Modello Personalizzato Ibrido per Analisi Commessa

Andrea Roberto Benvenuti
Analisi dei Componenti e Implementazione

4 settembre 2025

Indice

1 Introduzione

1.1 Motivazione

I modelli di previsione standard (ARIMA, ETS, regressione lineare) spesso falliscono nel catturare la complessità dei processi produttivi industriali, dove convivono:

- **Relazioni deterministiche**: più commesse → più produzione
- **Pattern stagionali**: agosto sempre più basso, dicembre picco
- **Dinamiche stocastiche**: effetti di momentum, shock esterni
- **Cambiamenti strutturali**: nuovi macchinari, processi

1.2 Approccio Ibrido

Il modello proposto adotta un approccio **ensemble multi-componente** che combina:

1. **Componente Deterministico** - Relazione quantità-commesse
2. **Componente Stagionale Adattiva** - Pattern ricorrenti
3. **Modello ARIMA sui Residui** - Dinamiche stocastiche
4. **Trend Breaks Detection** - Cambiamenti strutturali
5. **Ensemble Pesato** - Combinazione intelligente

2 Componente 1: Relazione Deterministica Quantità-Commesse

2.1 Concetto Fondamentale

Il primo componente modella la relazione tra il numero di commesse e la quantità prodotta attraverso una regressione non-lineare:

```
lm_commesse <- lm(qta_prodotta_tot ~ n_commesse + I(n_commesse^2),
                  data = train_data)
```

Listing 1: Modello deterministico

2.2 Formula Matematica

La relazione è espressa come:

$$Q_t = \alpha + \beta_1 \cdot N_t + \beta_2 \cdot N_t^2 + \epsilon_t \tag{1}$$

Dove:

$$Q_t = \text{Quantità prodotta al tempo } t \tag{2}$$
$$N_t = \text{Numero di commesse al tempo } t \tag{3}$$
$$\alpha = \text{Produzione base (intercetta)} \tag{4}$$
$$\beta_1 = \text{Effetto lineare delle commesse} \tag{5}$$
$$\beta_2 = \text{Effetto quadratico delle commesse} \tag{6}$$
$$\epsilon_t = \text{Termine di errore} \tag{7}$$

2.3 Perché Non Lineare?

2.3.1 Scenario A: Efficienze di Scala ($\beta_2 > 0$)

Esempio pratico:

- 1 commessa → 100 kg (molti setup, inefficienza)
- 2 commesse → 250 kg (batch più grandi)
- 3 commesse → 450 kg (economia di scala)

Interpretazione: Più commesse permettono ottimizzazioni che aumentano l'efficienza oltre il rapporto lineare.

2.3.2 Scenario B: Saturazione ($\beta_2 < 0$)

Esempio pratico:

- 1 commessa → 100 kg (tutto fluido)
- 2 commesse → 180 kg (qualche coda)
- 3 commesse → 240 kg (colli di bottiglia)

Interpretazione: Troppe commesse saturano la capacità produttiva con rendimenti decrescenti.

2.4 Gestione dei Residui

Dopo aver stimato la relazione, i residui contengono tutto ciò che non è spiegato dalle commesse:

```
residui_commesse <- residuals(lm_commesse)
```

Listing 2: Calcolo residui

Contenuto dei residui:

- Variazioni stagionali
- Trend temporali
- Shock casuali
- Effetti di momentum

2.5 Stima delle Commesse Future

Per le previsioni, il modello stima automaticamente il numero di commesse future usando medie stagionali ponderate:

```
for(i in seq_along(mesi_futuri)) {
    mese_target <- mesi_futuri[i]
    commesse_stesso_mese <- train_data$n_commesse[mesi_storici == mese_
        target]

    # Media ponderata: piu peso ai dati recenti
    weights <- exp(seq(-2, 0, length.out = length(commesse_stesso_mese))
        )
    n_commesse_future[i] <- weighted.mean(commesse_stesso_mese, weights,
         na.rm = TRUE)
}
```

Listing 3: Stima commesse future

Logica: "A marzo degli scorsi anni quante commesse abbiamo avuto? Usiamo quello per stimare marzo del prossimo anno, dando più peso agli anni recenti."

