POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Gestionale

Tesi di Laurea Magistrale

Progettazione e sviluppo di una metodologia data driven nel contesto automobilistico.

Caso di studio: Weight Management

Relatore

Prof.ssa Tania Cerquitelli

Candidato

Valentino Bolognin

Marzo 2020

INDICE

Introduzione 1

CAPITOLO 1. Azienda 3

CAPITOLO 2. Stato dell'arte 6

2.1 Selezione, pre-processing e trasformazione 8

2.1.1 Outlier detection 10

2.2 Data mining 13

2.2.1 Modelli di classificazione 14

2.2.2 Modelli di regressione 17

2.3 Interpretazione e validazione 20

CAPITOLO 3. Caso di studio 24

3.1 Weight Management 26

3.2 Struttura dei dati 28

3.2.1 Struttura dei dataset riguardanti le Pesate 28

3.2.2 Struttura dei dataset riguardanti gli Optional 29

3.3 Strumenti utilizzati 30

CAPITOLO 4. Metodologia proposta 33

4.1 Raccolta dati 34

4.1.1 Raccolta Pesate 34

4.1.2 Raccolta Optional 36

4.2 Pulizia dati 36

4.2.1 Pulizia Pesate 37

4.2.2 Pulizia Optional 40

4.3 Trasformazione dati 40

4.3.1 Trasformazione Pesate 40
4.3.2 Trasformazione Optional 42
4.3.3 Join tra Pesate e Optional 44
4.4 Creazione modello 44
CAPITOLO 5. Risultati sperimentali 47
5.1 Raccolta dati 47
5.2 Pulizia dati 50
5.2.1 Boxplot 52
5.2.2 EllipticEnvelope 53
5.3 Trasformazione dati 55
5.4 Creazione modello 56
5.4.1 Regressione lasso polinomiale 56
5.4.2 Random forest 58
5.5 Valutazione optional 60
Conclusioni e sviluppi futuri 62

Introduzione

Il lavoro di tesi è stato svolto presso l'azienda AMET S.r.l., che ha messo a disposizione i dati oggetto delle analisi presentate.

Il tema del progetto di tesi è legato all'esigenza aziendale di applicare metodologie e tecniche di *data analytics* per la soluzione di problematiche concrete dei clienti di AMET nel settore automobilistico. L'ambito operativo cui si riferiscono le esigenze di analisi dati è principalmente quello della gestione pesi (*Weight Management*) di componenti e sistemi del veicolo, che vanno a comporre il peso totale del veicolo stesso.

L'attività ha l'obiettivo di mettere a punto una metodologia di analisi dei dati relativi alle pesature effettuate su differenti versioni di autoveicoli, che consenta di stabilire una correlazione efficace tra il peso del veicolo e il set di optional disponibili su ogni modello.

Il presente lavoro di tesi è articolato in 5 capitoli, di cui viene di seguito presentato un sommario.

Nel **Capitolo 1** viene descritta l'azienda e l'organizzazione delle attività all'interno delle diverse *business lines*.

Nel **Capitolo 2** viene fornita un'introduzione generale al processo di estrazione della conoscenza a partire da dati eterogenei con particolare attenzione alle tecniche di *data analytics* utilizzate in questa tesi.

Nel **Capitolo 3** viene introdotto il caso di studio e i gli strumenti utilizzati per analizzare i dati ricevuti.

Nel **Capitolo 4** viene descritta la metodologia proposta con l'obiettivo finale di proporre un modello utilizzabile per i dati futuri.

Nel **Capitolo 5** vengono presentati i risultati sperimentali ottenuti applicando le tecniche di *data mining* sui dati disponibili.

CAPITOLO 1

Azienda

AMET **[1]** (acronimo di *Applied Mechatronic Engineering and Technologies*), impresa nata nel 1999 da uno dei primi spin-off del Politecnico di Torino tramite l'esperienza dell'incubatore I3P, è una società di ingegneria attiva nella progettazione e sperimentazione di prodotti meccanici e meccatronici, con l'impiego della simulazione numerica e delle metodologie *model-based*. AMET opera principalmente nel settore dei trasporti (automotive, ferroviario e aerospaziale).

AMET collabora con continuità con il Politecnico di Torino ed è inoltre particolarmente attiva all'interno dei programmi di ricerca finanziati, sia in ambito regionale e nazionale, sia in ambito europeo, con la partecipazione a diversi progetti, sia come partner tecnico, sia come supporto al *Project Management*.

Le attività di AMET si concentrano sulle soluzioni metodologiche e tecnologiche finalizzate a servizi di ingegneria per la progettazione e lo sviluppo di prodotti industriali, con un approccio integrato multi-dominio, basato su modelli, in ottica di ottimizzazione del processo e del prodotto. Fra i punti di forza di AMET si segnalano la capacità di integrare le competenze relative ai diversi domini che includono la meccanica, l'elettronica, i controlli automatici, l'analisi dei dati e lo sviluppo SW, grazie alla flessibilità nell'adattare il metodo di lavoro ai diversi contesti.

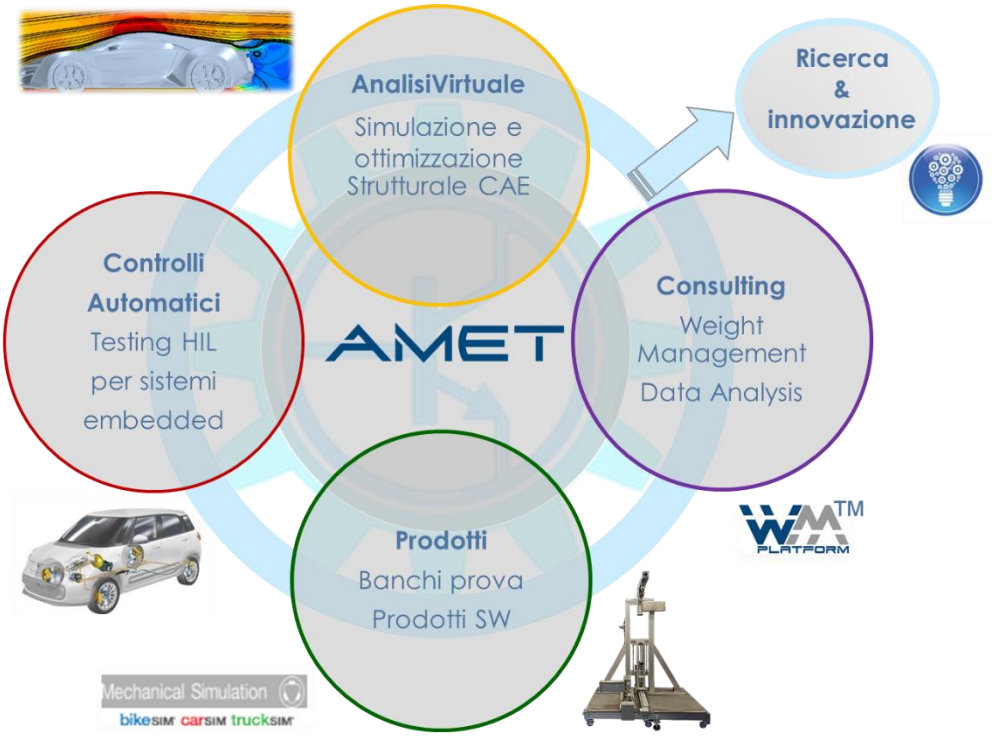

Figura 1.1: Organizzazione delle *business lines* di AMET

Le attività sono organizzate nelle seguenti *business lines* schematizzate in Figura 1.1:

- **Analisi e testing virtuale**: analisi e ottimizzazione strutturale, in particolare sicurezza passiva, biomeccanica, rumore e vibrazioni (NVH), fatica, dinamica veicolo, fluido-dinamica computazionale (CFD), utilizzando strumenti e metodi per la modellizzazione ad elementi finiti (FEM) e *multibody* (MBS).
- **Controlli automatici**: ingegneria dei sistemi di controllo automatico e delle reti di comunicazione, basata su modellazione dinamica.
- **Prodotti**: banchi prova per sperimentazione fisica a fatica di componenti (in particolare parti mobili di veicoli quali porte, cofano, portellone, baule) e banchi prova per validazione Hardware In the Loop (HIL) di controllori elettronici *embedded*. Inoltre, AMET è distributore di prodotti SW di terze parti.

- ***Consulting***: consulenza sullo sviluppo prodotto, analisi e ottimizzazione prestazioni, in particolare *data analytics* in applicazione alla gestione pesi (*Weight Management*) descritta in maggiore dettaglio nei capitoli successivi.

CAPITOLO 2

Stato dell'arte

La nascita delle metodologie *data analytics* risponde alla necessità di saper analizzare e ottenere conoscenze utili da grandi quantità di dati derivanti dai molteplici contesti operativi presenti all'interno della nostra società. Il termine *data analytics* rappresenta l'insieme di tecniche e metodologie che hanno per obiettivo l'estrazione di informazioni, e quindi di conoscenza, da grandi quantità di dati. Molte delle informazioni potenzialmente estraibili dai dati non sono direttamente evidenti e le analisi guidate da un analista potrebbero richiedere molto tempo per ottenere informazioni utili.

Prima ancora di descrivere nello specifico le tecniche di *data analytics*, è doveroso introdurre i fattori che hanno contribuito al loro sviluppo negli ultimi anni. Il fattore principale risiede nella grande velocità con cui grandi moli di dati vengono prodotti e memorizzati con l'ausilio di supporti informatici. Questi dati sono denominati "*big data*" proprio perché prodotti in volumi tali da rendere necessario l'uso di metodi innovativi e specifici per estrarne informazione. Inoltre, il termine "*big data*" si riferisce non solo alla numerosità dei dati, ma anche alla varietà che molto spesso li caratterizza, intesa come eterogeneità dei dati e delle loro sorgenti che, pertanto, necessitano di essere ben identificate e normalizzate.

Al giorno d'oggi le aziende raccolgono dati da sensori, da telefoni, da informazioni GPS e addirittura dai social network. Pertanto, l'informazione è diventata un bene prezioso che viene utilizzato per

superare la concorrenza: grandi colossi mondiali come Google, Facebook e Amazon sono riusciti ad anticipare i cambiamenti del mercato e a capire le preferenze del singolo cliente attraverso l'utilizzo di queste tecniche. Nonostante i progressi, sono molte le aziende che pur producendo e memorizzando grandissime quantità di dati, non utilizzano ancora tecniche di *data analytics* per estrarre conoscenza utile dai dati.

Il processo di estrazione della conoscenza è noto come KDD **[2]** (acronimo di *Knowledge Discovery from Data*) ed è stato coniato nel 1989 per descrivere la conoscenza come prodotto finale di un processo di ricerca *data driven* che, a partire dai dati grezzi, porta ad un'informazione spendibile per il *decision making*. È un processo iterativo e interattivo che si divide nelle seguenti fasi riportate in Figura 2.1:

- **Selezione dei dati**: consiste nell'individuare i dati rilevanti rispetto agli obiettivi dell'analisi.
- ***Pre-processing***: è una fase in cui si cerca di limitare l'effetto dei dati rumorosi, individuando una strategia per identificare e rimuovere gli *outlier* e le informazioni inutili.
- **Trasformazione dei dati**: in questa fase si eseguono delle apposite correzioni sul dataset, al fine di ottenere una struttura utile rispetto all'obiettivo dell'analisi.
- ***Data mining***: consiste nella scelta e nell'applicazione dell'algoritmo più appropriato. Per prima cosa è necessario decidere quale tipo di analisi eseguire, dopodiché si passa alla scelta dell'algoritmo più opportuno per estrarre una conoscenza utile.
- **Interpretazione e validazione della conoscenza**: l'ultima fase è basata sull'interpretazione e validazione dei dati ottenuti, così da consolidare la conoscenza estratta.

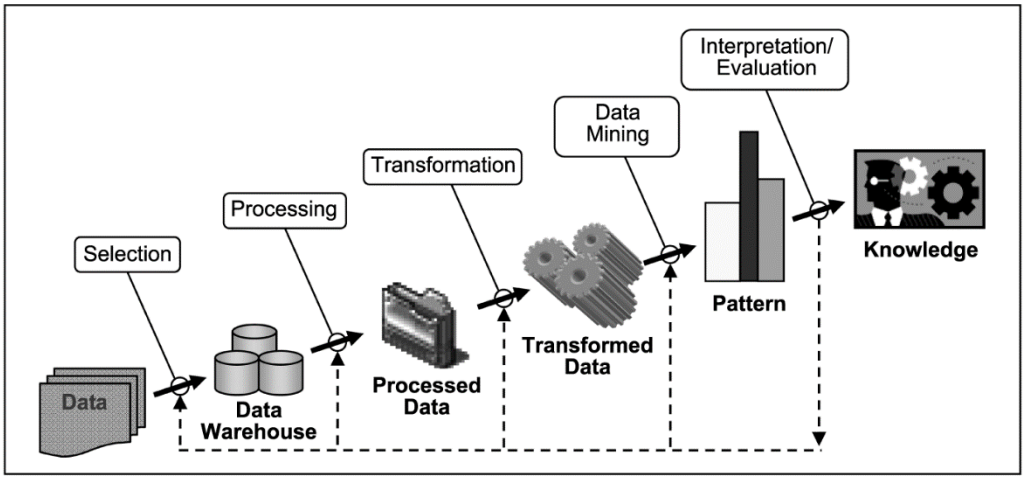

Figura 2.1: Processo di *Knowledge Discovery from Data*

Il KDD viene solitamente ripetuto varie volte modificando gli algoritmi scelti, finché non si giunge ad estrarre un tipo di conoscenza utile e che soddisfi l'obiettivo iniziale.

2.1 Selezione, pre-processing e trasformazione

La fase di preparazione dei dati è fondamentale per evitare di formulare un modello privo di significato. Questo perché nella realtà, raramente i dati sono utilizzabili senza elaborazioni. I principali problemi che si riscontrano nei dati sperimentali sono dovuti a valori mancanti, ridondanze, inconsistenze e presenza di errori e *outlier*. Ciò può dipendere da vari motivi come l'errato inserimento del dato nel sistema, un errore di rilevazione o una non corretta codifica del dato in fase di caricamento.

A tal scopo, innanzitutto occorrono una comprensione del dominio applicativo e una definizione degli obiettivi dell'analisi. In seguito, si procede con la selezione dei dati, focalizzandosi sulle variabili fondamentali per l'estrazione della conoscenza. Su questo sottoinsieme di dati scelti, si esegue la fase di *pre-processing*.

In questa fase, un'attività molto importante è la gestione dei dati mancanti. Gli approcci proposti per gestire i valori mancanti possono variare in base al problema in questione: si consiglia di ignorare i valori mancanti se non sono un numero considerevole, altrimenti si cerca di prevedere il dato mancante attraverso una stima basata sui dati noti.

