



UNIVERSITÀ DEGLI STUDI DI SALERNO

LAUREA TRIENNALE IN INFORMATICA

CORSO DI FONDAMENTI DI INTELLIGENZA ARTIFICIALE

PROF. F. PALOMBA

AutoMate

Autori

Alessandro Barretta

Sabato Malafronte

Repository GitHub:

<https://github.com/ZTunic/AutoMate>

Indice

1	Introduzione	2
1.1	Il progetto AutoMate	2
1.2	A cosa serve?	2
2	Definizione del problema	3
2.1	Obiettivi	3
2.2	Specifica PEAS	3
2.2.1	Caratteristiche dell'ambiente	4
2.3	Possibili soluzioni	4
2.4	Problematiche del Dataset	4
3	Metodologie di sviluppo	5
3.1	Ingegneria del Machine Learning	5
3.1.1	Il CRISP-DM	5
3.2	Risoluzione del problema	6
3.2.1	Business Understanding	6
3.2.2	Data Understanding	7
3.2.3	Data Preparation	12
3.2.4	Data Modeling	17
3.2.5	Evaluation	19
3.2.6	Deployment	21
4	Conclusioni	22
4.1	Considerazioni finali	22

Capitolo 1

Introduzione

1.1 Il progetto AutoMate

AutoMate è un progetto sviluppato da due studenti dell'Università degli Studi di Salerno per l'Insegnamento di Fondamenti di Intelligenza Artificiale. Il suo principale obiettivo è quello di realizzare un agente intelligente capace di stimare il valore di un'auto usata in base alle caratteristiche più rilevanti.

1.2 A cosa serve?

Capita molto spesso che, volendo mettere in vendita un'auto usata, sia difficile stimare il suo valore. Di solito, infatti, si tende a sovrastimare il suo prezzo reale.

In questo contesto AutoMate si propone di fornire uno strumento volto a risolvere questo tipo di problematiche, conferendo ai venditori un punto di riferimento per effettuare stime sul valore della propria auto, e guidando gli acquirenti a distinguere le offerte più convenienti da quelle meno convenienti.

Capitolo 2

Definizione del problema

2.1 Obiettivi

L'obiettivo fondamentale di AutoMate è quello di realizzare un'applicazione Desktop che gli utenti possono utilizzare per stimare il valore di un'auto usata, fornendo in input le informazioni di quest'ultima (ad esempio marca, potenza, chilometraggio, ecc.). La predizione viene effettuata sulla base di un **Dataset** contenente circa 38.000 istanze di inserzioni caricate su eBay Kleinanzeigen (versione tedesca di eBay Annunci), raccolte nel 2016.

2.2 Specifica PEAS

Un ambiente viene generalmente descritto tramite la formulazione PEAS, ovvero Performance, Environment, Actuators, Sensors.

Performance Fa riferimento alla misura di prestazione adottata per valutare l'operato dell'agente. In questo caso, la misura di prestazione è data dall'accuratezza della predizione rispetto al valore reale dell'auto.

Environment Descrizione degli elementi che formano l'ambiente. In questo caso l'ambiente è costituito dagli utenti che utilizzano l'applicazione e dall'insieme di auto e le loro caratteristiche.

Actuators Gli attuatori disponibili all'agente per intraprendere le azioni. In questo caso l'attuatore corrisponde ad uno schermo utilizzato per mostrare il valore della predizione all'utente.

Sensors I sensori attraverso i quali riceve gli input percettivi. In questo caso il sensore è rappresentato da un Form compilato dall'utente grazie ad una tastiera.

2.2.1 Caratteristiche dell'ambiente

Di seguito sono elencate le caratteristiche principali dell'ambiente:

- **Completamente osservabile:** in ogni momento si ha una completa conoscenza dello stato dell'ambiente.
- **Deterministico:** lo stato successivo dell'ambiente è completamente determinato dallo stato corrente e dall'azione eseguita dall'agente.
- **Episodico:** l'agente sceglie le proprie azioni indipendentemente dalle scelte prese in "episodi" precedenti.
- **Statico:** l'ambiente in cui l'agente opera non cambia nel tempo.
- **Discreto:** l'ambiente fornisce un numero limitato di percezioni e azioni distinte, chiaramente definite.
- **Singolo Agente:** l'ambiente è costituito da un unico agente.

2.3 Possibili soluzioni

Analizzando le caratteristiche del problema, emergono alcune informazioni utili alla sua comprensione.

In primo luogo lo scopo dell'agente è quello di predire un valore definito in un insieme continuo. Questo dettaglio mette in evidenza la natura del problema: si tratta infatti di un problema di regressione. Un'altra caratteristica rilevante è che la predizione verrà effettuata sulla base di molteplici proprietà dell'auto: si deduce quindi che la regressione è multipla.