3 Componente 2: Stagionalità Adattiva

3.1 Concetto di Stagionalità

La stagionalità rappresenta **pattern ricorrenti** che si ripetono in cicli regolari:

Esempi tipici nell'industria:

- **Agosto**: sempre più basso (ferie)
- **Dicembre**: picco per consegne di fine anno
- **Gennaio**: calo post-festività
- **Primavera**: aumento per rilanci commerciali

3.2 Decomposizione STL

Il modello utilizza la decomposizione STL (Seasonal and Trend decomposition using Loess):

```
ts_obj <- ts(values, frequency = period)
decomp <- stl(ts_obj, s.window = "periodic", robust = TRUE)
```

Listing 4: Decomposizione STL

La decomposizione separa:

$$X_t = T_t + S_t + R_t \tag{8}$$

Dove:

$$X_t = \text{Dati originali} \tag{9}$$
$$T_t = \text{Componente di trend} \tag{10}$$
$$S_t = \text{Componente stagionale} \tag{11}$$
$$R_t = \text{Residui} \tag{12}$$

3.3 Caratteristiche Avanzate

3.3.1 Robustezza (robust = TRUE)

Il parametro `robust = TRUE` rende l'algoritmo resistente agli outlier:

Senza robustezza:

Gen: 800, Feb: 900, Mar: 1000, Apr: 10000 (errore!), Mag: 1200
→ La stagionalità di Aprile sarebbe distorta dall'errore

Con robustezza:

Gen: 800, Feb: 900, Mar: 1000, Apr: 10000 (errore!), Mag: 1200
→ L'algoritmo riconosce l'anomalia e la ignora

3.3.2 Finestra Stagionale (s.window = "periodic")

Questo parametro definisce come cambia la stagionalità nel tempo:

- `s.window = "periodic"`: stagionalità fissa nel tempo
- `s.window = 7`: stagionalità può cambiare ogni 7 osservazioni

3.4 Fallback Intelligente

Se STL fallisce, il modello usa un metodo di backup:

```
seasonal_means <- tapply(values, rep(1:period, length.out = length(
    values)),
                         mean, na.rm = TRUE)
```

Listing 5: Metodo fallback

Calcola semplicemente:

- Media di tutti i Gennaio
- Media di tutti i Febbraio
- etc.

3.5 Esempio Concreto

3.5.1 Step 1: Dati Storici

2023: Gen=800, Feb=900, Mar=1000, ..., Ago=400, ..., Dic=1500
2024: Gen=850, Feb=950, Mar=1050, ..., Ago=450, ..., Dic=1600

3.5.2 Step 2: STL Estrae la Stagionalità

Gen: -50 (sempre sotto la media)
Feb: +20 (leggermente sopra)
Mar: +80 (picco primaverile)
...
Ago: -350 (crollo estivo)
...
Dic: +400 (picco natalizio)

3.5.3 Step 3: Previsione per 2025

Trend di base per 2025: 1100
+ Stagionalità Agosto: -350
= Previsione Agosto 2025: 750

4 Componente 3: Modello ARIMA sui Residui

4.1 Concetto dei Residui

I residui sono quello che **rimane** dopo aver tolto la logica deterministica:

```
# Prima togli la relazione commesse
residui_commesse <- residuals(lm_commesse)

# Poi togli anche la stagionalita
detrended <- residui_commesse - seasonal_comp
```

Listing 6: Calcolo residui finali

In pratica:

$$\text{Dati Originali} = \text{Rel. Commesse} + \text{Stagionalità} + \text{RESIDUI} \tag{13}$$

4.2 Cosa Contengono i Residui

I **RESIDUI** contengono:

- Variazioni casuali non prevedibili
- Pattern temporali sottili
- Effetti di "memoria" (quello che succede oggi influenza domani)
- Shock esterni non catturati dagli altri componenti

4.3 Perché ARIMA

ARIMA = **A**uto**R**egressive **I**ntegrated **M**oving **A**verage
È perfetto per i residui perché cattura **3 tipi di dipendenza temporale**:

4.3.1 1. AutoRegressive (AR): "Il passato influenza il presente"

$$R_t = \phi_1 R_{t-1} + \phi_2 R_{t-2} + \cdots + \epsilon_t \tag{14}$$

Esempio: Se a Febbraio hai prodotto +100 kg oltre il previsto, probabilmente a Marzo produrrai ancora qualcosa di extra (momentum, scorte, etc.)