Inoltre, il dataset va pulito da elementi duplicati o inconsistenti perché nella maggior parte dei casi non tutti gli attributi contenuti all'interno di un dataset risultano rilevanti ai fini dell'analisi. Per velocizzare le procedure di *data mining* può risultare utile generare nuovi attributi in modo da rappresentare meglio le informazioni di maggior interesse, utilizzando aggregazioni o trasformazioni su nuovi spazi.

Nel *pre-processing* l'attività principale è la pulizia dei dati che si esegue per attenuare i dati rumorosi, dovuti a errori di immissione o alla presenza di *outlier*. Per quanto riguarda gli errori di immissione si può effettuare un'ispezione manuale, con gli esperti del dominio applicativo, in modo da applicare alcuni vincoli per trovare e rimuovere questi errori. Invece, per l'*outlier detection* **[3]** esistono diversi metodi a seconda che si voglia eseguire un'analisi univariata o multivariata. Un'analisi univariata consiste nel trovare i valori anomali considerando una sola variabile alla volta, mentre si parla di analisi multivariata quando per trovare i valori anomali si considerano due o più variabili alla volta. Per l'analisi univariata, gli outlier possono essere identificati attraverso metodi parametrici che esaminano i principali indici statistici (minimo, massimo, mediana, media e deviazione standard) oppure attraverso metodi grafici (distribuzione gaussiana, istogrammi e boxplot). Per l'analisi multivariata, invece, si utilizzano tecniche che si basano su distanze (EllipticEnvelope) o su metodi grafici (DBSCAN).

2.1.1 Outlier detection

In statistica, un *outlier* è un valore estremo di una distribuzione che si caratterizza per assumere dei valori troppo alti o troppo bassi rispetto al resto della distribuzione e che identifica quindi un caso isolato. La definizione di *outlier* non è unica ed è influenzata dalla conoscenza personale del fenomeno. Lo stesso dato, dunque, può essere classificato in maniera diversa da metodi e persone diverse. In ogni caso è di fondamentale importanza utilizzare metodologie specifiche per identificali ed escluderli dalle analisi, dato che potrebbero impattare negativamente sulla qualità finale dei risultati.

In letteratura esistono differenti metodi per identificare gli *outlier* all'interno di un dataset. Nel nostro caso, come supporto decisionale all'esperto di dominio, sono state utilizzate sia tecniche univariate come il boxplot sia tecniche multivariate come l'EllipticEnvelope.

Il **boxplot [4]** è un metodo grafico che permette di visualizzare la distribuzione di un campione di dati attraverso i suoi quartili. In particolare, sono indicati la posizione del primo (Q1) e del terzo quartile (Q3), corrispondenti rispettivamente al segmento inferiore e al segmento superiore del rettangolo mostrato in Figura 2.2. La differenza tra il terzo e il primo quartile viene definito range interquartile (IQR) ed è una misura di dispersione simile alla deviazione standard o alla varianza, ma è molto più robusta rispetto agli *outlier*. Il boxplot riporta anche altri importanti valori come il valore minimo e il valore massimo definiti rispettivamente come $\mathrm{Q1} - 1.5 \cdot \mathrm{IQR}$ e $\mathrm{Q3} + 1.5 \cdot \mathrm{IQR}$. Tutti quei valori che si trovano al di sotto del valore minimo o al di sopra del valore massimo sono considerati *outlier*. Rispetto ad altri metodi grafici come l'istogramma, il boxplot può essere utile per valutare rapidamente

determinati macro comportamenti di un insieme di dati. In genere fornisce meno informazioni rispetto alla distribuzione cumulata del campione, ma diventa molto utile quando si vogliono confrontare sottogruppi di dati.

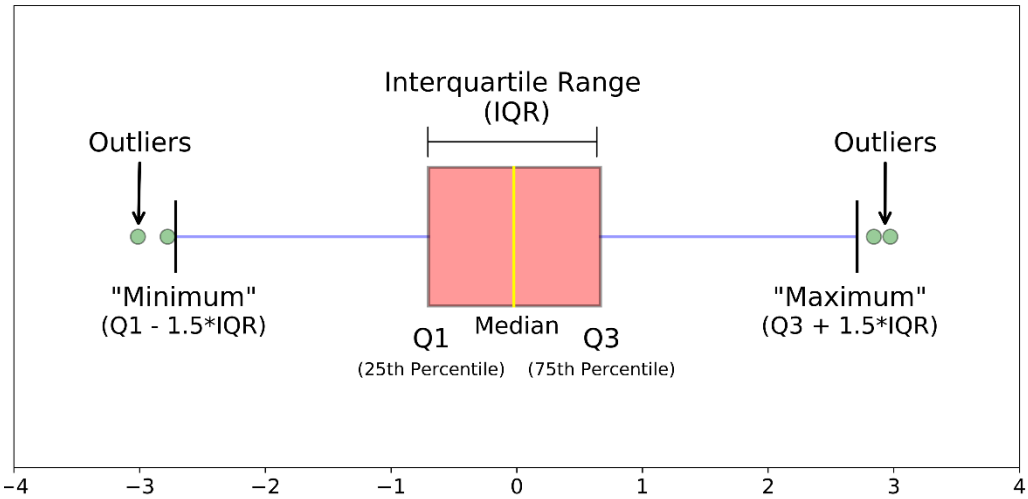

Figura 2.2: Boxplot

L'**EllipticEnvelope [5]** è un metodo basato sulle distanze che viene utilizzato per identificare valori anomali dai dati che seguono approssimativamente una distribuzione normale. Affinché questo metodo dia buoni risultati, è necessario confermare che la precedente ipotesi venga rispettata. Da questa ipotesi, generalmente si cerca di definire la "forma" dei dati e tutti quei valori che non rispettano la forma definita vengono considerati *outlier*. Questa tecnica stima la posizione e la covarianza degli *inlier* in modo robusto (cioè senza essere influenzata dagli *outlier*) e adatta un'ellisse ai punti centrali, ignorando tutti i punti che si trovano al di fuori. La distanza di un'osservazione x_i dalla forma definita può essere calcolata usando la distanza di Mahalanobis:

$$d_{(\mu,\Sigma)}(x_i)^2 = (x_i - \mu)'\Sigma^{-1}(x_i - \mu)$$

Dove μ e Σ rappresentano le stime della posizione e della covarianza della distribuzione normale. La consueta stima della covarianza

(Maximum Likelihood) è molto sensibile alla presenza di *outlier* nel dataset e, di conseguenza, lo saranno anche le corrispondenti distanze di Mahalanobis. Quindi, sarà meglio utilizzare uno stimatore robusto della covarianza (Minimum Covariance Determinant) per garantire che la stima sia resistente alle osservazioni "errate" nel dataset e che le rispettive distanze di Mahalanobis riflettano accuratamente le osservazioni. L'esempio in Figura 2.3 illustra come le distanze di Mahalanobis siano influenzate dalla presenza di *outlier*: le osservazioni tratte da una distribuzione contaminata non sono distinguibili da quelle provenienti dalla reale distribuzione normale con cui si vorrebbe lavorare. Utilizzando le distanze di Mahalanobis basate sul Minimum Covariance Determinant, le due popolazioni diventano distinguibili.

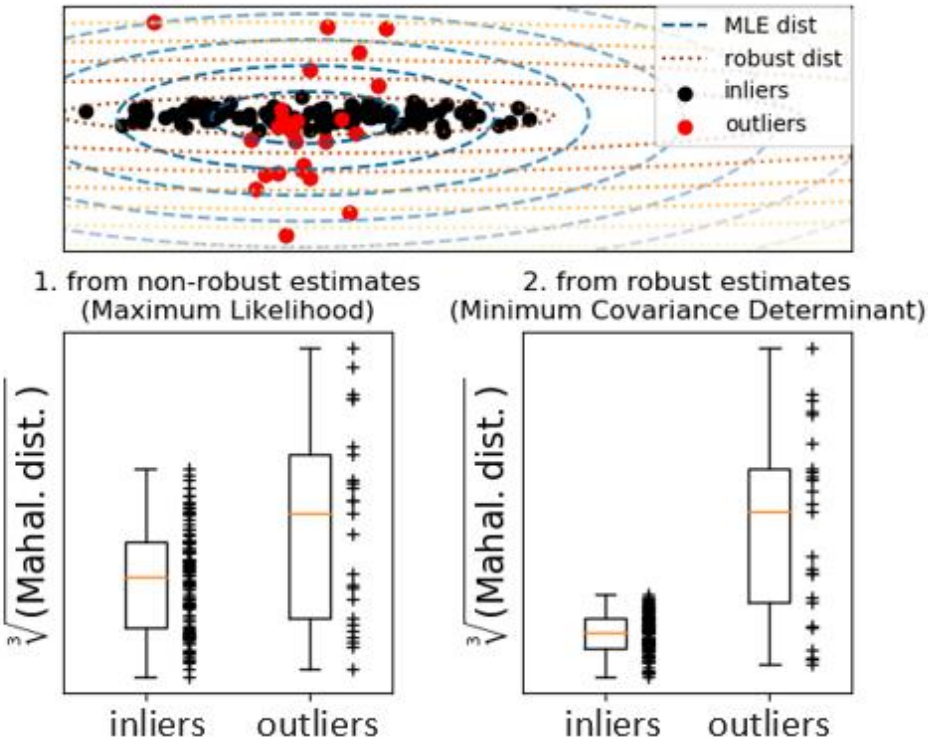

Figura 2.3: EllipticEnvelope

2.2 Data mining

Dopo la fase precedente che permesso di ottenere un dataset pulito, si procede con la fase di *data mining* **[6]**, nella quale si utilizzano tecniche fondate su specifici algoritmi che risultano utili se vengono applicati principalmente su grandi set di dati.

È importante osservare che non esiste una tecnica superiore alle altre, ma ogni tecnica è riferita a determinati obiettivi e tipologie di dati da analizzare. Spesso i migliori risultati per trasformare i dati in informazioni si ottengono attraverso la combinazione di diverse tecniche di *data mining*.

Le tecniche di *data mining* **[7]** possono avere obiettivi diversi e si dividono in base al grado di conoscenza del dataset da parte dell'utente. Esse possono essere suddivise in due categorie come mostrato in Figura 2.3: tecniche non supervisionate e tecniche supervisionate.

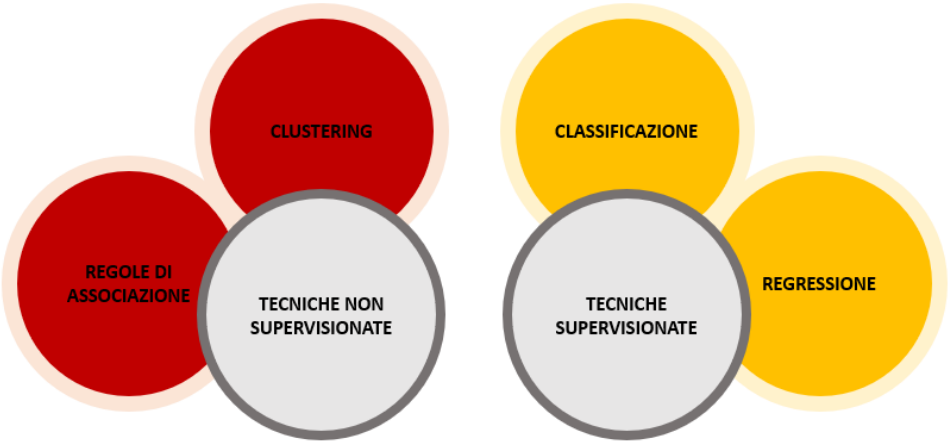

Figura 2.4: Tecniche di *data mining*

Le **tecniche non supervisionate** si utilizzano per eseguire un'analisi esplorativa dei dati e non richiedono nessun tipo di conoscenza a priori del dataset da parte dell'utente. Gli output forniti non hanno né una struttura definita né output associati. L'obiettivo dell'analisi esplorativa è

quello di ricercare e analizzare i pattern rilevanti che permettono di descrivere i dati. Le tecniche di apprendimento non supervisionato si basano fondamentalmente sul *clustering* o sulle regole di associazione.

Le **tecniche supervisionate** si utilizzano per creare modelli predittivi e prevedono la conoscenza a priori dei dati da parte dell'utente. I modelli predittivi si basano sull'uso di variabili o attributi noti con lo scopo di riuscire a predire il valore incognito o futuro di altre variabili di interesse. In particolare, la creazione di un modello predittivo consiste nell'esprimere una variabile di interesse y in funzione di diversi attributi noti $(X_1, \dots, X_n)$ chiamati anche predittori:

$$y = f(X_1, \dots, X_n) + \varepsilon$$

Dove f è una funzione incognita tramite la quale i predittori spiegano la variabile di interesse y, mentre ε indica un termine di errore indipendente dalle variabili di input. Questo termine è necessario perché un modello è sempre una semplificazione della realtà, perciò il vero il legame fra i predittori e la variabile d'interesse può venire al più approssimato. All'interno delle tecniche supervisionate, abbiamo una dicotomia di modelli utilizzabili che possono essere i modelli di classificazione e i modelli di regressione.

2.2.1 Modelli di classificazione

I **modelli di classificazione** si utilizzano quando l'output che si vuole ottenere è una variabile di tipo qualitativo o categorico, alla quale si assegna un'etichetta. Come input viene fornito un set di dati già etichettati in classi di appartenenza grazie ai quali viene creato un modello di classificazione, che in seguito verrà utilizzato su nuovi dati non etichettati in modo da assegnarli alla rispettiva classe.

Un esempio di applicazione delle tecniche di classificazione è mostrato in Tabella 2.1 e si ha quando le banche valutano se fornire un mutuo ai propri clienti analizzando le caratteristiche personali. In base ai dati storici, infatti, è possibile costruire un modello che associ a ogni persona una variabile etichetta che indica la possibilità di una frode oppure un'alta probabilità di restituire il debito contratto, mentre tutte le altre variabili sono utilizzate per costruire una logica di classificazione. Questa logica di classificazione sarà quindi utilizzata per classificare nuovi clienti non etichettati. A seconda del tipo di output, la classificazione sarà binaria se include solo due classi, oppure multi-classe se è possibile avere più di due output.

STATO CIVILE	REDDITO ANNUALE	ANNI RESIDUI	PAGA
Sposato	30K	3	SI
Sposato	10K	8	NO
Single	23K	4	SI
Single	25K	12	SI
Single	15K	1	SI
Sposato	8K	1	SI
Single	55K	11	SI
Sposato	17K	4	NO
Single	16K	3	NO
Sposato	22K	4	SI
Sposato	19K	4	NO
Sposato	7K	1	SI
Single	20K	5	SI
Single	30K	4	SI
Sposato	9K	3	NO
Single	21K	4	NO

STATO CIVILE	REDDITO ANNUALE	ANNI RESIDUI	PAGA
Single	10K	12	?
Sposato	22K	5	?
Single	23K	2	?

Tabella 2.1: Esempio di applicazione delle tecniche di classificazione

Esistono diverse tecniche di classificazione che si possono utilizzare su una serie di dati, tuttavia in questa sezione sono descritte le tecniche utilizzate all'interno di questo lavoro di tesi. Alcuni esempi sono gli alberi decisionali e il *random forest*.