In riferimento a queste considerazioni risulta quindi evidente che la soluzione più conveniente sia quella di modellare un agente capace di apprendere, e in particolare un regressore, dato che il loro scopo è proprio quello di risolvere problemi di questo tipo,

2.4 Problematiche del Dataset

Per il problema in esame è stata riscontrata un'elevata carenza di Dataset utili alla sua risoluzione. Per questo motivo è stato selezionato un Dataset che, nonostante sia adatto per il raggiungimento degli obiettivi del progetto, contiene dati risalenti al 2016 che rendono il modello di Machine Learning non attuale. È importante notare quindi che i risultati prodotti da AutoMate non sono una stima corretta del valore reale di un'auto rispetto alla situazione corrente del mercato. Inoltre, il prezzo di un'auto dipende da moltissimi fattori, che sono trascurati o comunque non presenti all'interno del Dataset. Ciò rappresenta un'ulteriore elemento che allontana il prezzo stimato dal modello da quello reale.

Capitolo 3

Metodologie di sviluppo

3.1 Ingegneria del Machine Learning

Approcciarsi allo sviluppo di un progetto senza preoccuparsi della sua ingegnerizzazione è uno degli errori più comuni e pericolosi dello sviluppo del Software. Senza una precisa metodologia, infatti, risulta quasi impossibile realizzare un sistema software affidabile o che riesca a fornire le corrette funzionalità agli utenti. È quindi di fondamentale importanza scegliere e seguire un modello di sviluppo del software per evitare (o mitigare) questo tipo di problematiche.

3.1.1 Il CRISP-DM

Quanto detto vale per lo sviluppo di qualunque tipo di software. Tuttavia la realizzazione di un modello di Machine Learning introduce nuove problematiche di cui tenere conto. In particolare, a differenza dei sistemi software tradizionali, nel Machine Learning si fa uso di grandi quantità di dati ed è quindi cruciale preoccuparsi della loro qualità e gestione.

Di conseguenza sono due le cose principali a cui pensare quando si progetta una soluzione basata su Machine Learning: Ingegneria del Software e Ingegneria dei Dati.

Per questi motivi, il modello scelto per progettare AutoMate è il CRISP-DM (Cross-Industry Standard Process for Data Mining), un modello non sequenziale in cui le diverse fasi possono essere eseguite un numero illimitato di volte e che prevede sia i tradizionali processi di Software Engineering che quelli di Data Engineering.

3.2 Risoluzione del problema

Come precedentemente specificato, lo sviluppo del progetto seguirà il modello CRISP-DM. Di seguito sono riportate le fasi del modello, e per ciascuna di esse sono state dettagliatamente descritte le scelte e le operazioni effettuate per raggiungere con successo gli obiettivi prefissati.

3.2.1 Business Understanding

La fase di **Business Understanding** prevede, come tutte le fasi iniziali di un progetto software, attività di raccolta dei requisiti e definizione degli obiettivi di business che si intende raggiungere. Inoltre è necessario determinare la disponibilità delle risorse, stimare i rischi e selezionare le tecnologie e i tool necessari al raggiungimento degli obiettivi.

Obiettivi di business AutoMate ha come obiettivo principe quello di conferire ai suoi utilizzatori uno strumento che sia in grado di stimare il valore di un'auto tenendo conto sia di caratteristiche più generali come marca e modello, sia di informazioni più specifiche come tipo di trasmissione, potenza, tipo di alimentazione.

Disponibilità delle risorse Come già discusso nella **Sezione 2.4** la bassa disponibilità di Dataset ha portato alla selezione di un insieme di dati non molto recente (i dati sono stati estratti nel 2016). Ciò non rappresenta un grosso problema dal punto di vista della realizzazione del modello poiché la struttura di un ideale Dataset più recente sarebbe abbastanza simile. Tuttavia ciò comporta un certo grado di inattualità che compromette l'utilizzo efficace di AutoMate in tempi odierni.

Stima dei rischi Tra i possibili rischi che si potrebbero presentare è opportuno prestare particolare attenzione alle caratteristiche su cui la predizione sarà effettuata. Se non accuratamente selezionate, le features utilizzate per stimare il valore dell'auto potrebbero portare a risultati non corretti o comunque molto distanti dalla realtà.

Tecnologie e tool utilizzati Per lo sviluppo del modello e le operazioni di acquisizione, analisi e preparazione dei dati verrà utilizzato il linguaggio **Python** e nello specifico le librerie **pandas** e **seaborn**. Per la creazione del modello verrà invece utilizzata la libreria **Scikit-learn**.

3.2.2 Data Understanding

La fase di **Data Understanding** si compone delle operazioni di identificazione, collezione e analisi dei Dataset che possono portare al raggiungimento degli obiettivi. Vengono infine identificati e discussi possibili problemi di qualità dei dati.

Identificazione e Collezione dei Dataset Il Dataset selezionato per la risoluzione del problema è stato recuperato da **Kaggle**, una delle piattaforme più utilizzate nei campi della Data Science e del Machine Learning. Il Dataset è reperibile a questo link: <https://www.kaggle.com/datasets/shaunolund/auto-sales-ebay-germany-random-50k-cleaned>

Analisi dei Dataset Il Dataset utilizzato contiene oltre 37.000 inserzioni di auto caricate su eBay Kleinanzeigen da utenti privati ed è rappresentato da un file in formato .csv (Comma-Separated Values) all'interno del quale ogni record è descritto da valori separati da virgola.