4.3.2 2. Moving Average (MA): "Gli shock passati influenzano il presente"

$$R_t = \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \epsilon_t \tag{15}$$

Esempio: Se a Febbraio c'è stato un guasto (-200 kg), a Marzo recupererai parte della produzione persa (+100 kg)

4.3.3 3. Integrated (I): "Ci sono trend persistenti nei residui"

$$\Delta R_t = R_t - R_{t-1} \tag{16}$$

Esempio: Se negli ultimi mesi i residui stanno crescendo (+10, +20, +30), continueranno a crescere

4.4 Selezione Automatica del Modello

```
arima_residui <- tryCatch({
    auto.arima(detrended, seasonal = FALSE, stepwise = FALSE,
               approximation = FALSE)
}, error = function(e) {
    arima(detrended, order = c(1,1,1))
})
```

Listing 7: Auto-selezione ARIMA

Parametri:

- `seasonal = FALSE`: stagionalità già rimossa
- `stepwise = FALSE`: testa TUTTE le combinazioni
- `approximation = FALSE`: calcolo esatto

4.4.1 Possibili Risultati:

- **ARIMA(1,0,0)**: Solo autoregression
- **ARIMA(0,1,1)**: Random walk con MA
- **ARIMA(2,1,2)**: Modello complesso

4.5 Esempio Pratico Completo

4.5.1 Situazione:

- **Marzo 2025**: 8 commesse previste
- **Componente 1**: 8 commesse → 1100 kg
- **Componente 2**: Marzo stagionalmente → +80 kg
- **Subtotale**: 1180 kg

4.5.2 Analisi Residui Storici:

Feb 2025: residuo +60 kg
Gen 2025: residuo +40 kg
Dic 2024: residuo +20 kg

4.5.3 ARIMA rileva:

"I residui stanno crescendo di +20 ogni mese" → ARIMA(0,1,0)

4.5.4 Previsione ARIMA per Marzo:

$$\text{Residuo Marzo} = \text{Residuo Feb} + \text{trend} = 60 + 20 = +80 \text{ kg} \tag{17}$$

4.5.5 Previsione Finale:

$$1180 \text{ kg (base)} + 80 \text{ kg (ARIMA)} = 1260 \text{ kg} \tag{18}$$

5 Componente 4: Trend Breaks Detection

5.1 Cosa Sono i Trend Breaks

I Trend Breaks sono **punti di rottura strutturale** dove il comportamento dei dati cambia significativamente:

Prima del break: Produzione cresce di +50 kg/mese
Dopo il break: Produzione cresce di +200 kg/mese

5.2 Esempi Reali di Trend Breaks

5.2.1 Cambiamenti Operativi:

- **Nuovi macchinari** → capacità produttiva raddoppia
- **Nuovo turno notturno** → +40% di produzione
- **Automazione** → efficienza in salto

5.2.2 Cambiamenti di Mercato:

- **Nuovo cliente importante** → volume costantemente più alto
- **Perdita cliente chiave** → caduta permanente
- **Nuovo prodotto** → mix produttivo diverso

5.3 Algoritmo di Rilevamento

```
detect_trend_breaks <- function(ts_data, min_size = 6) {
    values <- ts_data$qta_prodotta_tot
    n <- length(values)

    breaks <- c()
    for(i in seq(min_size, n - min_size)) {
        before <- lm(values[max(1, i-min_size):i] ~
                     I(max(1, i-min_size):i))$coefficients[2]
        after <- lm(values[i:(i+min_size)] ~
                    I(i:(i+min_size)))$coefficients[2]

        if(!is.na(before) && !is.na(after) &&
           abs(before - after) > sd(diff(values), na.rm = TRUE)) {
            breaks <- c(breaks, i)
        }
    }
    return(unique(breaks))
}
```