Gli **alberi decisionali** sono classificatori con struttura ad albero, simili a un diagramma di flusso, in cui ogni nodo interno rappresenta un determinato test effettuato su un singolo predittore e ogni ramo è una strada che viene percorsa in base all'esito del test. I nodi finali sono l'etichetta di classe a cui ogni dato può essere associato. I percorsi dalle radici alle foglie danno luogo alle regole di classificazione. Un esempio di albero decisionale è mostrato in Figura 2.5. Esistono sia gli alberi di classificazione che quelli di regressione **[8]** e il loro utilizzo dipende dal tipo di variabile che si vuole predire.

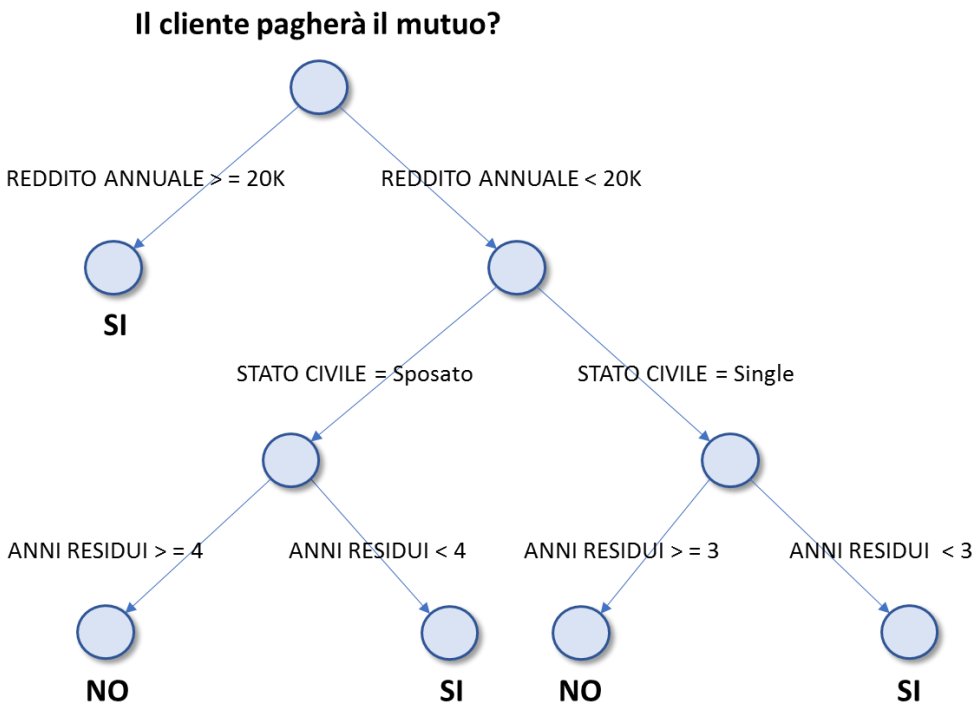

Figura 2.5: Esempio di albero decisionale

Il ***random forest*** è un modello che mette insieme i risultati di più alberi decisionali come mostrato in Figura 2.6. È un modello facilmente interpretabile, infatti, grazie alla sua struttura ad albero, si può capire quali siano stati i predittori più importanti nella separazione dei gruppi. Inoltre è anche piuttosto accurato siccome il suo risultato finale è ottenuto scegliendo i migliori risultati tra i vari alberi generati.

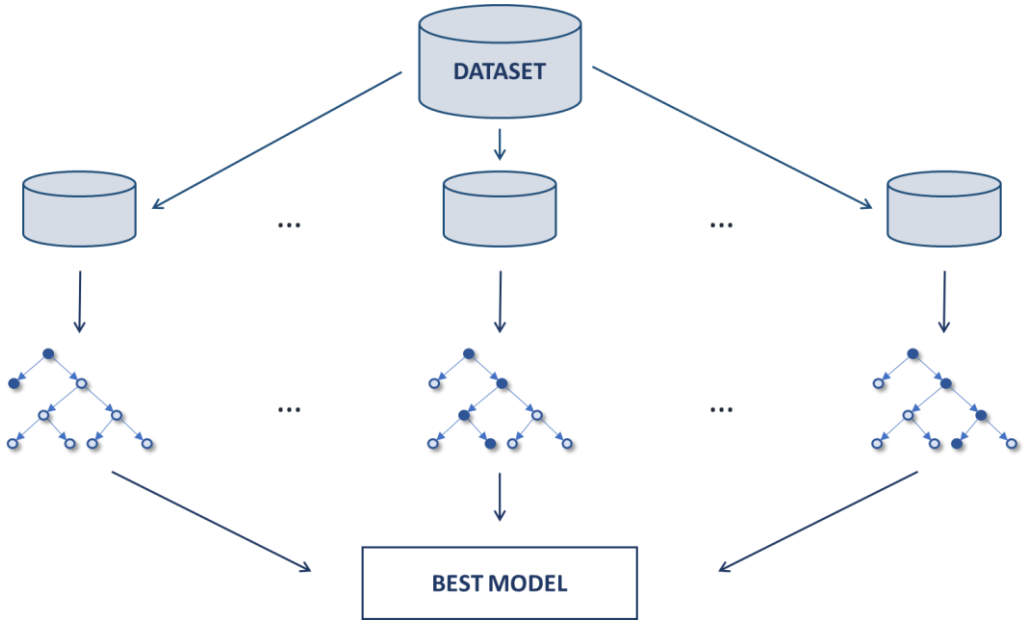

Figura 2.6: Esempio di *random forest*

Alberi decisionali con elevato numero di nodi e di divisioni possono portare ad un problema di *overfitting*, cioè il modello predice perfettamente i dati su cui è stato addestrato, ma non è affidabile per dati nuovi. Il modello risulta quindi troppo complesso per generalizzare la conoscenza. Viceversa, un problema di *underfitting* si manifesta quando il modello non è sufficientemente elaborato, caso che avviene sia quando si hanno pochi dati per addestrare il modello ma anche quando il modello è troppo semplice e non riesce a catturare la complessità dei dati. Il modello migliore si trova nel mezzo, ovvero un modello che riesca a descrivere bene i dati su cui è stato costruito e sia in grado di predire con una certa accuratezza i dati non ancora visti.

2.2.2 Modelli di regressione

I **modelli di regressione** si utilizzano, invece, quando l'output che si vuole ottenere è una variabile di tipo quantitativo. La regressione è quindi utilizzata quando i possibili valori che l'output può assumere sono

infiniti. La regressione è utile in ambiti come la previsione di valori azionari, che sono definiti in un insieme di valori numerici. In questa tesi, per via della natura quantitativa delle variabili cercate, saranno approfonditi i diversi tipi di regressione utilizzati. Di seguito viene fornita una panoramica su alcuni algoritmi di regressione utilizzati.

La **regressione lineare [9]** è un metodo che stima il valore atteso di una variabile dipendente y, dati i valori di altre variabili indipendenti. La regressione lineare semplice prevede la definizione di un'unica variabile indipendente X, mentre la regressione lineare multipla è l'estensione di quella semplice e considera un certo numero k di predittori $X_1, \dots, X_k$. Quest'ultimo modello assume la seguente forma:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k + \varepsilon$$

Dove X_j indica il j-esimo predittore, mentre β_j è un parametro che quantifica l'associazione tra questa variabile e la variabile dipendente y. I parametri $\beta_0, \dots, \beta_k$ sono incogniti e devono essere stimati in modo da poter fare le predizioni mediante la formula seguente:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \dots + \hat{\beta}_k x_k$$

Dove $\hat{y}$ indica una previsione di y sulla base di $X_1 = x_1, \dots, X_k = x_k$. Esistono diverse metodologie per stimare i parametri della funzione lineare, tra cui il metodo dei minimi quadrati, che minimizza la somma dei residui quadratici RSS (*Residual Sum of Squares*) definita come:

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \dots - \hat{\beta}_k x_{ik})^2$$

La differenza tra y_i e $\hat{y}_i$ esprime l'errore di predizione quando si sostituisce il valore esatto y_i con il valore predetto $\hat{y}_i$ dal modello lineare. In Figura 2.7 viene mostrato un esempio di come vengono calcolati gli errori nel caso di un modello lineare semplice. Perciò, la funzione lineare

che meglio approssima l'insieme di punti, è quella tale per cui tutti i punti risultano avere distanza quadratica minima.

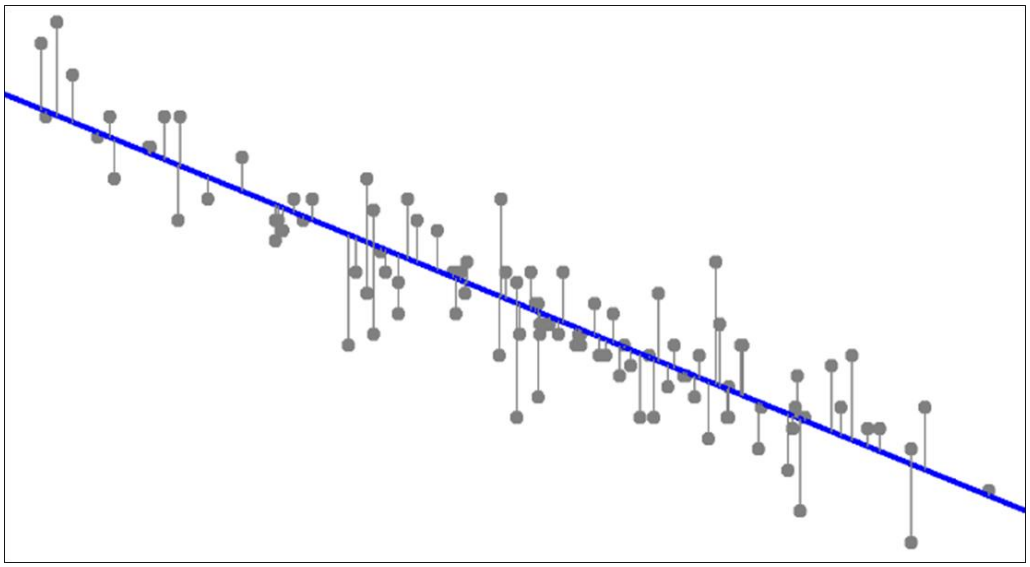

Figura 2.7: Metodo dei minimi quadrati

In alcuni casi, il legame tra la variabile y e i predittori $X_1, \dots, X_k$ potrebbe essere non lineare. Come mostrato in Figura 2.8 per estendere la regressione lineare si usa la **regressione polinomiale**, che utilizza l'interpolazione polinomiale dei dati. In generale si utilizzano polinomi di secondo grado, in questo caso il modello assume la seguente forma:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_{11} X_1^2 + \beta_{22} X_2^2 + \beta_{12} X_1 X_2 + \varepsilon$$

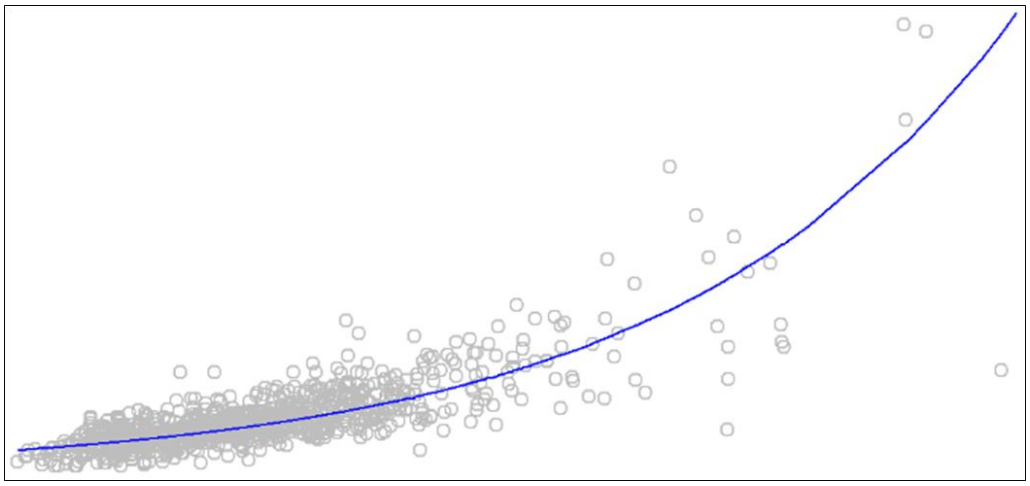

Figura 2.8: Regressione polinomiale

Il modello lineare può essere ulteriormente migliorato sostituendo al metodo dei minimi quadrati alcune procedure alternative. La **regressione lasso [10]** (*Least Absolute Shrinkage and Selection Operator*) è un metodo alternativo che considera tutti i predittori e utilizza una tecnica che regolarizza le stime dei coefficienti. In questo caso, i parametri del modello vengono scelti in modo tale da minimizzare una l'RSS in aggiunta ad un nuovo termine $\alpha \sum_{j=1}^{k} |\beta_j|$. La quantità che viene aggiunta è un termine di penalità dove α è un parametro che va scelto durante le prove sperimentali. La regressione lasso stima i parametri minimizzando l'RSS, tuttavia il termine di penalità diventa piccolo quando questi coefficienti sono vicini a zero. Quando il parametro α è sufficientemente grande, alcune stime possono essere esattamente uguali a zero, ciò significa che il metodo lasso può essere usato anche per eseguire una selezione di variabili significative per la predizione. L'effetto dell'aggiunta di questa penalità può ridurre significativamente la varianza presente nella stima dei parametri del modello lineare.

2.3 Interpretazione e validazione

Prima di utilizzare il modello, bisogna procedere alla sua validazione, per sapere quanto questo sia accurato e affidabile. Validare il modello sull'insieme di dati utilizzati per costruirlo, potrebbe condurre a risultati sbagliati a causa della sua eccessiva precisione. Solitamente, per valutare le performance di un modello predittivo si suddivide casualmente il dataset in due sottoinsiemi disgiunti chiamati *training set* e *test set*. Il *training set* è l'insieme dei dati che costituiscono la fase di addestramento del modello, mentre il *test set* è l'insieme dei dati che non sono stati coinvolti nella creazione del modello e che servono per

valutare l'errore di predizione commesso. Esistono diverse tecniche di partizionamento dei dati.

La tecnica più semplice è l'***holdout*** che partiziona i dati in maniera randomica in due insiemi indipendenti, tipicamente il 75% dei dati viene usato come *training set* e il restante 25% come *test set*.

Quando la numerosità del dataset non è sufficientemente grande, la suddivisione semplice in un insieme di *training* e un insieme di *test* può dare dei risultati fuorvianti per il calcolo dell'errore, perché magari lo si calcola su una partizione non significativa dei dati. Per evitare questo problema si può utilizzare la procedura ***cross validation***, la quale, come si può vedere in Figura 2.9, partiziona i dati in maniera randomica in k sottoinsiemi disgiunti, mutualmente esclusivi e di dimensione approssimativamente uguale, chiamati *k-fold*. Il *training* e il *test* vengono eseguiti k volte, tendendo da parte un *fold* ed utilizzando i rimanenti come *training*. Si ottengono in questo modo, diverse stime del tasso di errore il cui valore medio si chiama tasso di errore cross-validato.