Il Dataset presenta 18 campi, i quali vengono riportati di seguito con una breve descrizione:

- **Id**: identificativo dell'inserzione;
- **date_crawled**: data di estrazione dell'inserzione.
- **car_name**: testo dell'inserzione.
- **price_EUR**: prezzo dell'auto in euro.
- **ab_test**: se l'inserzione è inclusa in un test A/B.
- **vehicle_type**: tipo di veicolo.
- **registration_year**: anno di immatricolazione del veicolo.
- **transmission**: tipo di trasmissione dell'auto.
- **power_ps**: potenza dell'auto in cavalli (CV).
- **model**: modello dell'auto.
- **odometer_km**: chilometraggio dell'auto.
- **registration_month**: mese di immatricolazione del veicolo.
- **fuel_type**: tipo di carburante dell'auto.
- **brand**: marca dell'auto.
- **unrepaired_damage**: presenza di danni non riparati all'auto.

- **add_created**: data di creazione dell'inserzione.
- **postal_code**: codice postale della locazione dell'auto.
- **last_seen**: data dell'ultima visita da parte del crawler.

Come si può notare, alcuni dei campi sopraelencati forniscono informazioni soltanto sull'annuncio dell'auto e non sull'auto stessa: non saranno pertanto presi in considerazione.

Per quanto riguarda i restanti attributi, grazie all'uso delle librerie Python **pandas** e **seaborn**, sono state effettuate operazioni di esplorazione dei dati al fine di visualizzarli e rilevare eventuali relazioni.

I risultati più interessanti dell'esplorazione sono di seguito riportati e discussi.

Codice 3.1: Codice utilizzato per l'esplorazione dei dati.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#Importiamo il file .csv contenente i dati
df = pd.read_csv("autos_random_50k_cleaned.csv")

#Diamo un nome alla colonna degli Id
df.rename(columns={"Unnamed: 0": "Id"}, inplace=True)

#Settiamo gli Id del DataFrame considerando la colonna degli Id del Dataset
df.set_index("Id", inplace=True)

#Ricaviamo dalla colonna dell'anno di immatricolazione la colonna degli anni dell'auto
df['registration_year'] = 2016 - df['registration_year']

#Rinominiamo la colonna dell'anno di immatricolazione
df.rename(columns={'registration_year': 'anni'}, inplace=True)

#Costruiamo gli scatter plot tra le variabili indipendenti potenza, anni e chilometraggio
#e la variabile dipendente price_EUR
sns.pairplot(df, x_vars='power_ps', y_vars='price_EUR', height=5, aspect=1.5,
              kind='reg', plot_kws={'line_kws':{'color':'red'}})
sns.pairplot(df, x_vars='anni', y_vars='price_EUR', height=5, aspect=1.5,
              kind='reg', plot_kws={'line_kws':{'color':'red'}})
sns.pairplot(df, x_vars='odometer_km', y_vars='price_EUR', height=5, aspect=1.5,
              kind='reg', plot_kws={'line_kws':{'color':'red'}})

#Calcoliamo la correlazione tra le colonne del Dataset
correlazione = df.corr(numeric_only=True)

#Creiamo l'heatmap
sns.heatmap(correlazione, annot=True, cmap='coolwarm')

