà la previsione effettuata. Una volta creato un oggetto *MLManager* è possibile chiamare il metodo *MakePrediction(T inputData)* per calcolare una previsione a partire dall'oggetto T.



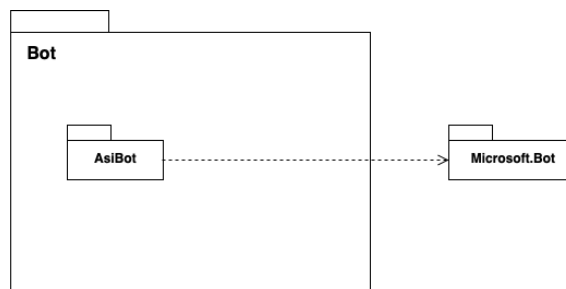
**Figura 3.6:** Diagramma della classe *MLManager*.

Di *default*, il modello restituisce un *array Scores* che contiene la percentuale di accuratezza con cui è associata la domanda in *input* a ciascuna categoria possibile. Per semplificare l'accesso ai singoli *score*, ho creato una funzione *ConvertScoresArrayIntoDictionary()* che converte l'*array Scores* in un dizionario, in modo da poter accedere ai singoli valori attraverso il nome della categoria al posto di un indice numerico. La classe contiene anche un metodo *IsPredictionReliable()* che, a partire dal dizionario di *Scores*, stabilisce se la previsione restituita dal modello sia affidabile o se il modello sia indeciso tra più categorie nel caso in cui le due categorie con indici di affidabilità più alti siano molto simili tra loro.

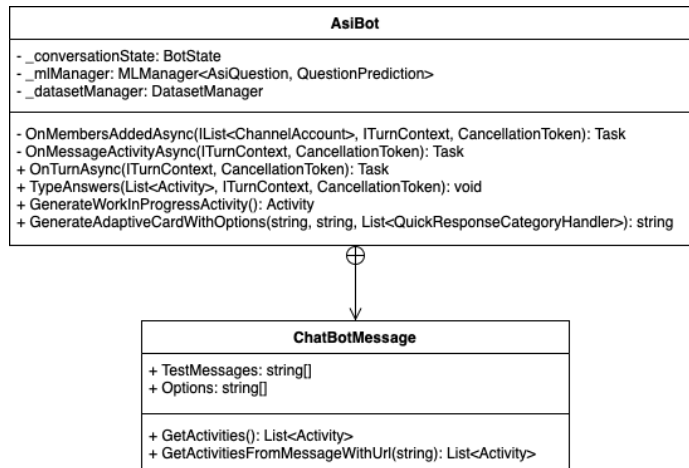
Per uniformare l'*output* fornito dalla classe *MLManager* dopo il calcolo di una previsione, ho definito una struttura *ReliabilityResponse* che contiene come campo dati un *array* di stringhe *AmbiguousCategories* con tutte le categorie tra cui è indeciso il modello in ordine decrescente di precisione e un campo dati *isReliable* - di tipo *booleano* - che indica se la previsione è affidabile controllando se il numero di categorie ambigue è pari a 1.

### 3.3.2 Modulo Bot

Il modulo Bot, illustrato di seguito in Figura 3.7, si compone invece di una sola classe *AsiBot* che gestisce tutti gli eventi in ingresso e in uscita dal *bot* per rispondere ai messaggi in modo appropriato.