Listing 8: Algoritmo trend breaks

5.4 Come Funziona

5.4.1 Step 1: Finestre Mobili

L'algoritmo scorre lungo tutta la serie temporale con una **finestra mobile**:

Dati: [Gen, Feb, Mar, Apr, Mag, Giu, Lug, Ago, Set, Ott, Nov, Dic]

Posizione i=6 (Giugno):
PRIMA: [Gen, Feb, Mar, Apr, Mag, Giu] ← finestra di 6 mesi
DOPO: [Giu, Lug, Ago, Set, Ott, Nov] ← finestra di 6 mesi

5.4.2 Step 2: Calcolo Trend Prima e Dopo

Per ogni posizione, calcola:

- **Trend PRIMA**: regressione lineare sui 6 mesi precedenti
- **Trend DOPO**: regressione lineare sui 6 mesi successivi

5.4.3 Step 3: Test di Significatività

Condizione per Break: La differenza tra i due trend deve essere **maggiore della volatilità normale** dei dati.

$$|\beta_{after} - \beta_{before}| > \sigma(\Delta X) \tag{19}$$

5.5 Esempio Concreto

5.5.1 Dati di Produzione:

Gen: 800, Feb: 850, Mar: 900, Apr: 950, Mag: 1000, Giu: 1050
Lug: 1200, Ago: 1400, Set: 1600, Ott: 1800, Nov: 2000, Dic: 2200

5.5.2 Analisi a Giugno (i=6):

Trend PRIMA (Gen→Giu):

$$y = 750 + 50 \times \text{mese} \quad \Rightarrow \quad \beta_{before} = +50 \text{ kg/mese} \tag{20}$$

Trend DOPO (Giu→Dic):

$$y = 800 + 200 \times \text{mese} \quad \Rightarrow \quad \beta_{after} = +200 \text{ kg/mese} \tag{21}$$

5.5.3 Test di Significatività:

$$\text{Differenza} = |200 - 50| = 150 \text{ kg/mese} \tag{22}$$

$$\text{Volatilità normale} = \sigma(\Delta X) \approx 30 \text{ kg/mese} \tag{23}$$

$$150 > 30 \quad \Rightarrow \quad \text{BREAK RILEVATO a Giugno!} \tag{24}$$

6 Componente 5: Ensemble Pesato

6.1 Concetto di Ensemble

Un ensemble combina **multiple previsioni** invece di affidarsi a un singolo modello:

$$\hat{Y}_{finale} = w_1\hat{Y}_1 + w_2\hat{Y}_2 + w_3\hat{Y}_3 + \cdots \tag{25}$$

Dove $w_1, w_2, w_3, \ldots$ sono i **pesi** che determinano quanto fidarsi di ogni componente.

6.2 Sistema di Pesatura

6.2.1 Validazione Rolling Window

Il modello testa la sua performance storica usando una **validazione rolling window**:

```
for(i in 12:(nrow(train_data)-1)) {
    train_subset <- train_data[1:i, ]
    actual <- train_data$qta_prodotta_tot[i+1]

    # Previsione semplice
    pred_naive <- train_subset$qta_prodotta_tot[nrow(train_subset)]
    pred_trend <- mean(tail(train_subset$qta_prodotta_tot, 3), na.rm =
        TRUE)

    error_naive <- abs(actual - pred_naive)
    error_trend <- abs(actual - pred_trend)

    errors <- c(errors, min(error_naive, error_trend))
}
```

Listing 9: Validazione rolling window

Processo:

1. Usa i primi 12 mesi per "addestrare"
2. Prevedi il 13° mese
3. Confronta con il valore reale
4. Ripeti spostando la finestra di 1 mese

6.2.2 Calcolo Performance Metric

```
performance_metric <- 1 / (1 + mean(errors, na.rm = TRUE))
```

Listing 10: Performance metric

Caratteristiche della formula:

- Se errore medio = 0 → performance_metric = 1.0 (perfetto)
- Se errore medio = 100 → performance_metric = 0.01 (pessimo)
- Sempre compresa tra 0 e 1