Figura 2.9: Esempio di *Cross validation* con 5 *fold*

I modelli ottenuti dalle analisi menzionate nelle pagine precedenti richiedono un processo di studio e di verifica. In particolare, risulta fondamentale il supporto dell'esperto di dominio per confermare l'interpretazione di tali modelli o per evidenziare possibili inconsistenze. La presenza di inconsistenze nei risultati ottenuti implica un processo di retroazione agli step precedenti per effettuare migliorie o ulteriori iterazioni al fine di potenziare l'efficacia dei modelli estratti.

Inoltre, occorre un consolidamento della conoscenza estratta che, oltre a essere validata dall'esperto di dominio, sia oggettivamente verificata attraverso metodi statistici. Si avranno metodi diversi per valutare la qualità del risultato a seconda della tipologia di problema in questione.

Nel nostro caso, per valutare la qualità di un modello di regressione si può usare la statistica R^2. Nel caso della regressione lineare per esempio, questa statistica indica quanto i dati sono vicini alla retta di regressione generata. Il calcolo della statistica R^2 si ottiene mediante la seguente formula:

$$R^2 = \frac{TSS - RSS}{TSS} = \frac{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2 - \sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

Dove con il termine TSS (*Total Sum of Squares*) si intende la differenza quadratica di y con la sua media $\bar{y}$, ovvero la variabilità legata all'attributo target prima della creazione del modello di regressione. D'altra parte, il termine RSS misura la variabilità non spiegata che rimane dopo aver eseguito la regressione. Perciò, la variabilità di y spiegata dal modello usando i diversi predittori determina il valore di R^2. In generale, l'R^2 rappresenta una percentuale di varianza della variabile target e quindi può assumere valori compresi tra 0 e 1. Se il valore è prossimo allo 0 indica che il modello di regressione non riesce a spiegare quasi nulla della variabilità di y attorno alla sua media. Invece un valore

di R^2 prossimo a 1 significa che il modello di regressione spiega quasi tutta la variabilità della variabile target attorno alla propria media. In generale, più alto è l'R^2, meglio il modello si adatta ai dati. Più è alta la varianza che è rappresentata dal modello di regressione, più i punti cadranno vicini alla linea di regressione.

CAPITOLO 3

Caso di studio

Il caso di studio di riferimento riguarda il *Weight Management*, ovvero l'attività di analisi e ottimizzazione del peso veicolo, attraverso le varie fasi di sviluppo di un progetto.

La valutazione accurata di come si compone il peso del veicolo, in funzione degli elementi e sistemi che lo costituiscono, è divenuta particolarmente importante nel settore automobilistico, soprattutto in relazione alle nuove normative sulle emissioni di CO_2 che stanno entrando in vigore negli ultimi anni. Per l'arco temporale che va dal 2017 al 2021, entrano in vigore i nuovi cicli di omologazione (WLTP) per le vetture, e per i veicoli commerciali leggeri è stato introdotto il ciclo RDE (*Real Driving Emission*), ovvero un ciclo che abbina ai classici test di laboratorio una prova su strada che simula l'utilizzo reale dell'autovettura, con accelerazioni e decelerazioni relative.

In particolare, la definizione della procedura WLTP (*Worldwide Harmonised Light duty vehicle Test Procedure*), ha lo scopo di definire una tipologia di prova più rappresentativa dei cicli di guida reali. Pertanto, per determinare i valori di consumo carburante sulla base dei dati del ciclo di guida, rispetto al ciclo obbligatorio per legge NEDC (*New European Driving Cycle*), il WLTP è più vicino al comportamento di guida reale.

Quindi, secondo il nuovo schema, i veicoli risultano suddivisi in famiglie e, anziché dichiarare un valore di CO_2 per un'intera famiglia di veicoli (come precedentemente richiesto dalla normativa), a ciascun veicolo

all'interno di una famiglia è associato un valore di CO_2 basato sull'effetto combinato dei valori di massa, resistenza al rotolamento, e resistenza aerodinamica, determinati in base alla sua configurazione base e alla dotazione di optional.

Ciascun veicolo all'interno di una famiglia è caratterizzato da uno specifico valore di emissione CO_2 in base all'allestimento. È richiesto il test su almeno due veicoli differenti per famiglia:

- *Worst case* (*High*): si riferisce al veicolo della famiglia CO_2 con la configurazione di caratteristiche rilevanti (ovvero massa, resistenza aerodinamica e resistenza al rotolamento) che produce il fabbisogno di energia più elevato;
- *Best case* (*Low*): si riferisce al veicolo della famiglia CO_2 con la configurazione di caratteristiche rilevanti che produce il fabbisogno di energia più basso.

Ovvero, secondo la procedura WLTP, oltre al peso del veicolo in configurazione base, vengono conteggiati anche gli optional (OPT) presenti e il carico utile, ma per comodità il metodo stabilisce solo 2 versioni testabili della famiglia:

1. Veicolo che richiede la minor quantità di energia per avanzare (in genere senza OPT, minimo RR, minimo CX);
2. Veicolo che richiede la maggior quantità di energia per avanzare (in genere full OPT, massimo RR, massimo CX).

Tutte le altre versioni della famiglia dovranno ricadere all'interno della linea di regressione, che collega direttamente i 2 casi secondo il metodo di interpolazione come mostrato in Figura 3.1.

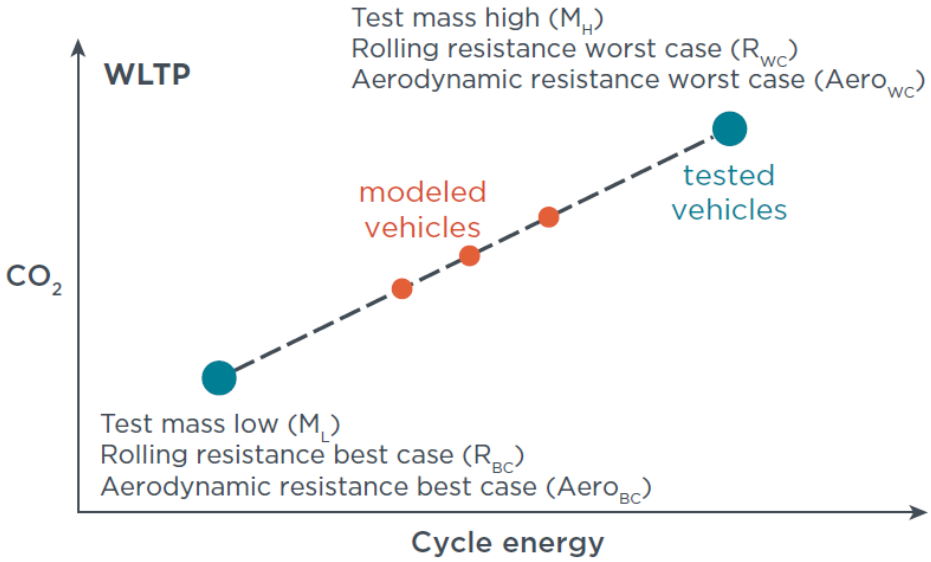

Figura 3.1: Metodo di interpolazione

In questo contesto diviene molto importante poter garantire una stima accurata della massa del veicolo in funzione del suo allestimento/configurazione.

3.1 Weight Management

Lo scopo delle attività di *Weight Management* è definire le regole operative e gli obiettivi per la gestione del peso dei veicoli, durante le varie fasi di sviluppo di un progetto.

Il processo di gestione della prestazione peso si compone delle seguenti attività principali:

- ***Benchmarking***: è l'analisi comparativa di un prodotto (veicolo) con i migliori concorrenti, nell'ambito di un perimetro omogeneo. Si analizzano i veicoli della concorrenza e si raccolgono dati per la definizione del peso target. La valutazione dei veicoli della concorrenza è utile per individuare possibili ottimizzazioni di peso:

l'obiettivo è definire il posizionamento del prodotto (veicolo) rispetto alla concorrenza ed evidenziarne punti di forza e debolezza. In quest'ottica si analizzano gli standard di prestazione, al fine di individuare la "*best performance*". In questa fase possono essere utili grafici che rappresentano il peso in funzione del volume e il peso in funzione della potenza, per avere un quadro di come sono posizionati i veicoli concorrenti rispetto al veicolo di riferimento. Il metodo di *Benchmarking* è uno strumento completo per confrontare i pesi dei sottosistemi veicolo, per la migliore impostazione del target di eventuale riduzione di massa. Il metodo utilizza analisi statistica di regressione, per individuare dove è possibile fare efficienza in termini di massa e confrontare gli alleggerimenti.

- **Definizione del peso target**: si definisce il peso target per ciascun sistema, sotto-sistema e singolo elemento, che compongono il peso totale del veicolo. La base dati in questa fase include: l'albero veicolo di riferimento (scomposizione del veicolo in sistemi, sotto-sistemi e componenti) e le distinte pesi dei veicoli di confronto. Attraverso analisi di dettaglio, per alcune componenti la stima di peso viene effettuata essenzialmente in base ai materiali e tramite il rilievo delle superfici. Tipicamente si tratta di componenti di carrozzeria e di interni.
- **Ottimizzazione peso**: sulla base di quanto emerso dalle fasi di *benchmarking* e definizione del peso target, si individuano e valutano le modifiche che si possono introdurre per ridurre il peso totale del veicolo o per ottimizzane la distribuzione.

3.2 Struttura dei dati

I dati da utilizzare per le analisi riguardano le pesature effettuate su differenti versioni di autoveicoli. Ogni versione è caratterizzata da un elenco di optional univoco, ovvero quella particolare combinazione di optional è presente solo in quella versione.

Inizialmente, sono stati forniti 8 file Excel suddivisi in:

- 7 file contenenti i dati che si riferiscono alle pesature effettuate separatamente per ogni semestre per gli anni che variano dal 2015 al 2018. Pertanto, il primo file conterrà le pesature effettuate nel primo semestre del 2015, mentre l'ultimo file conterrà le pesature effettuate nel primo semestre del 2018.
- 1 file unico contenente l'elenco degli optional disponibili su tutti i modelli che sono stati pesati dal primo semestre del 2015 al primo semestre del 2018.

3.2.1 Struttura dei dataset riguardanti le Pesate

Gli elementi contenuti nei file Excel riguardanti le pesate presentano in totale 12 colonne:

- DATAPESATURA: rappresenta la data in cui il veicolo ha subito il processo di pesatura.
- MDGP_CODE: rappresenta il codice di tipologie di versione veicolo.
- MDGP_DESC: rappresenta la descrizione di tipologie di versione veicolo.
- MRKT_CODE: rappresenta il codice di mercato.
- PRP_FDP_CODE: rappresenta la sigla dell'allestimento.
- PRP_MODEL_CODE: rappresenta la specializzazione della sigla dell'allestimento.

- VAN_CODE: rappresenta il codice del telaio, la chiave univoca che identifica la pesata.
- VCB_CODE: rappresenta la famiglia di veicoli.
- VP_CODE: rappresenta un allestimento specifico per una famiglia di veicoli.
- WEIGHT_AXE_1: rappresenta il peso dell'asse anteriore, una delle tre variabili da predire.
- WEIGHT_AXE_2: rappresenta il peso dell'asse posteriore, una delle tre variabili da predire.
- WEIGHT_TOTAL: rappresenta il peso totale, una delle tre variabili da predire.

Ogni riga descritta in questo formato rappresenta una singola pesata effettuata che viene identificata dalla chiave univoca VAN_CODE.
In totale si hanno 266.298 righe, quindi 266.298 pesate effettuate.

3.2.2 Struttura dei dataset riguardanti gli Optional

Gli elementi contenuti nel file Excel riguardante gli optional presentano 3 colonne:

- FDP_CODE: rappresenta la sigla dell'allestimento.
- VP_CODE: rappresenta un allestimento specifico per una famiglia di veicoli, la chiave univoca del singolo modello.
- CONFIGURATION: rappresenta l'elenco degli optional presenti sul veicolo, le variabili usate per addestrare il modello di predizione.

Ogni riga descritta in questo formato rappresenta un singolo modello che viene identificata dalla chiave univoca VP_CODE.
In totale si hanno 217.320 righe, quindi 217.320 modelli diversi.

3.3 Strumenti utilizzati

L'analisi dei dati è stata compiuta utilizzando il linguaggio Python **[11]** tramite Jupyter Notebook, un ambiente di sviluppo utilizzato soprattutto per questo linguaggio di programmazione. La motivazione principale che ha portato all'utilizzo di questo linguaggio è stata l'elevata dimensione dei dataset di input utilizzati per le analisi: infatti Python è uno dei linguaggi di programmazione più utilizzati per effettuare analisi da grandi quantità di dati.

Jupyter Notebook è un ambiente di sviluppo *open-source* accessibile tramite web. Questo permette di scrivere e avviare parti di codice in modo dinamico e, cosa più importante, ha accesso a molta documentazione per l'analisi dati. È stato utilizzato per la sperimentazione riguardante la predizione del valore del peso del veicolo ed è stato scelto perché possiede diverse librerie ottimizzate per l'analisi di dati che si sono rivelate fondamentali non solo per la formulazione dei modelli di predizione, ma anche proprio per la semplice gestione dei dati, grazie ad adeguate strutture interne.

Le principali librerie utilizzate sono riportate nel seguente elenco, insieme ad una sintetica descrizione delle operazioni che hanno permesso di svolgere:

- **Os**: è una libreria che permette di compiere diverse operazioni per far interagire il programma con il sistema operativo del computer (Windows, Linux o Mac OS). Nell'ambito di questa tesi questa libreria è stata utilizzata per leggere i nomi di alcuni file all'interno di una determinata cartella.
- **Pandas [12]**: è una libreria che permette di gestire i dati in forma tabellare (tramite i *DataFrames*) o sequenziale (tramite le *Series*). I

DataFrames sono delle strutture dati molto simili alle tabelle di un classico database che consentono, tra le altre cose, di indicizzare i dati e di manipolarli in maniera efficace. Le *Series* sono vettori monodimensionali, meno utilizzati in questa tesi rispetto ai *DataFrames*. Tra le sue più importanti caratteristiche c'è la capacità di compiere operazioni numeriche, statistiche e di visualizzare risultati in modo molto rapido e intuitivo, grazie alla grande varietà di funzioni disponibili, come ad esempio l'estrazione di solo alcune colonne o alcune righe. Oltre che per queste ragioni, Pandas è stata poi utilizzata per caricare e salvare file esterni in differenti formati, come ad esempio i CSV ed Excel.