**Figura 3.7:** Diagramma del modulo Bot.



**Figura 3.8:** Diagramma della classe *AsiBot*.

### Classe *AsiBot*

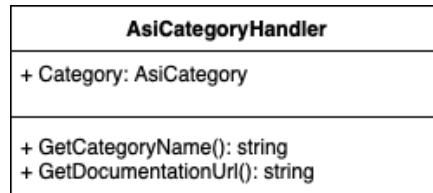
La classe *AsiBot*, mostrata in Figura 3.8, si compone dei seguenti metodi:

- \* **OnMembersAddedAsync()**: viene chiamato ogni volta che un nuovo utente si connette al *bot* e manda un messaggio di benvenuto per accogliere il nuovo utente nella conversazione
- \* **OnTurnAsync()**: viene chiamato all'inizio di ciascun turno della conversazione e il suo compito è quello di gestire le situazioni di errore e inoltrare l'*activity* in arrivo al corretto *activity handler*
- \* **OnMessageActivityAsync()**: gestisce tutte le *activity* di tipo *message* che gli vengono inoltrate dal metodo *OnTurnAsync()*
- \* **TypeAnswers()**: simula il comportamento di una persona reale inviando la lista di messaggi che gli vengono passati in *input* con un intervallo di 2 secondi di pausa, durante il quale è mostrata un'animazione che indica che il *bot* sta scrivendo

Per facilitare la creazione delle *activity* di tipo *message* contenenti *quick responses*, ho creato una struttura *ChatBotMessage* per modellarli facilmente. La struttura si compone di due campi dati: *textMessages* che rappresenta i vari messaggi da inviare all'utente e *options* che rappresenta la lista di opzioni da proporre all'utente che vengono mostrate alla fine dell'ultimo messaggio inviato. Questa scelta progettuale deriva dalla decisione di scomporre i messaggi con i collegamenti *URL* in più messaggi, isolando i singoli *URL* in un messaggio indipendente in modo da favorire un comodo copia incolla e *click* del link. *ChatBotMessage* include il metodo *GetActivities()* che genera un *array* di *activity* a partire dalle informazioni contenute nei propri campi dati, e un metodo *GetActivitiesFromMessageWithUrl()* che, a partire da una stringa di testo, individua i collegamenti riconoscendo il tag *-url-* che viene prefisso e postfisso a ciascun *URL* e restituisce una lista di *activity* separando gli *URL* in messaggi separati.

A causa delle limitazioni presenti negli *enum* del linguaggio *C#* ho creato un oggetto *AsiCategoryHandler* - illustrato in Figura 3.9 - per gestire le varie categorie di prodotto

da individuare. L'oggetto contiene un campo dati *Category* - di tipo *enum* - che indica la categoria a cui si riferisce l'oggetto e due metodi: *GetCategoryName()* che restituisce il nome completo della categoria e *GetDocumentationUrl()* che restituisce il link al manuale utente del prodotto a cui fa riferimento.



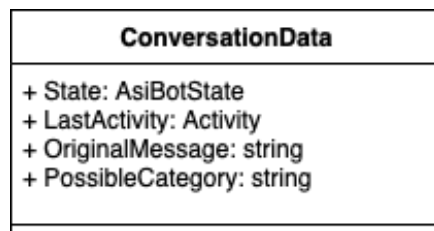
**Figura 3.9:** Diagramma della classe *AsiCategoryHandler*.

### Salvataggio dello stato della conversazione

Un *bot* è un tipo di servizio *stateless*, ovvero senza stato, come tutte le moderne [applicazioni web](#)<sup>[9]</sup>. Il *Bot Framework*, tuttavia, fornisce alcune funzioni per gestire il salvataggio delle informazioni più rilevanti riscontrate nel corso della conversazione. Dal momento che il *bot* viene testato in locale, ho implementato il tipo di archiviazione *memory storage*, consigliato solo a scopo di *test*, in quanto i dati salvati sono cancellati ad ogni riavvio del *bot*. Lo stato viene salvato nelle *state properties*, delle coppie chiave-valore che il *bot* è in grado di leggere e scrivere grazie all'oggetto *state management* che offre un'interfaccia ad alto livello.