#Mostriamo i grafici
plt.show()
```

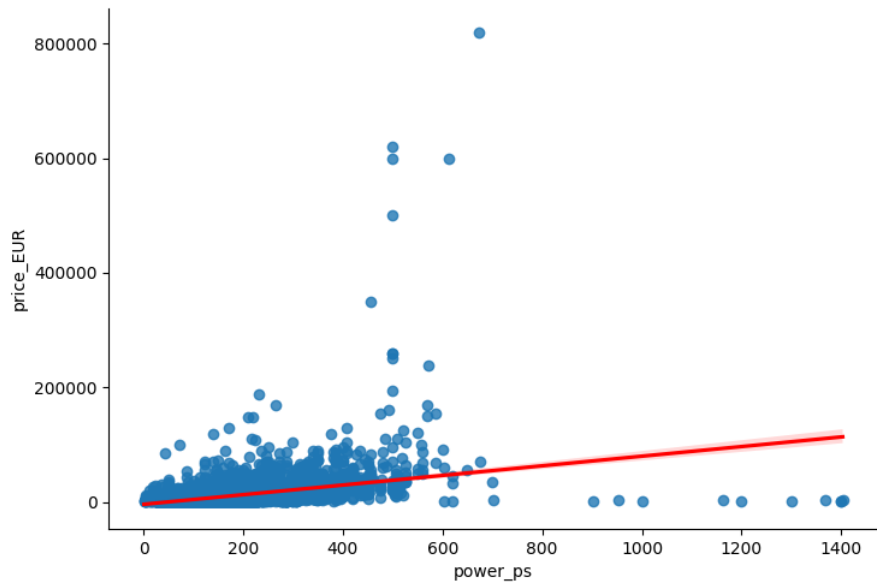


Figura 3.1: **Scatter Plot potenza**

Dal grafico a dispersione soprariportato si deduce che, com'era prevedibile, il prezzo di un'auto cresce all'aumentare della sua potenza. Ciò è verificabile facilmente dall'andamento della retta di regressione.

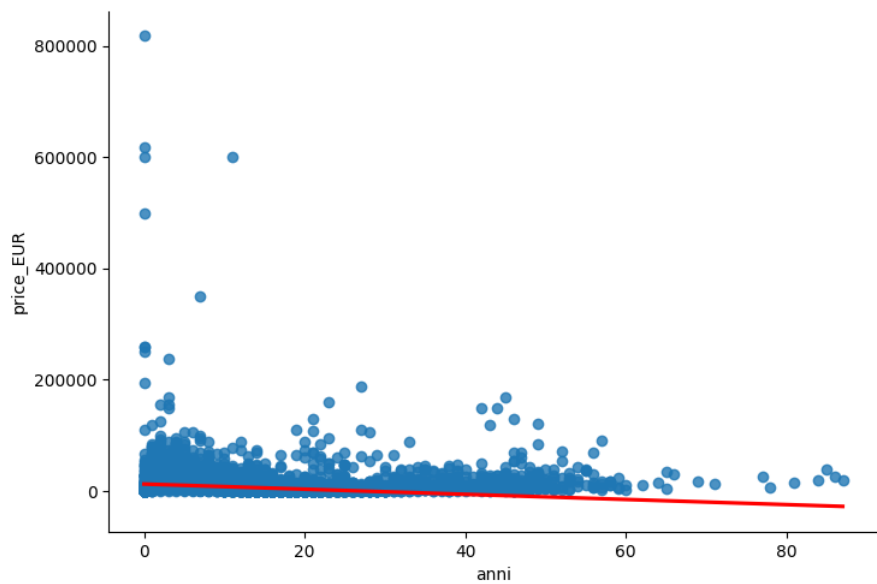


Figura 3.2: **Scatter Plot anni**

Per quanto riguarda gli anni, essi sono stati ricavati dalla colonna relativa all'anno di immatricolazione del veicolo: ciò è stato fatto per rendere più chiara e naturale l'interpretazione del grafico.

Contrariamente alla potenza, il prezzo dell'auto in questo caso diminuisce all'aumentare degli anni della stessa.

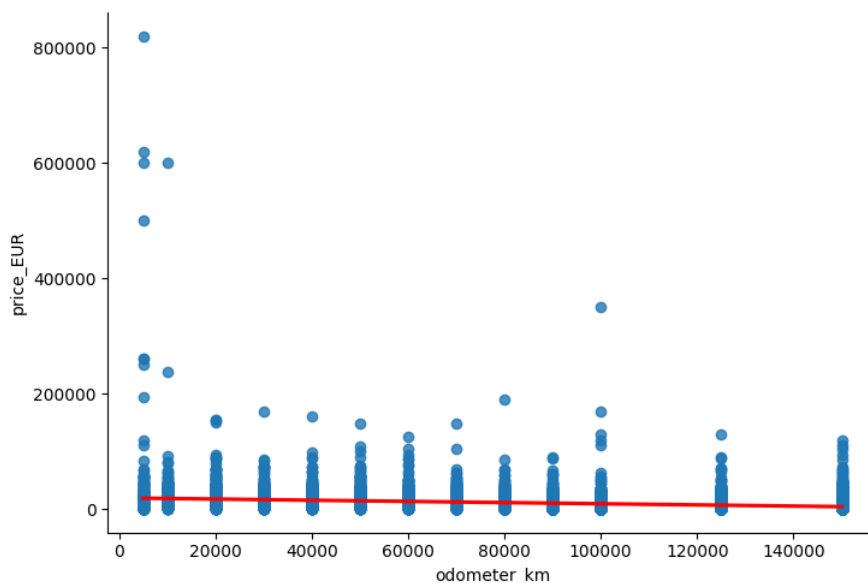


Figura 3.3: **Scatter Plot chilometraggio**

Come avveniva per gli anni, ad una quantità maggiore di chilometri percorsi corrisponde un prezzo minore. Dando un ulteriore sguardo al grafico, sembrerebbe che i valori impostati dai venditori per il chilometraggio delle proprie auto sia stato approssimato alla decina di migliaia più vicina al valore reale dei chilometri percorsi. Si potrebbe allo stesso modo approssimare il chilometraggio inserito dall'utente per effettuare la predizione.

Tra i pochi dati di tipo numerico presenti nel Dataset, questi appena descritti presentavano le relazioni più interessanti. Altri (come il mese di immatricolazione) non fornivano informazioni rilevanti rispetto alla predizione del prezzo. I dati di tipo qualitativo, invece, verranno trattati diversamente nella fase di Data Preparation, dato che non è possibile visualizzarli direttamente in un grafico a dispersione.

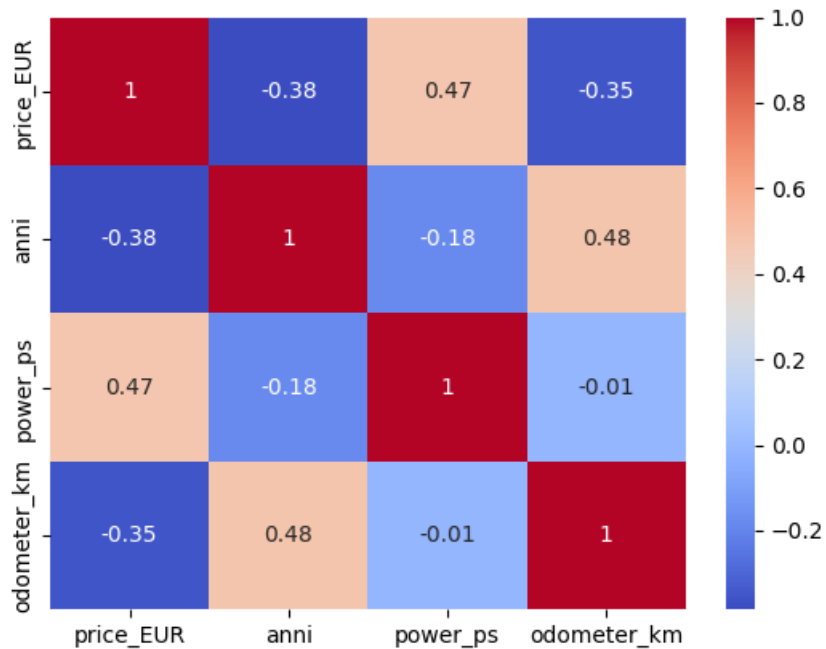


Figura 3.4: **HeatMap**

L'Heatmap (Mappa di calore) mostrata in figura contiene, per ognuna delle variabili indipendenti, il coefficiente di correlazione con le altre variabili indipendenti e il coefficiente di correlazione con la variabile dipendente. La mappa mette in evidenza diversi aspetti: il primo è che non ci sono problemi di multicollinearità per quanto riguarda le variabili utilizzate. Il secondo è che non ci sono predittori considerevolmente buoni. Ciò fa intuire che il problema della predizione del prezzo di un'auto è più complesso di quanto possa sembrare e che servirebbero più variabili per una stima più corretta.