- **Numpy [13]**: è una libreria specializzata in operazioni su vettori, matrici o altre strutture multidimensionali. È stata utilizzata per accedere alle funzioni matematiche da applicare ai dati in analisi.
- **Matplotlib [14]**: è una libreria grafica molto efficace e flessibile che consente la creazione di molti tipi di grafici in modo semplice e intuitivo. Durante questo lavoro è stata utilizzata questa libreria, poiché la visualizzazione grafica dei risultati ottenuti è stata una parte fondamentale del lavoro. Essa si è rivelata fondamentale in fase di pulizia del dataset per indirizzare le analisi verso una certa direzione. I grafici più utilizzati in questo lavoro sono stati gli istogrammi e i boxplot.
- **Scikit-learn [15]**: è una libreria che contiene una serie di strumenti per l'analisi dei dati e l'apprendimento automatico ed è stata progettata per essere usata insieme ad altre librerie come Numpy o Pandas. Contiene svariate classi e metodi per svolgere qualunque tipologia di analisi, grazie alla possibilità di implementare tutti i più importanti algoritmi di *data mining* come *clustering*, classificazione e

regressione. In questa tesi è stata utilizzata soprattutto per la costruzione e la validazione dei vari modelli di predizione progettati.

CAPITOLO 4

Metodologia proposta

Lo scopo del lavoro è stato quello di progettare e sviluppare una metodologia che permettesse, in ottica *data driven*, di predire il peso futuro di un veicolo, data una particolare configurazione di optional. Più precisamente, si è voluto capire quanto la presenza di un optional incidesse sul peso di un veicolo. Il lavoro è stato eseguito sia sul peso totale dei veicoli sia sulla ripartizione dei pesi su asse anteriore e posteriore.

Nei prossimi paragrafi saranno spiegati nei dettagli tutti i passaggi svolti e tutte le analisi effettuate per giungere all'obiettivo prefissato. Per ogni passaggio sarà spiegata la metodologia utilizzata e le scelte intraprese. Durante lo svolgimento del lavoro, spesso sono state sperimentate diverse soluzioni contemporaneamente e solo infine è stata scelta la migliore.

A partire dai dataset contenenti i dati delle pesature effettuate su differenti versioni (configurazioni) di autoveicoli e l'elenco degli optional disponibili su ogni modello, lo schema di massima dell'attività di analisi è mostrato in Figura 4.1. Di seguito sono presentati i principali blocchi caratterizzanti il lavoro svolto:

- **Raccolta dati**: fa parte della fase di selezione che riguarda la scelta dei dati rilevanti rispetto all'analisi.
- **Pulizia dati**: comprende la fase di *pre-processing* nella quale si cerca di eliminare i dati rumorosi.

- **Trasformazione dati**: in questa fase si eseguono le correzioni sul dataset per ottenere la struttura più utile rispetto all'obiettivo prefissato.
- **Creazione modello**: rappresenta la fase di creazione e validazione del modello predittivo.

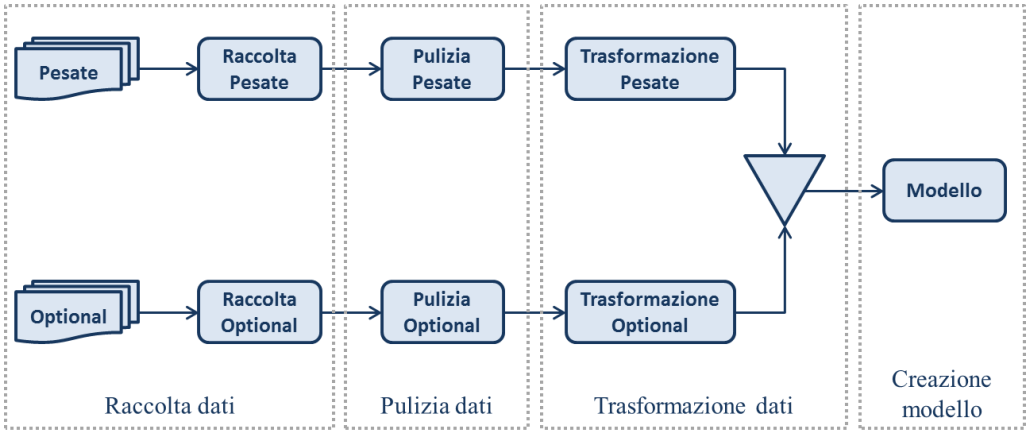

Figura 4.1: Schema delle attività

4.1 Raccolta dati

Inizialmente sono stati trasformati i file Excel in file CSV per rendere più agibile l'esportazione e l'importazione delle loro informazioni. Questi file CSV sono stati inseriti in due cartelle: all'interno della cartella "Pesate" sono stati inseriti i 7 file che contengono i dati delle pesate, mentre nella cartella "Optional" è stato inserito il file contenente l'elenco degli optional disponibili su ogni modello.

4.1.1 Raccolta Pesate

Per leggere tutti i file CSV all'interno della cartella "Pesate" è stato scritto uno script in Python. Questo script è stato scritto con il supporto delle librerie os e pandas. In particolare, della libreria os è stato utilizzato il metodo os.listdir() che restituisce una lista con tutti i nomi

dei file presenti all'interno del percorso inserito tra le parentesi. Il motivo per cui è stato scelto questo metodo risiede nella comodità di non dovere leggere tutti i nomi dei file manualmente, ma soprattutto perché nel caso in cui si dovessero ricevere nuovi file relativi alle pesate (ad esempio con i dati del secondo semestre del 2017) basterebbe inserirli all'interno della cartella "Pesate" e in automatico verrebbero inseriti all'interno della lista con tutti i nomi dei file. Della libreria pandas invece è stato utilizzato prima il metodo pd.read_csv() per inserire in un *DataFrame* uno per volta i file CSV presenti all'interno della nostra lista e poi il metodo pd.concat() per concatenarli. Questo secondo metodo è efficace anche nel caso in cui si aggiunga un nuovo file con colonne diverse. Infatti, oltre a concatenare liste che presentano tutte le colonne in comune, la concatenazione avviene comunque anche quando, come è accaduto tra il 2015 e il 2016, alcune colonne vengono aggiunte o eliminate.

Di seguito è stato riportato lo script utilizzato:

```
default_dir_pesate = 'C:\\Users\\valbol\\Desktop\\Tesi\\Python\\Pesate'
pesate = os.listdir(default_dir_pesate)

pesate_list = [ ]
for filename in pesate:
    path = os.path.join(default_dir_pesate, filename)
    pesate_csv = pd.read_csv(path, sep = ' ; ')
    pesate_list.append(pesate_csv)
pesate_df = pd.concat(pesate_list, sort = True)
pesate_df.info( )
```

Il numero totale delle pesate effettuate è 266.298, di queste si è deciso di selezionarne solo una parte significativa per le analisi. Questa scelta è dettata dal voler definire un peso di riferimento per ogni VP_CODE e, essendo presenti VP_CODE caratterizzati da una bassa numerosità, essi sono poco rappresentativi e non consentono di produrre successivamente delle stime affidabili. Pertanto si è deciso di considerare un VP_CODE se e solo se esso ha una numerosità maggiore di β. β è un parametro

modificabile che indica la cardinalità dei record per definire il peso di riferimento di ogni VP_CODE. Durante le analisi, con l'aiuto degli esperti di dominio, sono stati valutati diversi valori e il valore più opportuno nel nostro caso è stato il valore 5. Ciò nonostante, nel caso in cui si dovessero ricevere nuovi dati questo valore potrebbe essere modificato. Per fare ciò è stato scritto uno script in Python che utilizza il metodo groupby() della libreria pandas per raggruppare i record separatamente per ogni VP_CODE. Lo script utilizzato è il seguente:

```
selected_list = [ ]
for key, group in pesate_df.groupby( 'VP_CODE' ):
    if(len(group)>beta):
        selected_list.append(group)
pesate_selected_df = pd.concat(selected_list, sort = True)
pesate_selected_df.info( )
```

4.1.2 Raccolta Optional

Per leggere il file CSV all'interno della cartella "Optional" è stato scritto uno script in Python. Questo script è più semplice di quello riguardante le pesate perché si tratta di un solo file, infatti per leggere il file CSV utilizza solo il metodo pd.read_csv della libreria pandas. Di seguito è stato riportato lo script:

```
path = 'C:\\Users\\valbol\\Desktop\\Tesi\\Python\\Optional\\File.csv'
optional_df = pd.read_csv(path, sep = ' ; ')
optional_df
```

4.2 Pulizia dati

La fase di pulizia di dei dati è una fase determinante soprattutto per quanto riguarda i dati relativi alle pesate. Questi dati potrebbero contenere errori compiuti nell'inserimento dei dati ma soprattutto dei

possibili *outlier*. Per quanto riguarda i dati relativi agli optional invece è stato verificato che tra i modelli presenti non ci fossero valori mancanti.

4.2.1 Pulizia Pesate

La prima attività di pulizia è stata quella di trovare e rimuovere tutti i possibili errori che sono stati commessi nell'inserimento dei dati, perché questi errori potrebbero influenzare la fase di identificazione degli *outlier*. Inizialmente si è cercato di stabilire due valori soglia, un valore minimo e un valore massimo, al di fuori dei quali i dati sarebbero stati considerati errati. Questa metodologia però presentava due tipi di problemi: il primo problema risiedeva nella necessità di dover individuare diversi valori soglia per il peso anteriore, per il peso posteriore e per il peso totale. Il secondo problema, più limitante, era legato al fatto di dover trovare valori soglia diversi per ogni VP_CODE. Pertanto, per questi dati la definizione di valori soglia non è un'operazione che si può stabilire a priori. A seguito di queste motivazioni, con l'aiuto degli esperti di dominio, come risoluzione sono stati individuati 3 vincoli applicabili a tutto il dataset. Il primo vincolo prevede che i valori di peso anteriore, posteriore e totale siano maggiori di zero. Il secondo impone che la somma del peso anteriore e posteriore non debba variare del 10% rispetto al peso totale. Infine, con il terzo vincolo si richiede che il peso anteriore sia sempre maggiore del peso posteriore. Tutti i valori che non rispettano anche solo uno di questi vincoli sono considerati errori. Per realizzare ciò è stato utilizzato il seguente script di Python:

```
err_list = [ ]

Axe1 = pesate_selected_df [ 'WEIGHT_AXE_1' ]
Axe2 = pesate_selected_df [ 'WEIGHT_AXE_2' ]
Total = pesate_selected_df [ 'WEIGHT_TOTAL' ]
```

```
vincolo1 = (Axe1<=0) | (Axe2<=0) | (Total<=0)
vincolo2 = (Axe1+Axe2<0.9*Total) | (Axe1+Axe2>1.1*Total)
vincolo3 = (Axe1<Axe2)

pesate_err = pesate_df[ ~ ( vincolo1 | vincolo2 | vincolo3 )]
err_list.append(pesate_err)
pesate_err_df = pd.concat(err_list, sort = True)
pesate_err_df.info( )
```

Dopo la rimozione degli errori, la seconda attività di pulizia che è stata eseguita riguarda la rimozione degli *outlier* presenti per ogni VP_CODE tra i valori di peso anteriore, posteriore e totale. Poiché non è possibile sapere a priori quale metodo potrebbe essere efficace nel caso in specie, sono stati sperimentati due metodi: una rimozione univariata con il boxplot e una rimozione multivariata con l'EllipticEnvelope.

Per quanto riguarda la rimozione degli outlier attraverso il boxplot, in Python non esiste un metodo che rimuova automaticamente gli *outlier*. Quindi per fare ciò è stato scritto uno script in Python che rimuove tutti i valori che sono minori di $Q1 - 1.5 \cdot IQR$ oppure maggiori di $Q3 + 1.5 \cdot IQR$. Se almeno uno tra i valori di peso anteriore, peso posteriore e peso totale non è compreso tra i valori minimi e massimi sopra citati, quel dato viene considerato *outlier* perché lo scopo è quello di ottenere un modello di predizione per ogni tipo di peso, ossia un modello per peso anteriore, posteriore e totale. Lo script utilizza il metodo groupby() della libreria pandas per raggruppare i record separatamente per ogni VP_CODE. Lo script è il seguente:

```
out_list = [ ]

for key, group in pesate_err_df.groupby( 'VP_CODE' ):
    if (len(group_out) > beta):
        AXE1 = group[ [ 'WEIGHT_AXE_1' ] ]
        AXE2 = group[ [ 'WEIGHT_AXE_2' ] ]
        TOTAL = group[ [ 'WEIGHT_TOTAL' ] ]

        Q1_Axe1 = AXE1.quantile(0.25)
        Q3_Axe1 = AXE1.quantile(0.75)
```

```
        IQR_Axe1 = Q3_Axe1 - Q1_Axe1
        outlier_Axe1 = ( AXE1 < (Q1_Axe1-1.5*IQR_Axe1) )
                       ( AXE1 > (Q3_Axe1+1.5*IQR_Axe1) )

        Q1_Axe2 = AXE2.quantile(0.25)
        Q3_Axe2 = AXE2.quantile(0.75)
        IQR_Axe2 = Q3_Axe2 - Q1_Axe2
        outlier_Axe2 = ( AXE2 < (Q1_Axe2-1.5*IQR_Axe2) )
                       ( AXE2 > (Q3_Axe2+1.5*IQR_Axe2) )

        Q1_Total = TOTAL.quantile(0.25)
        Q3_Total = TOTAL.quantile(0.75)
        IQR_Total = Q3_Total - Q1_Total
        outlier_Total = ( TOTAL < (Q1_Total -1.5*IQR_Total) )
                        ( TOTAL > (Q3_Total+1.5*IQR_Total) )

        group_out = group[ ~ ( (outlier_Axe1).any(axis=1) |
                               (outlier_Axe2).any(axis=1) |
                               (outlier_Total).any(axis=1) ) ]
        if (len(group_out) > beta):
            out_list.append(group_out)
pesate_out_df = pd.concat(out_list, sort = True)
pesate_out_df.info()
```

Per quanto riguarda l'EllipticEnvelope, la rimozione degli *outlier* è stata eseguita attraverso uno script in Python tramite un metodo predisposto appositamente dalla libreria Scikit-learn. Si tratta del metodo EllipticEnvelope().fit_predict(), il quale restituisce il valore 1 se il dato è un *inlier* e il valore -1 se il dato è un *outlier*. Dopo aver compiuto le giuste sperimentazioni, si è deciso di utilizzare come coppia di variabili di input "peso anteriore e peso posteriore" perché questi due valori sono tra loro indipendenti e permettono di ottenere il risultato più appropriato. Le altre coppie di variabili, rispettivamente "peso anteriore e peso totale" e "peso posteriore e peso totale", non hanno portato agli stessi risultati poiché sono variabili dipendenti tra di loro. Lo script utilizzato è il seguente:

```
out_list = [ ]
for key, group in pesate_err_df.groupby( 'VP_CODE' ):
```

```
    if (len(group_out) > beta):
        AXE1_AXE2 = group[ [ 'WEIGHT_AXE_1', 'WEIGHT_AXE_2' ] ]
        out = EllipticEnvelope(random_state=0).fit_predict(AXE1_AXE2)
        group_out = group[ out==1 ]
        if (len(group_out) > beta):
            out_list.append(group_out)
pesate_out_df = pd.concat(out_list, sort = True)
pesate_out_df.info( )
```

4.2.2 Pulizia Optional

Per quanto riguarda i dati relativi agli optional per verificare che tra i modelli presenti non ci fossero valori mancanti è stato utilizzato il metodo isna() della libreria pandas che permette di identificare le colonne con valori nulli. Di seguito viene riportato lo script utilizzato:

```
null = optional_df[ 'CONFIGURATION' ].isna()
optional_null_df = optional_df[ ~null ]
optional_null_df
```

4.3 Trasformazione dati

Prima di elaborare i modelli di predizione, è stata eseguita la fase di trasformazione dei dati. Per i dati relativi alle pesate, è stato assegnato un peso di riferimento per ogni VP_CODE, che sarà la variabile da predire. Mentre per i dati relativi agli optional, questa fase è stata di fondamentale importanza perché ha permesso di eseguire le opportune correzioni sulla colonna CONFIGURATION, nella quale sono presenti gli optional che verranno utilizzati per allenare i modelli di predizione.