Per modellare i dati rilevanti della conversazione ho creato la classe *ConversationData*, illustrata di seguito in Figura 3.10. Il campo dati *originalMessage* contiene il messaggio originale inviato dall'utente all'inizio della conversazione. Il campo dati *lastActivity* contiene l'ultima *activity* che il *bot* ha inviato all'utente e che viene riproposta nel caso in cui l'utente sbaglia a rispondere. Il campo dati *possibleCategory* contiene la categoria del prodotto più appropriata che è stata calcolata dal messaggio originale dell'utente. Infine, per gestire lo stato della conversazione, ho creato un *enum* *AsiBotState* con i seguenti valori:

- \* **Ready:** il *bot* è pronto a gestire una nuova richiesta di supporto
- \* **WaitingDisambiguation:** il *bot* è indeciso su più categorie da assegnare alla domanda e ha chiesto all'utente di indicare a quale di quelle si riferisce il suo problema
- \* **WaitingApproval:** il *bot* ha mostrato al cliente il link al manuale utente del prodotto con cui sta riscontrando dei problemi e sta aspettando un *feedback* se è stato d'aiuto alla risoluzione del problema o meno



**Figura 3.10:** Diagramma della classe *ConversationData*.

## Capitolo 4

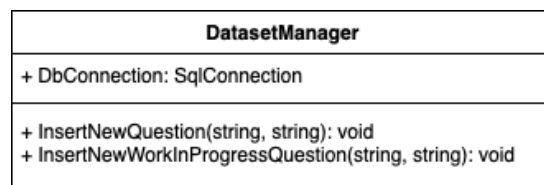
# Contenuti opzionali

### 4.1 Salvataggio delle nuove domande

Terminato lo sviluppo dei requisiti di minima e medi, ho iniziato a lavorare sui requisiti opzionali del prodotto. La prima funzionalità a cui mi sono dedicato è stata il salvataggio delle nuove domande formulate dall'utente, in modo da poter elaborare progressivamente un modello di *Machine Learning* sempre più affidabile.

Per determinare solo le domande risolte grazie all'assistente virtuale, dopo che il *bot* ha fornito all'utente il link al manuale del prodotto, richiede un *feedback* positivo o negativo. Se la risposta è positiva, allora salvo la domanda originale formulata dall'utente insieme alla categoria del prodotto individuata in modo permanente in un *database*, alternativamente non salvo nulla.

Come consigliato dal tutor aziendale, ho utilizzato il servizio di *Database SQL Server Express*<sup>1</sup> di Microsoft, essendo gratuito e facile da utilizzare. Per gestire le funzioni di connessione al *database* e salvataggio dati, ho creato una classe *DatasetManager* - illustrata di seguito in Figura 4.1 - che raccoglie tutto il codice per interfacciarsi con il *database*.



**Figura 4.1:** Diagramma della classe *DatasetManager*.


Il costruttore della classe *DatasetManager* stabilisce una connessione con il *database* per ogni oggetto creato e si occupa della chiusura della connessione nel distruttore della classe. La classe contiene il metodo *InsertNewQuestion()* che salva una nuova domanda all'interno della tabella *Questions* del *database* formulando una *query* in *SQL* di tipo *INSERT* e inoltrando la richiesta al *database*.

<sup>1</sup>SQL Server Express: <https://www.microsoft.com/it-it/sql-server/sql-server-2019>.

## 4.2 Risoluzione dei problemi più frequenti

L'ultima funzionalità aggiunta al *bot* è stata implementata per iniziare a raccogliere dati e poter sviluppare in futuro un nuovo servizio di risoluzione delle problematiche più comuni che si possono riscontrare nell'utilizzo di un qualsiasi prodotto della *suite* Plain. Se l'utente non riesce a venire a capo del problema con il manuale utente fornito dal *bot*, gli viene proposta una serie di soluzioni rapide che risolvono i problemi più comuni e diffusi, come illustrato di seguito in Figura 4.2. Se l'utente riesce a risolvere il problema con una delle soluzioni proposte lascia un *feedback* indicando quale delle soluzioni ha utilizzato. Viene quindi salvata sul *database* la domanda originale dell'utente insieme alla categoria associata alla soluzione utilizzata, utilizzando la funzione *InsertNewWorkInProgressQuestion()* definita nella classe *DatasetManager* approfondita nella Sezione 4.1.