Possibili problemi di qualità Il Dataset utilizzato non è purtroppo esente da difetti. Un esempio è il campo *power_ps* (potenza): alcune istanze presentano valori superiori a 1000 CV, il che fa dedurre che probabilmente, per queste istanze, è stata inserita la cilindrata al posto della potenza.

Un altro problema è rappresentato dal fatto che all'interno del Dataset sono presenti istanze di auto d'epoca le quali, facendo parte di un mercato completamente diverso, rischiano di influenzare negativamente la stima effettuata dal modello.

Infine alcune righe del Dataset contengono valori "Unknown" (sconosciuti) o "Andere/Sonstige" (altro) in corrispondenza di alcune colonne. Essi equivalgono a tutti gli effetti a delle istanze con campi nulli. Sarà quindi necessario utilizzare tecniche di Data Cleaning per risolvere il problema.

3.2.3 Data Preparation

L'obiettivo della fase di **Data Preparation** è quello di preparare i dati in maniera tale che possano essere utilizzati nei successivi passi del processo.

È in questa fase infatti che vengono selezionate le caratteristiche (**features**) del problema che hanno maggiore potenza predittiva, vengono puliti i dati sulla base dei problemi di qualità riscontrati nella fase precedente e vengono infine formattati i dati in maniera tale che possano essere utilizzati da un modello di Machine Learning.

Gli step della Data Preparation sono quindi essenzialmente 4: **Data Cleaning**, **Feature Scaling**, **Feature Selection**, **Data Balancing**.

Data Cleaning In questa fase vengono risolti i problemi di qualità riscontrati nella fase di esplorazione e analisi dei dati.

Nella fattispecie, il dataset presentava alcuni problemi riguardanti la colonna della potenza (*power_ps*) e la colonna degli *anni*.

Per quanto riguarda la potenza, sono state esaminate più accuratamente le istanze che presentavano un valore di *power_ps* maggiore di 1000 CV. Trattandosi di potenze ben superiori alla media, probabilmente è stato commesso qualche errore dal venditore in fase di inserimento delle informazioni del veicolo. L'intenzione era quindi quella di rimuovere queste righe dal Dataset.

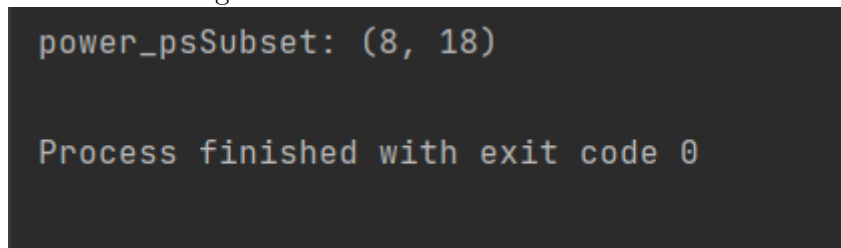
Tuttavia il codice utilizzato per l'analisi produceva questi risultati:

Codice 3.2: Codice utilizzato per analizzare le righe del Dataset con potenza maggiore di 1000 CV.