4.3.1 Trasformazione Pesate

Per assegnare un peso di riferimento si è deciso di raggruppare i dati separatamente per VP_CODE, in modo che ogni modello fosse rappresentato non più dall'insieme dei suoi valori, ma da alcune metriche

di statistica descrittiva in grado di raggruppare l'informazione circa la distribuzione di tali dati.

Le metriche calcolate sono elencate di seguito:

- media del peso anteriore;
- media del peso posteriore;
- media del peso totale;
- deviazione standard del peso anteriore;
- deviazione standard del peso posteriore;
- deviazione standard del peso totale.

Prima di assegnare i pesi di riferimento però sono stati solamente selezionati i gruppi di VCB_CODE tali che la numerosità di VP_CODE diversi all'interno fosse maggiore di α. α è la cardinalità dei record per creare e testare il modello di predizione. Per valutare la bontà del modello durante la sperimentazione sono stati scelti diversi valori di α che saranno specificati con maggiore dettaglio nel capitolo successivo. Di seguito è riportato lo script utilizzato per la trasformazione dei dati relativi alle pesate:

```
distinct = pesate_out_df.groupby( 'VCB_CODE' ).VP_CODE.nunique()
filtered = distinct[distinct>alfa]
mask = pesate_out_df[ 'VCB_CODE' ].isin(filtered.index)
pesate_filtered_df = pesate_out_df.loc[mask]

grouped_df = pesate_filtered_df.groupby( 'VP_CODE' )
avg = grouped_df[['WEIGHT_AXE_1',
                  'WEIGHT_AXE_2',
                  'WEIGHT_TOTAL']].mean()
dev = grouped_df[['WEIGHT_AXE_1',
                  'WEIGHT_AXE_2',
                  'WEIGHT_TOTAL']].std()
stats_df = pd.merge(avg, dev, left_on='VP_CODE',
                              right_on='VP_CODE',
                              suffixes=( '_AVG', '_DEV' ))
stats_df.describe( )
```

4.3.2 Trasformazione Optional

Il primo passaggio è stato quello di trasformare la colonna CONFIGURATION perché, come mostrato in Figura 4.2, presentava una struttura in cui i valori erano separati dal carattere " | ". A causa di questa struttura la colonna non poteva essere utilizzata per la creazione del modello di predizione.

CONFIGURATION
CMY00 | 79295 | COC00 | 14987 | D0000 | 02007 | D1000 | 00137

Figura 4.2: Colonna CONFIGURATION

Considerando come primo valore il valore 0, i valori pari rappresentavano la famiglia di optional, mentre i valori dispari rappresentavano il tipo di optional che possiede ogni modello.

La trasformazione attuata su tale colonna ha permesso di distribuire i dati su varie colonne, come mostrato in Figura 4.3, così da permettere il loro utilizzo per la creazione del modello.

CMY00	COC00	D0000	D1000
79295	14987	02007	00137

Figura 4.3: Colonne Optional

Per alleggerire il file da processare e per eliminare le colonne che contenevano informazioni non interessanti ai fini del progetto, si è deciso di tenere solo le famiglie di optional che influenzano il peso del veicolo, ovvero solamente le colonne formate da 5 caratteri. Per fare ciò è stato scritto il seguente script in Python:

```
joined_df = pd.merge(stats_df, optional_null_df, left_on = 'VP_CODE',
                                                 right_on = 'VP_CODE' )
```

```
colonne = [ ]
for i in range(len(joined_df[ 'CONFIGURATION' ])):
    lista = joined_df[ 'CONFIGURATION' ] [ i ].split(" | ")
    for j in range( len(lista) - 1):
        if(j%2==0)&(len(lista[j])==5):
            colonne.append(lista[j])
opt = sorted(list(set(colonne)))

valori = [ ]
for i in range(len(joined_df[ 'CONFIGURATION' ])):
    valori_lista = [ ]
    lista = joined_df[ 'CONFIGURATION' ] [ i ].split(" | ")
    for k in range(len(opt)):
        trovato = 0
        for j in range( len(lista) - 1):
            if(j%2==0)&(len(lista[j])==5)&(lista[j]==opt[k])&(trovato==0):
                valori_lista.append(lista[j+1])
                trovato = 1
        if(trovato==0):
            valori_lista.append( 'NULL' )
    valori.append(valori_lista)

opt_df = pd.DataFrame(valori, columns = opt)
opt_df
```

L'ultima attività è stata quella di convertire qualsiasi colonna categorica in una colonna numerica. Questo perché i modelli di predizione funzionano su dati numerici e non riescono ad utilizzare dati non numerici senza alcun tipo di trasformazione. È importante prestare attenzione durante la conversione di una variabile categorica in una variabile numerica. Un approccio sbagliato potrebbe essere quello di assegnare un numero a ciascun possibile valore della colonna, così facendo si introduce un ordine tra i valori che prima non esisteva. Pertanto si è deciso di applicare la codifica *One-hot*. Una colonna che può avere N valori distinti è stata convertita in N colonne booleane: per ogni riga, solo una delle N colonne sarà impostata su 1 (ovvero la colonna associata al valore di quella riga per la colonna originale), le altre saranno impostate su 0.

Per fare ciò è stata utilizzata la libreria pandas che offre il metodo pd.get_dummies() per convertire le colonne categoriche in colonne con codifica *One-hot* come mostrato nel seguente script:

```
opt_dummy_df = pd.get_dummies(opt_df, columns = opt_df.columns)
opt_dummy_df.info()
```

4.3.3 Join tra Pesate e Optional

Dopo aver pulito e trasformato i dati riguardanti le pesate e gli optional per ottenere il dataset finale è stato fatto un JOIN usando come chiave univoca VP_CODE tramite il metodo pd.merge() della libreria pandas. Lo script utilizzato è il seguente:

```
final_df = pd.merge(joined_df, opt_dummy_df, left_index = True,
                                                right_index = True)
final_df
```

4.4 Creazione modello

Una volta ottenuto il dataset finale, si è passati alla creazione e validazione dei modelli di predizione. La bontà dei modelli è stata valutata confrontando i risultati delle predizioni con i dati appartenenti ad un dataset di test.

È stato scritto uno script in Python che si occupa di separare le colonne degli optional dalla colonna da predire (peso anteriore, peso posteriore o peso totale) e crea due diverse strutture di dati, denominate X e y. Queste due strutture dati vengono inserite nel metodo train_test_split() della libreria scikit-learn che permette di dividere il dataset in *training set* e *test set*, così facendo si ottengono due partizioni di X e y. X_train e y_train saranno utilizzate per addestrare il modello di regressione, X_test sarà impiegata per eseguire le predizioni sui valori di peso scelto, y_test

sarà utilizzata per quantificare la bontà del modello ottenuto. Lo script di Python utilizzato è il seguente:

```
X = final_df.drop(columns=final_df.columns[:10]).values
y = final_df[final_df.columns[3]].values

X_train, X_test, y_train, y_test = train_test_split(X, y)
```

La natura dei dati da predire ha influenzato la scelta della tecnica di *data mining* da eseguire. Infatti, trattandosi di valori numerici si è pensato di utilizzare la regressione. Tra gli algoritmi di regressione esistenti sono stati sperimentati la regressione lasso polinomiale e il *random forest*.

In Python entrambi gli algoritmi non possono essere utilizzati senza aver inserito all'interno alcuni parametri che permettono di ottenere un risultato più accurato. La scelta di questi parametri è un'operazione decisiva per la creazione del regressore e, per questo motivo, la libreria scikit-learn fornisce la classe GridSearchCV() al fine di semplificare la ricerca dei parametri da inserire. Anche questa classe richiede alcuni parametri:

- reg: è il tipo di regressore che si deve validare.
- param_grid: rappresenta un dizionario chiave – valore dove ogni chiave rappresenta il nome di uno specifico parametro del regressore e ogni valore corrisponde una lista di valori che esso può assumere. Tale attributo crea diversi regressori con i parametri indicati e sceglie il modello che massimizza la metrica scelta nel campo scroring.
- scroring: è la metrica che permette di valutare la bontà dei modelli creati, nel nostro caso trattandosi di regressori sarà utilizzato l'R^2.
- cv: serve per settare il parametro di partizione del dataset con la procedura *cross validation*, in questo caso si è deciso di dividere il dataset in 5 sezioni, delle quali a turno ognuna svolgerà il ruolo di *test set* e le altre il ruolo di *training set*.

Per l'implementazione del *random forest* la libreria scikit-learn dà la possibilità di usare la classe RandomForestRegressor() come mostrato nel seguente script:

```
reg = RandomForestRegressor()

param_grid = { "n_estimators": [100, 250, 500],
               "criterion": ["mse", "mae"],
               "max_features": ["auto", "sqrt", "log2"],
               "random_state": [42],
               "n_jobs": [-1] }

gridsearch = GridSearchCV(reg, param_grid, scoring="r2", cv=5)

res = gridsearch.fit(X_train, y_train)

print("Best model configuration is:", res.best_params_)
print("with R2=%.2f" % res.best_score_)

final_model = res.best_estimator_
final_model.fit(X_train, y_train)
y_test_pred = final_model.predict(X_test)

r2_score(y_test, y_test_pred)
```

Per implementare invece il modello di regressione lasso polinomiale la libreria scikit-learn mette a disposizione il metodo make_pipeline() all'interno del quale vengono inserite PolynomialFeatures(2), Lasso() come mostrato nel seguente script di Python:

```
reg = make_pipeline(PolynomialFeatures(2), Lasso())

param_grid = { 'lasso__alpha' : [0.1, 0.2, 0.4],
               'lasso__tol' : [0.1, 0.2, 0.4] }

gridsearch = GridSearchCV(reg, param_grid, scoring = 'r2', cv = 5)

res = gridsearch.fit(X_train, y_train)

print("Best model configuration is:", res.best_params_)
print("with R2=%.2f" % res.best_score_)

final_model = res.best_estimator_
final_model.fit(X_train, y_train)
y_test_pred = final_model.predict(X_test)

r2_score(y_test, y_test_pred)
```

CAPITOLO 5

Risultati sperimentali

Come è già stato anticipato nel capitolo precedente questo lavoro di tesi è stato articolato in 4 fasi. Durante ogni fase sono state sperimentate diverse soluzioni in parallelo che hanno portato a diversi risultati intermedi.

La maggior parte dei risultati intermedi sono stati salvati su file Excel o CSV, mentre altri sono stati riportati su grafici.

Nei prossimi paragrafi saranno mostrati nei dettagli tutti i risultati acquisiti durante ogni fase e le scelte intraprese per giungere all'obiettivo finale.

5.1 Raccolta dati

Durante la prima fase del lavoro è stata eseguita una selezione dei dati. Questa selezione è stata eseguita per definire un peso di riferimento per ogni VP_CODE. Per fare ciò è necessario che ogni VP_CODE abbia un numero totale di pesature che lo renda affidabile, pertanto il processo di analisi è stato compiuto solo sui VP_CODE con numerosità maggiore di parametro β scelto uguale a 5. A seguito di questa selezione sono stati eliminati 101.746 dati sul totale di 266.298, ovvero il 38,21%.

Infine è stata compiuta un'analisi esplorativa dei dati selezionati per cogliere gli aspetti più critici. Focalizzandosi sulle tre variabili da predire, come si può notare in Figura 5.1, le distribuzioni si sovrappongono, pertanto prima di eseguire la fase di pulizia dei dati si è

deciso di procedere con un raggruppamento dei pesi separatamente per VP_CODE.

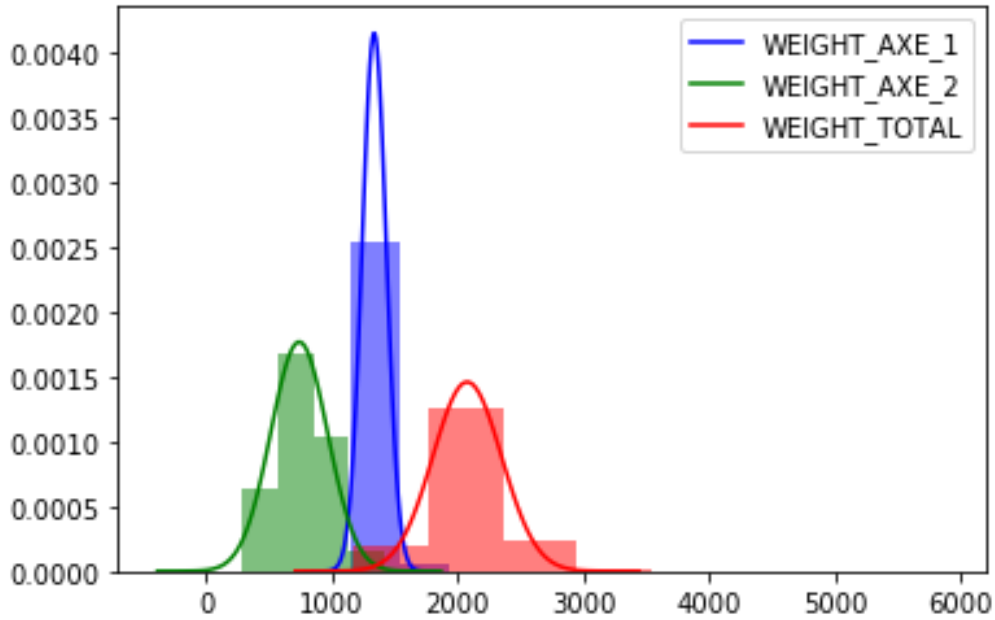

Figura 5.1: Distribuzioni iniziali peso anteriore, posteriore e totale

Inoltre sono state estratte le statistiche più rilevanti come media, deviazione standard, minimo, 25° percentile, 50° percentile, 75° percentile e massimo. Queste statistiche sono state inserite in Tabella 5.1, dove si può notare che per ognuna delle tre variabili considerate ci sono valori di pesi non corretti. Considerando per esempio il peso anteriore, possiamo notare come la sua media sia 1336 kg ma sono presenti anche valori non accettabili come 0 kg oppure 3847 kg. Esaminando il peso posteriore invece, si nota una media di 739 kg ma ancora dei valori come 0 kg oppure 2833 kg. Infine, per quanto riguarda il peso totale, con una media di 2075 kg contiene anche valori sempre di 0 kg oppure di 5891 kg.