**Aiutaci a migliorare il servizio**

 **Asi Bot**  
Created at 7 giu 2020

Sfortunatamente non siamo riusciti a risolvere il problema da lei richiesto ma stiamo lavorando allo sviluppo di un nuovo servizio di risoluzione dei problemi più comuni. Troverai di seguito una lista di soluzioni comuni per problemi di dominio generale, se una delle soluzioni proposte sia risulti in grado di risolvere il suo problema sarebbe gradito un suo feedback per aiutare lo sviluppo del nuovo servizio, grazie

- A)** Controlla di avere una connessione a internet stabile e funzionante
- B)** Controlla che l'abbonamento al servizio non sia scaduto
- C)** Controlla che la stampante sia collegata correttamente al computer facendo una stampa di prova
- D)** Prova a spegnere e riaccendere il computer e controlla se il problema persiste
- E)** Nessuna delle precedenti

A) Problemi di connessione

B) Servizio scaduto

C) Stampante non disponibile

D) Riavvio manuale del sistema

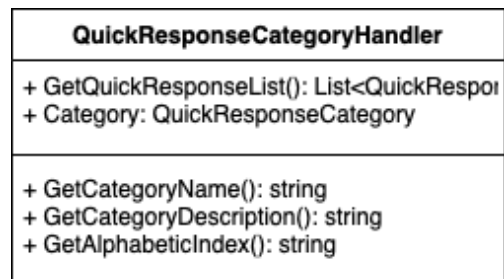
E) Nessuna delle precedenti

**Figura 4.2:** Interfaccia della scheda per raccogliere il *feedback* dell'utente.

Le soluzioni proposte al momento riguardano problemi di connessioni, problemi con la configurazione della stampante, problemi che richiedono il riavvio dell'intero sistema e il controllo se l'abbonamento al servizio offerto dal prodotto sia scaduto.



Queste categorie sono state modellate all'interno della classe *QuickResponseCategoryHandler* - illustrata di seguito in Figura 4.3 - per raggruppare il nome della categoria insieme alla descrizione della soluzione proposta in modo compatto, come fatto in precedenza con la classe *AsiCategory* descritta nella Sezione 3.3.2.



**Figura 4.3:** Diagramma della classe *QuickResponseCategoryHandler*.



## Capitolo 5

# Considerazioni finali

### 5.1 Raggiungimento degli obiettivi

Tutti gli obiettivi definiti nel piano di lavoro dello *stage* sono stati raggiunti, nonostante siano stati modificati alcuni requisiti in corso d'opera insieme al tutor aziendale. Inizialmente, infatti, era stata prevista l'integrazione della *chat* all'interno del *sito web* aziendale, ma si è preferito aumentare le funzionalità offerte dal *bot*. L'azienda, infatti, era più interessata alle funzionalità legate all'intelligenza artificiale del *bot* piuttosto che renderlo subito disponibile sul proprio sito, dal momento che è ancora in fase *beta* e non ha ancora un grado di precisione elevato.

L'ultima funzione sviluppata, ovvero quella per la raccolta dati dei problemi più ricorrentemente riscontrati dagli utenti, è stata aggiunta per quando il *bot* verrà reso disponibile al pubblico. Una volta ottenuti abbastanza dati, sarà sufficiente utilizzare il modulo ML descritto nella Sezione 3.3.1 per elaborare un nuovo modello di *Machine Learning* che riesce a stabilire se il problema a cui fa riferimento l'utente fa parte di uno dei problemi più ricorrenti o meno, e, in caso positivo, mostra all'utente come risolvere tale situazione.