```
#Analizziamo le istanze che presentano un valore di power_ps maggiore di 1000 CV
power_psSubset = df[(df['power_ps'] >= 1000)]

#Visualizziamo il numero di righe del sottoinsieme creato
print("power_psSubset:", power_psSubset.shape)
```

Figura 3.5: Risultati del Codice 3.2



```
power_psSubset: (8, 18)

Process finished with exit code 0
```

I risultati soprariportati evidenziano che il Dataset contiene solo 8 auto con un valore di *power_ps* superiore a 1000 CV, poco più dello 0,02% dell'intero insieme di dati.

A tal proposito si è deciso di mantenerli all'interno del Dataset poiché la loro quantità rispetto al numero totale di righe (37866) fa sì che la loro influenza sul regressore sia quasi nulla e quindi trascurabile.

Per quanto concerne invece la colonna degli *anni*, il problema preso in considerazione riguardava la diversità tra il mercato delle auto d'epoca e il mercato delle auto usate ordinarie. Sono state per questo motivo escluse dal Dataset attraverso il codice di seguito riportato:

Codice 3.3: Codice utilizzato per escludere le auto d'epoca dal Dataset.

```
#Rimuoviamo le auto d'epoca
df = df[(df['anni'] >= 0) & (df['anni'] <= 30)]
```

Infine, come precedentemente anticipato, occorre risolvere il problema dei dati mancanti. A tal proposito possono essere utilizzate tecniche di **Data Imputation**. In questo caso si è deciso di rimuovere le righe del Dataset che presentavano i valori "Unknown", "Andere" e "Sonstige_autos" (altre auto) in uno o più campi. Per motivi di Feature Selection descritti nei paragrafi successivi, queste righe sono state rimosse soltanto dopo aver escluso alcune features non rilevanti. Sotto viene riportato il codice utilizzato:

Codice 3.4: Codice utilizzato per la Data Imputation.

```
#Riduciamo il Dataset alle sole Features selezionate e la variabile Target  
#I motivi sono descritti nel paragrafo relativo alla Feature Selection  
df = df[['price_EUR', 'vehicle_type', 'anni', 'transmission', 'power_ps', 'odometer_km', 'fuel_type', 'brand']]  
  
#Rimuoviamo le righe del Dataset con valori "Unknown" o "altro (andere/sonstige)" (sconosciuti)  
df = df[df != "Unknown"]  
df = df.dropna(axis=0)  
df = df[df != "andere"]  
df = df.dropna(axis=0)  
df = df[df != "sonstige_autos"]  
df = df.dropna(axis=0)
```

Feature Scaling Nella fase di Feature Scaling lo scopo è quello di normalizzare l'insieme di valori delle caratteristiche del Dataset, in maniera tale da conferire la stessa importanza ad ognuna delle features e quindi evitare che il modello sovrastimi o sottostimi l'importanza di una variabile (se questa ha una distribuzione superiore o inferiore rispetto a quella delle altre).

Esistono diverse tecniche di normalizzazione. Nel caso specifico, per garantire che tutte le caratteristiche abbiano lo stesso "peso", si è adottata la **Z-Score normalization**. Questa procedura consiste nel trasformare ogni valore in una unità di deviazione standard rispetto alla media della serie di dati.

Di seguito viene raffigurato il codice che ci ha permesso di effettuare l'operazione di normalizzazione sulle colonne numeriche del Dataset:

Codice 3.5: Codice utilizzato per la Z-Score normalization.

```
#Selezioniamo il sottoinsieme di features da normalizzare  
toNormalize = df[['power_ps', 'anni', 'odometer_km']]  
  
#Sovrascriviamo le colonne del Dataset originale  
df[['power_ps', 'anni', 'odometer_km']] = toNormalize.apply(zscore)
```

Le altre features di tipo numerico non sono state normalizzate a causa del loro significato: l'*Id* è soltanto un riferimento ad una precisa istanza del Dataset; il *postal_code* e *registration_month* sono informazioni superflue per il problema in esame.

Feature Selection In questa fase avviene il processo di definizione delle features che possono caratterizzare gli aspetti principali del problema e, quindi, avere una buona potenza predittiva. Il processo che definisce le metodologie per identificare le caratteristiche dai dati grezzi estraibili è detto **Feature Engineering**.

Una delle tecniche più utilizzate in questo contesto è la **Feature Selection**, che consiste nel selezionare le variabili più significative partendo da quelle a disposizione.

Rispetto al problema considerato, segue una descrizione dettagliata del processo di Feature Selection messo in atto.

- **Features scartate:** le features che sono state scartate sono le seguenti: *date_crawled*, *ad_created*, *postal_code*, *last_seen*, *car_name*, *registration_month*, *ab_test*, *unrepaired_damage* e *model*.

Le **features in rosso** sono state scartate perché fornivano dettagli relativi all'inserzione piuttosto che all'auto, ed aggiungevano quindi solo rumore.

Le **features in verde** invece, sono state scartate perché non utili ai fini della risoluzione del problema (il mese di immatricolazione è molto meno importante dell'anno e i dati per l'A/B Testing non aiutano in alcun modo il modello).

La feature relativa alla presenza di danni non riparati (*unrepaired_damage*), seppur apparentemente discriminante, non può essere utilizzata efficacemente. Questo perché il valore di un'auto, in caso di danni, è influenzato non solo dalla loro presenza ma anche dalla loro gravità.

Infine, il *model* è stato rimosso poiché caratterizzare il problema prendendo in considerazione anche i modelli delle auto risultava molto complesso. Inoltre il modello sarebbe stato incapace di stimare il valore di un'auto nei casi in cui, per il modello di quest'ultima, non ci fossero state corrispondenze nel Training Set.

- **Feature selezionate:**

Le features che sono state invece mantenute sono: *brand*, *vehicle_type*, *registration_year* (convertito in *anni*), *transmission*, *power_ps*, *odometer_km*, *fuel_type*

Trattandosi di un problema di regressione, bisogna gestire le features qualitative in maniera diversa. L'approccio utilizzato è il seguente:

Codice 3.6: Codice utilizzato per la gestione delle variabili qualitative.