6.3 Combinazione dei Componenti

6.3.1 Ensemble Base

```
pred_base <- pred_commesse + seasonal_future + pred_residui
```

Listing 11: Combinazione base

Ogni componente ha peso = 1.0:

- Componente commesse: peso 1.0
- Componente stagionale: peso 1.0
- Componente ARIMA: peso 1.0

6.3.2 Aggiustamento Trend

```
trend_recente <- mean(diff(tail(train_data$qta_prodotta_tot, 6)), na.rm
    = TRUE)
if(!is.na(trend_recente) && abs(trend_recente) > 0) {
    trend_adjustment <- trend_recente * seq(1, h) * 0.5
    pred_base <- pred_base + trend_adjustment
}
```

Listing 12: Aggiustamento trend

Damping Factor = 0.5:

- Mese 1: trend × 1 × 0.5 = 50% del trend
- Mese 2: trend × 2 × 0.5 = 100% del trend
- Mese 3: trend × 3 × 0.5 = 150% del trend

6.4 Vincoli Realistici

```
min_val <- min(train_data$qta_prodotta_tot, na.rm = TRUE) * 0.3
max_val <- max(train_data$qta_prodotta_tot, na.rm = TRUE) * 1.5
pred_final <- pmax(min_val, pmin(max_val, pred_base))
```

Listing 13: Vincoli di sicurezza

Sistema di Sicurezza:

- **Minimo**: Non può prevedere meno del 30% del minimo storico
- **Massimo**: Non può prevedere più del 150% del massimo storico

6.5 Intervalli di Confidenza

```
historical_errors <- diff(train_data$qta_prodotta_tot)
volatility <- sd(historical_errors, na.rm = TRUE) * sqrt(1:h)

# Intervalli al 80% e 95%
lower_80 <- pred_final - 1.28 * volatility
upper_80 <- pred_final + 1.28 * volatility
lower_95 <- pred_final - 1.96 * volatility
upper_95 <- pred_final + 1.96 * volatility
```

Listing 14: Calcolo intervalli di confidenza

Caratteristiche:

- La volatilità **cresce** con l'orizzonte temporale ($\sqrt{h}$)
- Più lontano nel futuro = più incertezza
- Coefficienti standard: 1.28 (80%), 1.96 (95%)

7 Architettura Completa del Modello

7.1 Flusso di Esecuzione

1. **Caricamento Dati**: Pulizia e preparazione dataset
2. **Split Train/Test**: Separazione per validazione
3. **Componente 1**: Modellazione relazione commesse
4. **Componente 2**: Estrazione stagionalità adattiva
5. **Componente 3**: ARIMA sui residui
6. **Componente 4**: Rilevamento trend breaks
7. **Componente 5**: Ensemble pesato e vincoli
8. **Output**: Previsioni con intervalli di confidenza

7.2 Vantaggi dell'Approccio Ibrido

7.2.1 Robustezza

Se un componente fallisce, gli altri compensano:

Componente commesse: 1000 kg (normale)
Componente stagionale: 0 kg (fallito)
Componente ARIMA: +100 kg (compensa)
→ Risultato: ancora ragionevole

7.2.2 Adattività

I pesi si aggiornano automaticamente con nuovi dati.

7.2.3 Interpretabilità

Puoi vedere il contributo di ogni componente:

Previsione finale: 1,200 kg
- Commesse: +1,000 kg
- Stagionale: +80 kg
- ARIMA: +50 kg
- Trend: +70 kg

7.2.4 Scalabilità

Il modello può essere esteso con nuovi componenti senza modificare l'architettura base.