L'obiettivo che ha richiesto l'impegno maggiore è stato quello della creazione manuale del *data set* di domande da utilizzare per allenare il modello. Ho dovuto innanzitutto studiare il manuale utente di ciascuno dei 5 principali prodotti della *suite* Plain, e poi scrivere delle possibili domande inerenti a qualsiasi problema in cui sarei potuto incorrere nell'utilizzo del prodotto, mantenendo uno stile molto colloquiale in quanto dovevano somigliare il più possibile alle domande che verranno formulate in futuro dagli utenti.

### 5.2 Possibili miglioramenti futuri

Le funzionalità offerte dal *bot* attualmente sono molto limitate e richiedono una collaborazione attiva da parte dell'utente, che deve leggere il manuale fornito dall'assistente virtuale e cercare di capire come risolvere il proprio problema; questo, però, è dovuto al fatto che non era disponibile un *dataset* iniziale di domande da elaborare, ma si è dovuto crearne uno manualmente.

Si è quindi deciso di dare molta importanza alla raccolta e salvataggio delle nuove domande formulate dagli utenti in modo da creare un *dataset* consistente per lo sviluppo di future funzionalità di intelligenza artificiale. Nello specifico, si vorrebbe introdurre

la risoluzione istantanea dei problemi più comuni riscontrati nell'utilizzo di un generico prodotto della *suite* Plain, e per farlo abbiamo introdotto le nuove funzionalità illustrate nella Sezione 4.2.

### 5.3 Valutazione personale

Questa opportunità di stage ha arricchito notevolmente la mia esperienza in ambito professionale, permettendomi di entrare in contatto con le dinamiche aziendali, e in ambito formativo, permettendomi di approfondire nuove tecnologie e linguaggi di programmazione che non avevo mai utilizzato in precedenza.

Ho infatti avuto l'opportunità di studiare e approfondire il *Machine Learning*, iniziando con l'utilizzo di un *framework* molto semplice da impostare fino alla realizzazione di un modello personalizzato. Ho dovuto anche imparare il linguaggio di programmazione *C#*, che è largamente utilizzato in ambito aziendale grazie ai numerosi servizi offerti dalla piattaforma *.NET*, e che volevo aggiungere al mio *curriculum* in quanto è molto richiesto nel mondo del lavoro.

Complessivamente sono molto soddisfatto della mia esperienza di *stage* presso ASI srl, in quanto mi ha permesso di approfondire il campo dell'intelligenza artificiale, per cui nutro molto interesse e curiosità, e perchè mi ha lasciato abbastanza libertà nello sviluppo del prodotto privilegiando un confronto costruttivo sulle funzionalità da implementare.

# Glossario

**applicazioni web** indica genericamente tutte le applicazioni accessibili o fruibili via web per mezzo di una network, offrendo determinati servizi all'utente.. [27](#)

**chat bot** è un software che simula una conversazione umana, di natura scritta o parlata, e che permette agli utenti di interfacciarsi al sistema digitale come se stessero comunicando con una persona reale.. [2](#), [4](#)

**database** è un insieme di dati strutturati, ovvero omogeneo per contenuti e formato, che rappresentano un archivio dati in versione digitale.. [11](#), [29](#), [30](#)

**dataset** costituisce un insieme di dati strutturati in forma relazionale, in cui ciascuna colonna della tabella rappresenta una variabile e ogni riga corrisponde ad un determinato membro del dataset in questione.. [11](#), [12](#), [14](#), [19](#), [20](#), [23](#), [33](#)

**eCommerce** indica il processo di acquisto e vendita di prodotti e beni con l'utilizzo di piattaforme digitali, quali le applicazioni mobili e Internet.. [9](#), [13](#)

**framework** è una piattaforma per lo sviluppo di applicazioni software. Fornisce una base a partire dalla quale gli sviluppatori software possono scrivere programmi per specifiche piattaforme.. [3](#), [4](#), [9](#), [10](#), [14](#), [34](#)