	WEIGHT_AXE_1	WEIGHT_AXE_2	WEIGHT_TOTAL
count	164552.000000	164552.000000	164552.000000
mean	1336.169995	739.427044	2075.597039
std	96.034805	225.453673	273.246127
min	0.000000	0.000000	0.000000
25%	1282.000000	601.000000	1896.000000
50%	1322.000000	642.000000	2026.000000
75%	1381.000000	893.000000	2280.000000
max	3847.000000	2833.000000	5891.000000

Tabella 5.1: Statistiche iniziali peso anteriore, posteriore e totale

Per mostrare i passaggi successivi sono stati scelti due VP_CODE caratterizzati da un'elevata differenza di peso e numerosità: il primo è un modello più leggero con una numerosità di 1171, mentre il secondo è un modello più pesante con una numerosità di 21. Come si può vedere in Figura 5.2, nonostante i due modelli siano tra loro differenti, se i tre pesi vengono raggruppati per VP_CODE vi è una netta separazione.

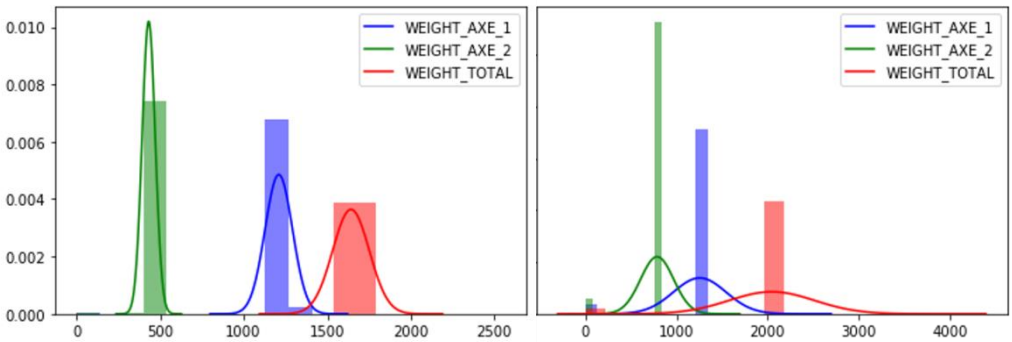

Figura 5.2: Distribuzioni iniziali di due VP_CODE

Anche in questo caso però, come si può dedurre dalle statistiche riportate in Tabella 5.2, per entrambi i VP_CODE il raggruppamento non esclude la presenza di pesi non corretti e di valori anomali.

	WEIGHT_AXE_1	WEIGHT_AXE_2	WEIGHT_TOTAL	WEIGHT_AXE_1	WEIGHT_AXE_2	WEIGHT_TOTAL
count	1171.000000	1171.000000	1171.000000	21.000000	21.000000	21.000000
mean	1209.968403	431.806149	1641.774552	1255.095238	788.857143	2043.952381
std	82.158612	39.111948	109.515860	287.713035	180.881532	468.484095
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1209.000000	428.000000	1638.000000	1313.000000	822.000000	2142.000000
50%	1214.000000	431.000000	1645.000000	1318.000000	828.000000	2145.000000
75%	1218.000000	435.000000	1652.000000	1323.000000	830.000000	2150.000000
max	1410.000000	1336.000000	2560.000000	1341.000000	844.000000	2179.000000

Tabella 5.2: Statistiche iniziali di due VP_CODE

Per questo motivo, la fase di pulizia dei dati, risulta essere fondamentale.

5.2 Pulizia dati

Prima di procedere con la rimozione degli outlier, si è notato che nei dati selezionati erano presenti degli errori commessi durante l'inserimento dei dati. Questo si può notare facilmente guardando le statistiche dei dati in cui si trovano dei valori non logicamene accettabili come 0 kg. Inoltre, con l'aiuto degli esperti di dominio, si è scoperto che il peso anteriore deve essere sempre maggiore del peso posteriore. Altri errori si possono trovare quando la somma del peso anteriore e posteriore varia di più del 10% del peso totale. E' stato necessario pertanto eseguire una prima fase di pulizia inserendo dei vincoli per eliminare questi errori. Sono stati posti i seguenti vincoli:

- Peso anteriore > 0 & Peso posteriore > 0 & Peso totale > 0
- Peso anteriore + Peso posteriore > 0,9* Peso totale &
  Peso anteriore + Peso posteriore < 1,1*Peso totale
- Peso anteriore > Peso posteriore

Applicando questi vincoli sono stati eliminati 917 dati sui 164.552 selezionati inizialmente, ovvero lo 0,56%.

Come si può notare in Figura 5.3 la rimozione dei dati errati determina un miglioramento delle condizioni. Infatti, l'ampiezza delle distribuzioni gaussiane sembra essere diminuita e, per quanto riguarda il secondo VP_CODE i valori sembrano rientrare tutti all'interno delle distribuzioni.

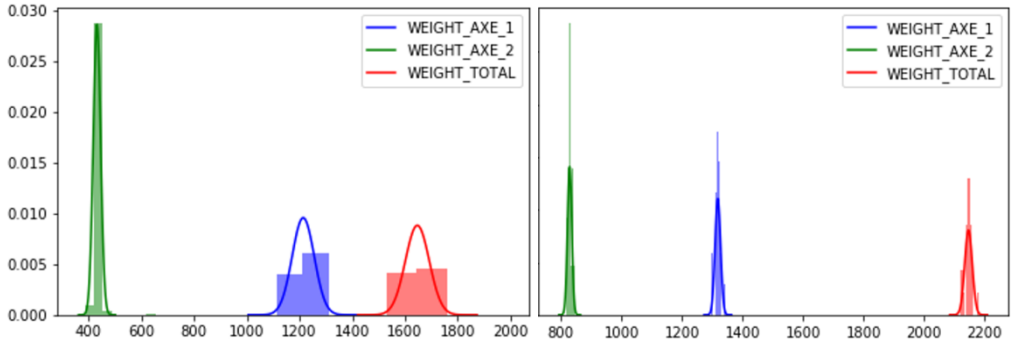

Figura 5.3: Distribuzioni senza errori di due VP_CODE

Guardando anche le statistiche riportate in Tabella 5.3, la rimozione degli errori ha determinato valori statistici più ragionevoli. Per quanto riguarda il primo VP_CODE sono diminuite le deviazioni standard ma le medie sono rimaste quasi invariate. Mentre per quanto riguarda il secondo VP_CODE la rimozione degli errori oltre ad aver diminuito notevolmente le deviazioni standard, ha portato una grande variazione dei valori medi. In particolare, il peso anteriore è passato da 1255 kg a 1317 kg, il peso posteriore da 788 kg a 828 kg e il peso totale da 2043 kg a 2146 kg. Questo è dovuto al fatto che un VP_CODE con bassa numerosità è più influenzato dalla presenza di errori.

	WEIGHT_AXE_1	WEIGHT_AXE_2	WEIGHT_TOTAL	WEIGHT_AXE_1	WEIGHT_AXE_2	WEIGHT_TOTAL
count	1166.000000	1166.000000	1166.000000	20.000000	20.000000	20.000000
mean	1214.107204	432.512007	1646.619211	1317.850000	828.300000	2146.150000
std	41.653379	13.922758	45.202994	9.027881	7.079251	12.393865
min	424.000000	389.000000	847.000000	1298.000000	817.000000	2121.000000
25%	1209.000000	428.000000	1638.250000	1313.000000	822.750000	2142.750000
50%	1214.000000	431.000000	1645.000000	1318.000000	828.000000	2146.000000
75%	1218.000000	435.000000	1652.000000	1323.000000	831.000000	2150.000000
max	1410.000000	719.000000	1987.000000	1341.000000	844.000000	2179.000000

Tabella 5.3: Statistiche senza errori di due VP_CODE

Una volta eliminati gli errori, si passa alla seconda fase di pulizia, nella quale si cerca di trovare ed eliminare gli *outlier* con l'utilizzo di due diversi metodi: il boxplot e l'EllipticEnvelope.

5.2.1 Boxplot

Applicando il boxplot per l'*outlier detection* sono stati eliminati 25.443 dati sui 163.635 rimasti dopo la pulizia degli errori, ovvero il 15,55%.

L'*outlier detection* con il boxplot ha migliorato notevolmente le distribuzioni. Come si può notare in Figura 5.4 l'ampiezza delle distribuzioni gaussiane è diminuita e, per entrambi i VP_CODE i valori rientrano tutti all'interno delle distribuzioni.

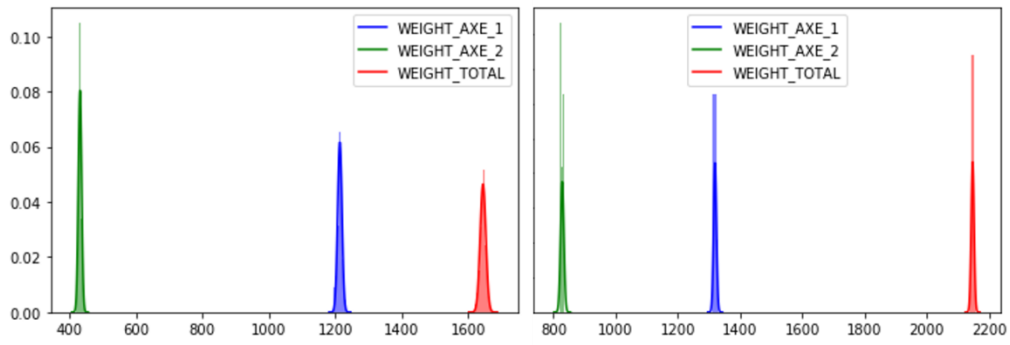

Figura 5.4: Distribuzioni senza *outlier* con il boxplot di due VP_CODE

Inoltre, i valori statistici dopo l'*outlier detection* con il boxplot riportati in Tabella 5.4, mostrano per entrambi i VP_CODE delle ottime riduzioni delle deviazioni standard.

	WEIGHT_AXE_1	WEIGHT_AXE_2	WEIGHT_TOTAL	WEIGHT_AXE_1	WEIGHT_AXE_2	WEIGHT_TOTAL
count	990.000000	990.000000	990.000000	15.00000	15.000000	15.00000
mean	1213.061616	430.979798	1644.041414	1319.20000	828.000000	2147.20000
std	6.472876	4.956188	8.590051	4.64758	5.318432	4.61674
min	1196.000000	418.000000	1619.000000	1312.00000	822.000000	2140.00000
25%	1209.000000	428.000000	1639.000000	1316.50000	823.500000	2144.50000
50%	1214.000000	431.000000	1644.000000	1319.00000	828.000000	2147.00000
75%	1217.000000	434.000000	1650.000000	1323.00000	830.000000	2150.00000
max	1231.000000	445.000000	1671.000000	1328.00000	838.000000	2158.00000

Tabella 5.4: Statistiche senza *outlier* con il boxplot di due VP_CODE

5.2.2 EllipticEnvelope

Applicando invece l'EllipticEnvelope per l'*outlier detection* sono stati eliminati 23.046 dati sui 163.635 rimasti dopo la pulizia degli errori, ovvero il 14,08%.

Anche con il metodo dell'EllipticEnvelope l'*outlier detection* porta a un elevato miglioramento delle distribuzioni. Anche in questo come si evince in Figura 5.5 l'ampiezza delle distribuzioni gaussiane è diminuita e, per entrambi i VP_CODE i valori rientrano tutti all'interno delle distribuzioni.

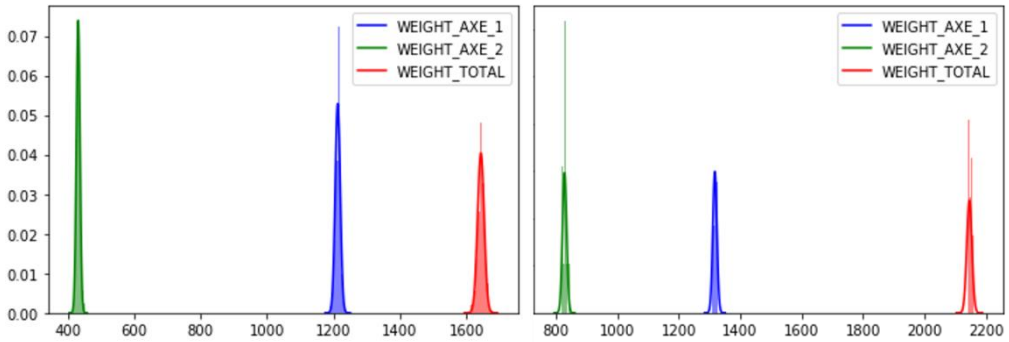

Figura 5.5: Distribuzioni senza *outlier* con l'EllipticEnvelope di due VP_CODE

Guardando anche le statistiche riportate in Tabella 5.5, i valori statistici dopo l'*outlier detection* con l'EllipticEnvelope, mostrano per entrambi i VP_CODE delle ottime riduzioni delle deviazioni standard.

	WEIGHT_AXE_1	WEIGHT_AXE_2	WEIGHT_TOTAL	WEIGHT_AXE_1	WEIGHT_AXE_2	WEIGHT_TOTAL
count	1049.000000	1049.000000	1049.000000	18.000000	18.000000	18.000000
mean	1212.778837	431.085796	1643.864633	1317.444444	828.277778	2145.722222
std	7.543430	5.395103	9.875792	6.679400	6.737128	8.414312
min	1187.000000	412.000000	1614.000000	1298.000000	817.000000	2126.000000
25%	1209.000000	428.000000	1638.000000	1313.750000	823.250000	2143.250000
50%	1214.000000	431.000000	1644.000000	1318.000000	828.000000	2146.000000
75%	1217.000000	434.000000	1650.000000	1322.250000	830.000000	2150.000000
max	1237.000000	450.000000	1679.000000	1328.000000	844.000000	2160.000000

Tabella 5.5: Statistiche senza *outlier* con l'EllipticEnvelope di due VP_CODE

I due metodi presentano alcune differenze come ad esempio il numero di dati eliminati: il boxplot rimuove il 15,55% dei dati, mentre l'EllipticEnvelope ne elimina solamente il 14,08%. Un'altra differenza che si può notare riguarda i valori delle deviazioni standard: infatti, il boxplot rimuovendo più dati presenta dei valori di deviazione standard leggermente inferiori rispetto a quelli dell'EllipticEnvelope.

Nonostante queste piccole differenze, ciò che è più rilevante è che i due metodi presentano approssimativamente le stesse medie per i valori di peso anteriore, posteriore e totale. Dato che i risultati dei due metodi sono approssimativamente uguali si può supporre che la media ottenuta sia una buona stima della media del campione. Per questo motivo è stato deciso di utilizzarli entrambi per la predizione e scegliere, in seguito ai risultati, quale metodo sia il migliore.