8 Conclusioni

Il modello ibrido presentato rappresenta un approccio innovativo che combina:

- **Logica di business** (relazione commesse-produzione)
- **Pattern temporali** (stagionalità e trend)
- **Dinamiche stocastiche** (effetti di memoria e shock)
- **Robustezza operativa** (vincoli e fallback)
- **Adattabilità** (aggiornamento automatico dei parametri)

8.1 Confronto con Modelli Standard

8.2 Risultati Attesi

Il modello è progettato per fornire:

1. **Accuratezza superiore**: Grazie alla combinazione di multiple fonti di informazione
2. **Intervalli realistici**: Confidenza che cresce con l'orizzonte temporale

Caratteristica	ARIMA	ETS	Modello Ibrido
Rel. Commesse	×	×	✓
Stagionalità	✓	✓	✓
Trend Breaks	×	×	✓
Robustezza	×	×	✓
Interpretabilità	×	×	✓
Vincoli Business	×	×	✓

Tabella 1: Confronto caratteristiche modelli

3. **Insight operativi**: Decomposizione dei fattori che influenzano la produzione
4. **Robustezza**: Resilienza a dati mancanti e outlier

9 Appendici

9.1 Appendice A: Codice Completo delle Funzioni

9.1.1 Funzione Media Mobile Ponderata

```
weighted_moving_average <- function(x, weights = c(0.5, 0.3, 0.2)) {
    n <- length(x)
    if(n < length(weights)) return(rep(mean(x, na.rm = TRUE), n))

    result <- numeric(n)
    for(i in seq_along(weights)) {
        result[1:(length(weights)-1)] <- mean(x[1:length(weights)], na.
            rm = TRUE)
    }

    for(i in length(weights):n) {
        if(i == length(weights)) {
            result[i] <- sum(x[(i-length(weights)+1):i] * rev(weights),
                na.rm = TRUE)
        } else {
            result[i] <- sum(x[(i-length(weights)+1):i] * rev(weights),
                na.rm = TRUE)
        }
    }
    return(result)
}
```

Listing 15: Media mobile ponderata

9.1.2 Funzione Componente Stagionale Adattiva

```
adaptive_seasonal <- function(ts_data, period = 12) {
    if(nrow(ts_data) < period * 2) return(rep(0, nrow(ts_data)))

    values <- ts_data$qta_prodotta_tot
    seasonal_component <- numeric(length(values))

    # Calcola stagionalita usando decomposizione robusta
    if(length(values) >= period * 2) {
        ts_obj <- ts(values, frequency = period)
        decomp <- tryCatch({
            stl(ts_obj, s.window = "periodic", robust = TRUE)
        }, error = function(e) {
            # Fallback: stagionalita semplice
            seasonal_means <- tapply(values, rep(1:period, length.out =
                length(values)),
                                     mean, na.rm = TRUE)
            list(time.series = cbind(seasonal = rep(seasonal_means,
                                                    length.out = length(
                                                        values))))
        })

        if(is.list(decomp) && "time.series" %in% names(decomp)) {
            seasonal_component <- as.numeric(decomp$time.series[, "
                seasonal"])
        } else {
            seasonal_component <- rep(0, length(values))
        }
    }

    return(seasonal_component)
}
```

Listing 16: Stagionalità adattiva completa

9.2 Appendice B: Parametri di Configurazione

9.2.1 Parametri Principali

Parametro	Valore	Descrizione
`min_size`	6	Finestra minima per trend breaks
`period`	12	Frequenza stagionale (mensile)
`damping`	0.5	Fattore di attenuazione trend
`min_bound`	0.3	Limite inferiore (30% del minimo)
`max_bound`	1.5	Limite superiore (150% del massimo)
`conf_80`	1.28	Coefficiente intervallo 80%
`conf_95`	1.96	Coefficiente intervallo 95%

Tabella 2: Parametri di configurazione del modello

9.2.2 Criteri di Qualità

- **R^2 minimo**: 0.3 per componente commesse
- **Osservazioni minime**: 24 mesi per training affidabile
- **Soglia break**: $|\beta_{after} - \beta_{before}| > \sigma(\Delta X)$
- **Performance threshold**: 0.1 per considerare il modello valido

9.3 Appendice C: Estensioni Possibili

9.3.1 Miglioramenti Algoritmici

1. **Change Point Detection avanzato**: Algoritmi CUSUM, PELT
2. **Modelli regime-switching**: Diversi modelli per diversi periodi
3. **Machine Learning**: Random Forest, XGBoost per ensemble
4. **Bayesian updating**: Aggiornamento continuo dei parametri