```
#Creiamo un sottoinsieme di variabili qualitative
var_qualitative = df.select_dtypes(include=['object'])

#Convertiamole in variabili dummies
var_dummies = pd.get_dummies(var_qualitative, drop_first=True)

#Rimuoviamo dal dataset le colonne qualitative originali
df = df.drop(list(var_qualitative.columns), axis=1)

#Inseriamo le variabili qualitative all'interno del Dataset
df = pd.concat([df, var_dummies], axis=1)
```

Il procedimento utilizzato prevede la conversione delle variabili qualitative in variabili dummy ("manichino"). Attraverso queste ultime è possibile integrare in un modello di regressione anche variabili di tipo non numerico. La conversione si basa sul creare una nuova variabile binaria per ogni possibile valore della variabile qualitativa (ad eccezione di una, dato che il suo valore è derivabile da quello delle altre). Queste variabili binarie saranno poi incluse nella funzione di regressione. Come riportato nel Codice 3.6, sono state prodotte le variabili dummy per ciascuna delle features qualitative selezionate (*transmission*, *fuel_type*, *brand* e *vehicle_type*).

Data Balancing Per i problemi di regressione, prevedere operazioni di Data Balancing non è tanto importante quanto per i problemi di classificazione, dato che possibili problematiche relative a sbilanciamenti del Dataset per i valori della classe target sono assenti.

3.2.4 Data Modeling

Una volta sistemati i dati, si prosegue con la fase di **Data Modeling**, dove viene definito l'algoritmo di Machine Learning in relazione al problema in esame e ai dati a disposizione.

Una volta scelto il modello da realizzare, esso viene addestrato. Ciò viene effettuato dividendo il Dataset di partenza in maniera tale da considerare alcune delle istanze come non note.

Il Training Set conterrà le istanze che l'algoritmo utilizzerà per allenarsi; il Test Set, invece, raggrupperà quelle istanze per cui l'algoritmo allenato dovrà predire il valore.

Tale suddivisione è dettata dal fatto che addestrare e validare un modello di machine learning sullo stesso Dataset porta ad avere risultati totalmente inaffidabili.

Data la natura del problema in esame, come specificato nei capitoli precedenti, si è deciso di sviluppare un modello di regressione lineare.

Esistono diversi modi per dividere Training e Test set. La tecnica scelta in questo contesto è stata la **10-fold cross validation**, tecnica che consiste nella ripetuta partizione e valutazione dell'insieme dei dati di partenza.

Nella pagina successiva viene mostrato il codice utilizzato per la realizzazione del modello di regressione lineare:

Codice 3.7: Codice utilizzato per la realizzazione del modello di regressione lineare.

```
#Selezioniamo il sottoinsieme di features che il modello utilizzerà per le sue predizioni
x = df.drop(columns="price_EUR", axis=1)

#Isoliamo la variabile dipendente
y = df["price_EUR"]

#Creiamo il modello di regressione lineare
reg = linear_model.LinearRegression(fit_intercept=True)

#Inizializziamo la 10-fold cross validation
ten_fold = model_selection.KFold(n_splits=10, shuffle=True, random_state=0)

#Creiamo un Array in cui memorizzeremo i MAE ad ogni iterazione
array_MAE = []

#Eseguiamo la convalida incrociata
for train_index, test_index in ten_fold.split(x):
    x_train, x_test = x.iloc[train_index], x.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    #Addestriamo il modello con la partizione di training corrente
    reg.fit(x_train, y_train)

    #Effettuiamo la predizione
    y_pred = reg.predict(x_test)

    #Calcoliamo il MAE per questa iterazione
    MAE = metrics.mean_absolute_error(y_test, y_pred)
    array_MAE.append(MAE)

#Calcoliamo la media dei MAE ottenuti in ogni iterazione
mean_MAE = sum(array_MAE) / len(array_MAE)

#Arrotondiamo alle prime 2 cifre decimali
mean_MAE = round(mean_MAE, 2)