Osservando invece i valori medi prima e dopo questa fase, soprattutto per il VP_CODE che presenta una bassa numerosità, risultano consistenti differenze, per cui si evince che la fase di pulizia è stata fondamentale

per assegnare pesi di riferimento corretti e garantire buoni risultati nel modello di predizione.

5.3 Trasformazione dati

Come è già stato anticipato nel capitolo precedente, per assegnare un peso di riferimento sono stati trasformati i dati lasciando solamente per ogni VP_CODE i seguenti dati:

- media del peso anteriore;
- media del peso posteriore;
- media del peso totale;
- deviazione standard del peso anteriore;
- deviazione standard del peso posteriore;
- deviazione standard del peso totale.

Le medie sono state utilizzate per addestrare il modello di predizione, mentre le deviazioni standard sono importanti per capire fino a che punto il valore di peso predetto dal modello può essere considerato esatto.

Per valutare la bontà del modello, durante la sperimentazione sono stati selezionati i gruppi di VCB_CODE tali che la numerosità di VP_CODE diversi all'interno fosse maggiore di un parametro α. Sono stati scelti diversi di valori di α, che corrispondono a diverse famiglie di veicoli mostrate in Tabella 5.6.

α	VCB_CODE	VP_CODE
75	**1**	78
70	**3**	78+73+72
45	**8**	78+73+72+59+47+47+47+46

Tabella 5.6: Famiglie di veicoli al variare di α

Nel primo caso con $\alpha = 75$ si crea un modello con una sola famiglia di veicoli ma con pochi dati per addestrarlo. Mentre negli altri due casi al diminuire di α aumentano le famiglie di veicoli all'interno del modello e aumentano anche i dati per l'addestramento. Non sono stati scelti valori di α inferiori perché troppi dati richiedevano troppo tempo per la creazione dei modelli.

5.4 Creazione modello

Per la creazione del modello, tra gli algoritmi di regressione esistenti, sono stati sperimentati la regressione lasso polinomiale e il *random forest*. Per valutare la qualità dei modelli generati si è deciso di usare la statistica R^2 che, in seguito ai risultati ottenuti, ha permesso di valutare quale metodo fosse il migliore.

5.4.1 Regressione lasso polinomiale

Per creare il primo modello predittivo è stata scelta la regressione lasso polinomiale ed è stata applicata per ognuna delle tre variabili da predire. In questo modo sono stati creati tre modelli: un modello per il peso anteriore, uno per il peso posteriore e uno per il peso totale.

Di seguito sono elencati per ogni modello i vari valori di R^2 ottenuti al variare sia del metodo utilizzato per rimuovere gli *outlier* sia del numero di famiglie di veicoli.

Per il **peso anteriore**, come mostrato in Tabella 5.7, il valore di R^2 migliore è ottenuto utilizzando il metodo boxplot per la rimozione degli *outlier*, con un valore di α pari a 45 che corrisponde a 8 famiglie di veicoli.

	Boxplot	EllipticEnvelope
$\alpha = 75$	47.07%	53.90%
$\alpha = 70$	96.73%	93.73%
$\alpha = 45$	97.70%	97.29%

Tabella 5.7: Valori di R^2 del peso anteriore per la regressione

Per il **peso posteriore** invece, come mostrato in Tabella 5.8, il valore di R^2 migliore è ottenuto utilizzando il metodo boxplot per la rimozione degli *outlier*, con un valore di α pari a 70 che corrisponde a 3 famiglie di veicoli.

	Boxplot	EllipticEnvelope
$\alpha = 75$	28.77%	41.30%
$\alpha = 70$	98.16%	96.47%
$\alpha = 45$	93.05%	97.41%

Tabella 5.8: Valori di R^2 del peso posteriore per la regressione

Per il **peso totale**, come mostrato in Tabella 5.9, il valore di R^2 migliore è ottenuto utilizzando il metodo boxplot per la rimozione degli *outlier*, con un valore di α pari a 70 che corrisponde a 3 famiglie di veicoli.

	Boxplot	EllipticEnvelope
$\alpha = 75$	41.17%	77.18%
$\alpha = 70$	99.01%	98.55%
$\alpha = 45$	98.83%	98.33%

Tabella 5.9: Valori di R^2 del peso totale per la regressione

Per i tre modelli di predizione si osserva come il miglior valore di R^2 è sempre ottenuto scegliendo il metodo del boxplot. Tuttavia, anche il metodo EllipticEnvelope restituisce valori di R^2 ottimi, per cui a prescindere dal metodo, i risultati sono soddisfacenti. Gli unici risultati insoddisfacenti si ottengono con un valore di α pari a 75 che corrisponde ad una sola famiglia di veicoli, perché per l'addestramento il modello sono stati utilizzati pochi dati.

5.4.2 Random forest

Per creare il secondo modello predittivo invece è stato scelto il *random forest*, anch'esso applicato ad ognuna delle tre variabili da predire. Anche in questo caso sono stati creati tre modelli: un modello per il peso anteriore, uno per il peso posteriore e uno per il peso totale.

Di seguito vengono elencati i vari valori di R^2 per ogni modello al variare del metodo utilizzato per rimuovere gli *outlier* e del numero di famiglie di veicoli.

Per il **peso anteriore**, come mostrato in Tabella 5.10, il valore di R^2 migliore è ottenuto utilizzando il metodo boxplot per la rimozione degli *outlier*, con un valore di α pari a 45 che corrisponde a 8 famiglie di veicoli.

	Boxplot	EllipticEnvelope
α = 75	45.21%	90.75%
α = 70	97.09%	97.22%
α = 45	97.64%	97.05%

Tabella 5.10: Valori di R^2 del peso anteriore per il *random forest*

Per il **peso posteriore** invece, come mostrato in Tabella 5.11, il valore di R^2 migliore è ottenuto utilizzando il metodo boxplot per la rimozione degli *outlier*, con un valore di α pari a 70 che corrisponde a 3 famiglie di veicoli.

	Boxplot	EllipticEnvelope
$\alpha = 75$	45.68%	68.69%
$\alpha = 70$	99.80%	98.53%
$\alpha = 45$	94.14%	99.54%

Tabella 5.11: Valori di R^2 del peso posteriore per il *random forest*

Per il **peso totale**, come mostrato in Tabella 5.12, il valore di R^2 migliore è ottenuto utilizzando il metodo boxplot per la rimozione degli *outlier*, con un valore di α pari a 45 che corrisponde a 8 famiglie di veicoli.

	Boxplot	EllipticEnvelope
$\alpha = 75$	91.13%	83.72%
$\alpha = 70$	99.17%	99.08%
$\alpha = 45$	99.35%	98.89%

Tabella 5.12: Valori di R^2 del peso totale per il *random forest*

Anche in questo caso si osserva che per i tre modelli di predizione il miglior valore di R^2 è sempre ottenuto scegliendo il metodo del boxplot. Tuttavia, anche in questo caso il metodo EllipticEnvelope restituisce valori di R^2 ottimi, per cui a prescindere dal metodo, i risultati sono soddisfacenti. Diversamente dalla regressione lasso polinomiale, in questo caso si ottengono buoni valori di R^2 anche utilizzando un valore di α pari a 75, ciò significa che il *random forest* funziona bene anche con pochi dati.

5.5 Valutazione optional

Per capire quanto la presenza di un optional possa incidere sul peso di un veicolo, sono stati selezionati gli optional più rilevanti per ogni modello di predizione. Per ogni variabile di interesse sono stati scelti i modelli con l'R^2 maggiore.

Per il **peso anteriore** il modello migliore è risultato il modello di regressione lasso polinomiale che utilizza il metodo del boxplot per l'*outlier detection* con un valore di α pari a 45. In questo modello, che presenta un R^2 del 97.70%, gli optional più rilevanti sono:

- **CAMBIO DI VELOCITÀ**: 2840.6 O.D.
- **CAMBIO DI VELOCITÀ**: 2835.6 D.O.D.
- **GIRI MOTORE**: 3500
- **GIRI MOTORE**: 3900

Questi optional, oltre ad avere un ruolo decisivo nel calcolo del peso anteriore, hanno permesso al modello di predizione di distinguere le 8 famiglie di veicoli che abbiamo inserito per addestrarlo.

Per il **peso posteriore** invece, il modello migliore è risultato il *random forest* che utilizza il metodo del boxplot per rimuovere gli *outlier* con un valore di α pari a 70. In questo modello, che presenta il valore di R^2 più alto pari al 99.80%, gli optional più importanti sono:

- **PARACABINA PER CASSONE**: SENZA PARACABINA
- **CASSONE E SOVRASTRUTTURA**: SENZA CASSONE
- **LUCI SAGOMA POSTERIORE**: SENZA LUCI

In questo caso invece, questi optional hanno ruolo più importante nel calcolo del peso posteriore perché il modello di predizione è stato addestrato utilizzando solo 3 famiglie di veicoli.

per il **peso totale**, il modello migliore è risultato il *random forest* che utilizza il metodo del boxplot per l'*outlier detection* con un valore di α pari a 45. In questo modello, che presenta un valore di R^2 molto alto pari al 99.35%, gli optional più importanti sono:

- **VARIAZIONE CARICO SU ASSI**: 2100-2600
- **CASSONE E SOVRASTRUTTURA**: SENZA CASSONE
- **LUCI SAGOMA POSTERIORE**: SENZA LUCI
- **PARACABINA PER CASSONE**: SENZA PARACABINA

Anche in questo caso, oltre ad avere un ruolo decisivo nel calcolo del peso anteriore, questi optional hanno permesso al modello di predizione di distinguere le 8 famiglie di veicoli che abbiamo inserito per addestrarlo. Si può notare come l'optional più importante non fosse presente nei modelli di peso anteriore e posteriore, mentre gli altri erano già presenti nel modello del peso posteriore.

Conclusioni e sviluppi futuri

La seguente tesi è stata svolta in collaborazione con AMET, azienda che opera principalmente nel settore automobilistico. È stato proposto un modello che permettesse di predire il peso di un veicolo, data la sua configurazione di optional e i risultati sono stati soddisfacenti. Tra gli obiettivi finali però, non vi era solamente quello di creare un modello di predizione del peso, ma anche quello di capire quanto la presenza di un optional incidesse sul peso di un veicolo. A livello teorico, gli esperimenti hanno permesso di capire quanto la presenza di un optional possa impattare sul peso anteriore, sul peso posteriore e sul peso totale di un veicolo. Da questa conoscenza, la creazione di un nuovo modello di veicolo potrà essere ottimizzata, scegliendo la combinazione ottimale di optional da inserire. Inoltre, se si conoscessero i costi di produzione, i costi dei materiali e le quantità prodotte si potrebbe capire anche quanto influisca la presenza o meno di un optional sul costo totale del veicolo.

Affinché il modello progettato sia usabile e mantenibile si potrebbe pensare di inserire un'interfaccia grafica. L'interfaccia grafica scelta dovrà recuperare i dati di interesse utilizzando il linguaggio Python.

Potrebbe essere divisa in 4 sezioni:

1. La prima sezione riguarderà la fase di **selezione dei dati**, nella quale si inseriranno dei parametri che vengono definiti dall'utente. Ad esempio, se l'utente vorrà scegliere solo i modelli con una certa numerosità oppure se vorrà analizzare solo i dati di anno specifico, ci vorrà un'interfaccia grafica, dove potrà inserire questi parametri.
2. La seconda sezione dell'interfaccia grafica riguarderà la fase di **pulizia dei dati**, dove l'utente sceglierà il metodo o i metodi che vorrà utilizzare per la pulizia dei dati.

3. La terza sezione riguarderà la fase di **creazione del modello**, dove l'utente sceglierà la tecnica o le tecniche di regressione che vorrà utilizzare.
4. La quarta fase riguarderà l'**esplorazione del risultato**. Eventualmente si potrà inserire una sezione **comparativa** per confrontare i risultati fatte con sessioni di analisi diverse. In questo caso sarebbero 5 sezioni.

BIBLIOGRAFIA E SITOGRAFIA

[1] AMET. url: http://www.amet.it/

[2] Brachman, Anand. "*The Process of Knowledge Discovery in Database*". Advances in knowledge discovery and data mining, 1996

[3] Aggarwal, Yu. "*Outlier Detection for High Dimensional Data*". ACM SIGMOD Record, 2001.

[4] Michael Galarnyk. "*Understanding Boxplots*". url: https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51

[5] Robust covariance estimation and Mahalanobis distances relevance. url: https://scikit-learn.org/stable/auto_examples/covariance/plot_mahalanobis_distances.html

[6] Kumar, Steinbach, Tan. "*Introduction to Data Mining*". McGraw Hill, 2006.

[7] Han, Kamber, Pei. "*Data Mining: Concepts and Techniques*". Morgan Kaufmann Publishers, 2012.

[8] Breiman. "*Classification and Regression Trees*". Routledge, 2017.

[9] Montgomery, Peck, Vining. "*Introduction to Linear Regression Analysis*". Wiley, 2012.

[10] Tibshirani. "*Regression Shrinkage and Selection Via the Lasso*". Royal Statistical Society, 1996.

[11] Python Software Foundation. Python Language Reference, version 3.8.2. url: https://docs.python.org/3/

[12] Pandas. url: https://pandas.pydata.org/

[13] Numpy. url: https://numpy.org/

[14] Matplotlib. url: https://matplotlib.org/

[15] Scikit-learn. url: https://scikit-learn.org/stable/

RINGRAZIAMENTI

Desidero ringraziare tutte le persone che mi hanno supportato nel mio percorso.

Ringrazio tutti i dipendenti di AMET per avermi proposto questa tesi e per il supporto fornitomi durante la stesura.

Ringrazio la Prof.ssa Tania Cerquitelli per avermi permesso di lavorare a questo progetto di tesi. I suoi consigli sono stati fondamentali per la realizzazione di questo lavoro.

Ringrazio la Dott.ssa Evelina Di Corso per avermi seguito e sostenuto sia sul lato tecnico che sul lato emotivo, la sua pazienza e disponibilità sono state fondamentali. Inoltre, ringrazio il Dott. Andrea Pasini per aver speso del tempo per aiutarmi a risolvere le difficoltà riscontrate.

Ringrazio i miei compagni di corso e i miei amici di Torino che hanno rallegrato le lunghe giornate durante le sessioni esami e mi hanno regalato momenti indimenticabili.

Ringrazio i miei amici storici che, nonostante la distanza che ci separa da anni, ci sono sempre stati.

Ringrazio la mia ragazza, che mi ha supportato e sopportato durante la stesura di questa tesi ed è sempre stata al mio fianco in questi due anni. Se sono riuscito a terminare questo percorso è anche grazie a lei.

Infine, un ringraziamento speciale va ai miei genitori, senza i loro sacrifici non avrei potuto portare a termine questo percorso universitario.