9.3.2 Fonti Dati Aggiuntive

1. **Indicatori esterni**: Indici economici, stagionalità settore
2. **Dati operativi**: Ore macchina, tasso di utilizzo, manutenzioni
3. **Dati commerciali**: Pipeline vendite, promozioni, new entries
4. **Feedback real-time**: Aggiornamento con dati giornalieri

9.3.3 Output Avanzati

1. **Scenario analysis**: Previsioni multiple con assunzioni diverse
2. **Sensitivity analysis**: Impatto di variazioni parametri
3. **Risk metrics**: VaR, Expected Shortfall per la produzione
4. **Business intelligence**: Dashboard interattivi, alert automatici

10 Implementazione Pratica

10.1 Requisiti Sistema

10.1.1 Software

- **R**: Versione ≥ 4.0
- **Pacchetti**: fpp3, forecast, dplyr, lubridate, readxl

10.1.2 Formato Dati Input

- **File**: Excel (.xlsx) o CSV
- **Colonne richieste**: data_mese, qta_prodotta, commessa
- **Frequenza**: Mensile (minimo 24 osservazioni)
- **Qualità**: Massimo 5% valori mancanti

10.2 Workflow di Esecuzione

1. **Preparazione**:
   - Verifica qualità dati
   - Backup dataset originale
   - Definizione data cutoff train/test
2. **Esecuzione**:
   - Lancio script automatico
   - Monitoring log di esecuzione
   - Verifica output intermedi
3. **Validazione**:
   - Controllo coerenza previsioni
   - Analisi residui e diagnostiche
   - Confronto con benchmark
4. **Deploy**:
   - Salvataggio modello finale
   - Esportazione risultati
   - Schedulazione aggiornamenti

10.3 Troubleshooting

10.3.1 Errori Comuni

10.3.2 Performance Tuning

- **Speed**: Usare `stepwise=TRUE` per dataset grandi
- **Memory**: Processare in batch per serie molto lunghe
- **Accuracy**: Aumentare `approximation=FALSE` per precisione
- **Stability**: Validazione incrociata per parametri critici

Errore	Causa	Soluzione
STL fallisce	Troppi NA	Imputazione o periodo più lungo
ARIMA instabile	Serie non stazionaria	Differenziazione aggiuntiva
R^2 basso commesse	Relazione debole	Variabili aggiuntive o trasformazioni
Break detection rumoroso	Volatilità alta	Aumentare min_size o smoothing
Previsioni estreme	Outlier recenti	Pulizia dati o vincoli più stretti

Tabella 3: Errori comuni e soluzioni

11 Bibliografia e Riferimenti

11.1 Riferimenti Teorici

1. Hyndman, R.J., & Athanasopoulos, G. (2021). *Forecasting: principles and practice*, 3rd edition, OTexts.
2. Box, G.E.P., Jenkins, G.M., & Reinsel, G.C. (2015). *Time Series Analysis: Forecasting and Control*, 5th edition.
3. Cleveland, R.B., Cleveland, W.S., McRae, J.E., & Terpenning, I. (1990). STL: A seasonal-trend decomposition procedure based on loess.
4. Bai, J., & Perron, P. (2003). Computation and analysis of multiple structural change models. *Journal of Applied Econometrics*.

11.2 Riferimenti Implementativi

1. R Core Team (2023). R: A language and environment for statistical computing.
2. Hyndman, R.J., et al. (2023). `forecast`: Forecasting functions for time series and linear models.
3. Wickham, H., et al. (2023). `dplyr`: A grammar of data manipulation.
4. O'Hara-Wild, M., et al. (2023). `fpp3`: Data for "Forecasting: Principles and Practice" (3rd Edition).

11.3 Documentazione Online

- `https://otexts.com/fpp3/` - Forecasting: Principles and Practice
- `https://cran.r-project.org/web/packages/forecast/` - Pacchetto forecast
- `https://robjhyndman.com/hyndsight/` - Blog Rob Hyndman
- `https://www.rdocumentation.org/` - Documentazione R