print("Media_MAE:", mean_MAE)
```

Come si può vedere sono state innanzitutto selezionate le features che il modello avrebbe dovuto utilizzare (*brand*, *vehicle_type*, *anni*, *transmission*, *power_ps*, *odometer_km* e *fuel_type*) e la variabile target da predire (*price_EUR*), ed è stato creato il modello di regressione lineare. Il regressore è stato poi addestrato e valutato seguendo la tecnica della convalida incrociata. Nella sezione successiva verranno forniti ulteriori dettagli riguardo le metriche di valutazione utilizzate per valutare le prestazioni del modello creato.

3.2.5 Evaluation

La fase di **Evaluation** ha l'obiettivo di valutare se i risultati sono chiari, se sono in linea con gli obiettivi di business e se rivelano delle prospettive aggiuntive alle quali il progettista non aveva pensato. Viene inoltre valutato l'intero processo di sviluppo, interrogandosi su eventuali aspetti non convincenti, l'uso di metodologie alternative e il loro impatto sui risultati.

In questa fase verranno discussi i risultati ottenuti dal modello di regressione lineare sviluppato per AutoMate.

In primo luogo, nel contesto dei problemi di regressione, esistono diverse metriche con le quali è possibile effettuare un'analisi dei risultati ottenuti:

$$\text{MAE (Mean Absolute Error): } \frac{\sum_{i=1}^n |\tilde{y} - y|}{n}$$

La metrica MAE indica la differenza media osservata tra i valori predetti e i valori reali del test set

$$\text{MSE (Mean Squared Error): } \frac{\sum_{i=1}^n (\tilde{y} - y)^2}{n}$$

La metrica MSE indica l'errore quadratico medio commesso sui dati presenti nel test set.

$$\text{RMSE (Root Mean Squared Error): } \sqrt{\frac{\sum_{i=1}^n (\tilde{y} - y)^2}{n}}$$

La metrica RMSE indica la radice quadrata dell'errore quadratico medio commesso sui dati presenti nel test set.

La metrica utilizzata per valutare le prestazioni del modello creato è la MAE, poiché la sua interpretazione è molto semplice. In riferimento al Codice 3.7, l'istruzione relativa alla stampa produce questi risultati:

```
Media MAE: 2856.08

Process finished with exit code 0
```

Figura 3.6: Risultati del Codice 3.7

Possiamo interpretare il risultato ottenuto come: il prezzo di un'auto predetto dal modello di regressione lineare è, in media, distante di circa **2850 euro** dal prezzo reale.

Come si può notare, l'esito non è molto soddisfacente: il modello creato è poco preciso. Si è deciso quindi di provare a rimodellare e rivalutare il regressore considerando anche la variabile *unrepaired_damage*, con l'intenzione di fornire informazioni (seppur, come specificato nella sezione relativa alla Feature Selection, non molto significative) sulla presenza di eventuali danni. Il codice utilizzato è il seguente:

Codice 3.8: Rivalutazione del modello utilizzando una variabile aggiuntiva: *unrepaired_damage*.

```
#Riduciamo il Dataset alle sole Features selezionate (e "unrepaired_damage")
e la variabile Target
df = df[['price_EUR', 'vehicle_type', 'anni', 'transmission', 'power_ps',
        'odometer_km', 'fuel_type', 'brand', 'unrepaired_damage']]

...
... #Creiamo il modello di regressione come
... #fatto precedentemente
...

#Stampiamo la media dei MAE del nuovo modello
print("Media_MAE:", mean_MAE)
```

Il modello, con questa configurazione, ha una MAE di circa 2910, di poco superiore al valore della metrica del modello precedente. Ciò fa intuire che probabilmente le prestazioni sono negativamente influenzate da fattori esterni alla fase di modellazione. Come si spiega allora il comportamento del regressore?

Una prima possibile motivazione è che il problema della predizione del prezzo di un'auto è tutt'altro che semplice: le colonne presenti nel Dataset utilizzato fanno riferimento ad un piccolo sottoinsieme delle caratteristiche tecniche di un'auto. Basare la predizione su un insieme così limitato porterà inevitabilmente il modello a commettere errori più penalizzanti.

Un'altra possibile causa del comportamento del modello è che il prezzo di un'auto è influenzato non solo dalle sue caratteristiche tecniche e dalle sue condizioni, ma anche da fattori difficilmente prevedibili legati al commercio online, come ad esempio la competenza del venditore, le conoscenze dell'acquirente sul mercato automobilistico, il luogo in cui viene scambiata l'auto venduta, e altri elementi che fanno oscillare il prezzo di vendita di un'auto e di conseguenza, se non considerati, causano ulteriori imprecisioni nelle predizioni del modello.

3.2.6 Deployment

Una volta giunti alla fase di **Deployment**, si procede mettendo in funzione l'approccio definito rendendo usabile il modello generato. In altre parole, questa fase vede il passaggio dall'ingegneria del machine learning all'ingegneria del software e all'ingegneria dell'usabilità.

Il modello costruito è reso disponibile attraverso un'App Desktop. L'interfaccia dell'applicazione presenta un form nel quale un utente può inserire le informazioni di un'auto della quale vuole stimare il prezzo. Compilati tutti i campi, l'utente può visualizzare il risultato cliccando su bottone che avvia il processo di predizione del prezzo sulla base dei valori forniti.

A questo link, nella sezione "Guida all'esecuzione", è possibile consultare una guida all'utilizzo dell'applicazione: <https://github.com/ZTunic/AutoMate/blob/main/README.md#guida-allesecuzione>.

Capitolo 4

Conclusioni

4.1 Considerazioni finali

Alla fine dei conti, come si è visto, AutoMate è a tutti gli effetti un agente intelligente che riesce a stimare, seppur con la presenza di un margine di errore, il prezzo di un'auto. La realizzazione di un modello preciso avrebbe dovuto prevedere l'uso di un Dataset più completo, che comprendesse tutte le informazioni mancanti citate nei capitoli precedenti. Tuttavia, la realizzazione di questo progetto puntava non a realizzare un modello di Machine Learning che fosse quanto più esatto possibile, quanto più a **"sporcarsi le mani"** e a comprendere meglio cosa significasse sviluppare un agente capace di apprendere, seguendo in dettaglio tutti i passi di un modello di Ingegneria del Machine Learning.