**freelance** indica un soggetto che lavora come libero professionista per diverse società o organizzazioni, senza alcun rapporto di dipendenza con esse.. [5](#)

**multi-piattaforma** può essere riferito ad un'applicazione software o ad un linguaggio di programmazione che funziona su più sistemi o piattaforme.. [9](#)

**open source** è un codice progettato per essere disponibile pubblicamente. Può essere modificato e distribuito da chiunque secondo le proprie necessità. Viene sviluppato tramite un approccio decentralizzato e collaborativo.. [9](#), [14](#)

**pattern** è il modo regolare o ripetuto in cui succede qualcosa o viene fatta qualcosa.. [9](#)

**pipeline** indica un insieme di componenti software collegati tra loro in modo sequenziale, in modo che il risultato prodotto da uno degli elementi (output) sia l'ingresso di quello immediatamente successivo (input).. [22](#), [24](#)

**reti neurali** sono modelli matematici realizzati mediante l'utilizzo di neuroni artificiali che si ispirano al funzionamento del cervello umano. Sono indispensabili per risolvere problemi di Intelligenza Artificiale e richiedono avanzati chip hardware a supporto.. [4](#)

**suite** viene usato in informatica per indicare una collezione organizzata di prodotti software, per esempio un gruppo di applicazioni integrate tra di esse.. [1](#), [14](#), [20](#), [30](#)

# Bibliografia

## Siti web consultati

*ASI srl*. URL: <https://www.plain.it/>.

*C#*. URL: <https://docs.microsoft.com/en-us/dotnet/csharp/>.

*Data preprocessing*. URL: <https://docs.microsoft.com/it-it/dotnet/machine-learning/how-to-guides/prepare-data-ml-net>.

*F#*. URL: <https://docs.microsoft.com/it-it/dotnet/fsharp/>.

*IBM*. URL: <https://www.ibm.com/it-it>.

*Intelligenza Artificiale*. URL: <https://www.intelligenzaartificiale.it/machine-learning/>.

*Javascript*. URL: <https://developer.mozilla.org/it/docs/Web/JavaScript>.

*Microsoft Bot Framework*. URL: <https://dev.botframework.com>.

*Microsoft*. URL: <https://www.microsoft.com/it-it/>.

*Microsoft One Drive*. URL: <https://www.microsoft.com/it-it/microsoft-365/onedrive/online-cloud-storage>.

*Microsoft SQL Server*. URL: <https://www.microsoft.com/it-it/sql-server/sql-server-2019>.

*ML.NET*. URL: <https://docs.microsoft.com/it-it/dotnet/machine-learning/how-does-mldotnet-work>.

*ML.NET API*. URL: <https://docs.microsoft.com/it-it/dotnet/machine-learning/>.

*Model Builder tool*. URL: <https://docs.microsoft.com/it-it/dotnet/machine-learning/automate-training-with-model-builder>.

*Modello matematico*. URL: [https://it.wikipedia.org/wiki/Modello\\_matematico](https://it.wikipedia.org/wiki/Modello_matematico).

*Piattaforma .NET*. URL: <https://dotnet.microsoft.com>.

*Plain ERP commercio*. URL: <https://www.plain.it/plain-software-gestione-aziendale-commercio/>.

*Plain ERP industria*. URL: <https://www.plain.it/plain-software-gestionale-per-aziende-manifatturiere/>.

*Python*. URL: <https://www.python.org>.

*Skype*. URL: <https://www.skype.com/it/>.

*Visual Studio*. URL: <https://visualstudio.microsoft.com/it/vs/>.

*Wikipedia*. URL: [https://it.wikipedia.org/wiki/Pagina\\_principale](https://it.wikipedia.org/wiki/Pagina_principale).

*Yahoo Answers*. URL: <https://it.answers.yahoo.com>.