Relazione Progetto Data Science Lab

Gabriele Carrara, 814720; Davide Porcellini, 816586; Simone Tufano, 816984

09 luglio 2020

Abstract

Il seguente progetto si propone di sviluppare due differenti approcci (uno di tipo discreto e uno di carattere Time Series) per cercare di stimare la curva di offerta di elettricità sul mercato italiano del giorno dopo. In particolare è stato preso in considerazione l' orario delle 15 dal giorno 01/01/2018 al 31/12/2019 per costruire i modelli che prevedano le curve dal giorno 01/01/2020 al 29/02/2020 (considerata la particolare situazione italiana si è deciso di fermarsi a questa data).

Parole chiave

Elettricità, Clustering, R, Time Series.

Caricamento dei dati e filtraggio iniziale

Dopo aver scaricato i file, abbiamo utilizzato il file make_one_dataset.R fornito dal professore per creare il dataframe oggetto di analisi. Per esigenze computazionali, si è deciso di conservare solo le seguenti variabili: "PURPOSE_CD", "UNIT_REFERENCE_NO", "INTERVAL_NO", "BID_OFFER_DATE_DT","QUANTITY_NO", "ENERGY_PRICE_NO" e di limitare lo studio alle sole ore 15 di ogni giorno. Il dataset iniziale è composto da 1325521 osservazioni e 6 variabili.

```
zipfiles <- dir(pattern = ".zip")
files <- zipfiles
substr(files, 29, 31) <- "xml"

for (i in 1:length(files)) unzip(zipfiles[i], files[i])

# ---- build data.frame from XML
library(xml2)
msd <- NULL # will contain full dataframe

fnames <- dir(pattern = "*.xml")

for (fname in fnames){
  cat("File", fname, date(), "\n")
  mb <- read_xml(fname)

  # extract variable names and types
  schema <- as_list(xml_child(mb, 1))[[1]][[1]][[1]][[1]][[1]][[1]]
```

```
  var_keep  <- c("PURPOSE_CD", "UNIT_REFERENCE_NO", "INTERVAL_NO", "BID_OFFER_DATE_DT",
                 "QUANTITY_NO", "ENERGY_PRICE_NO")
  var_names <-sapply(schema, function(x) attr(x, "name"))
  var_types <- sapply(schema, function(x) attr(x, "type"))

  nrecords <- length(xml_children(mb)) - 1
  df <- data.frame(PURPOSE_CD = character(nrecords),
                   UNIT_REFERENCE_NO = character(nrecords),
                   INTERVAL_NO = integer(nrecords),
                   BID_OFFER_DATE_DT = integer(nrecords),
                   QUANTITY_NO = numeric(nrecords),
                   ENERGY_PRICE_NO = numeric(nrecords),
                   stringsAsFactors = FALSE
  )

  for (i in 1:length(var_keep)) {
    cat("Working on ", var_keep[i])
    var_content <- xml_find_all(mb, paste0("//", var_keep[i]))
    nrectmp <- length(var_content)
    cat("  n =", nrectmp, "\n")
    df[[i]] <- as(xml_text(var_content), typeof(df[[i]]))
  }
  msd <- if (fname == fnames[1]) df else rbind(msd, df)
}

save(msd, file = "MSDOffertePubbliche.Rdata")
```

```
load(file = "MSDOffertePubbliche.Rdata")
df_off_15 <- msd %>%
  filter(PURPOSE_CD == 'OFF') %>%
  filter(INTERVAL_NO == 15)
save(df_off_15, file= "OffertePubblicheFiltrate.Rdata")
```

APPROCCIO CLUSTER

Introduzione

Il primo approccio consiste nel clusterizzare curve simili al fine di stimare l'appartenenza ad un cluster per i giorni ignoti, così da prevedere la curva tramite un andamento che rappresenti il gruppo d'appartenenza.

Creazione della curva delle offerte

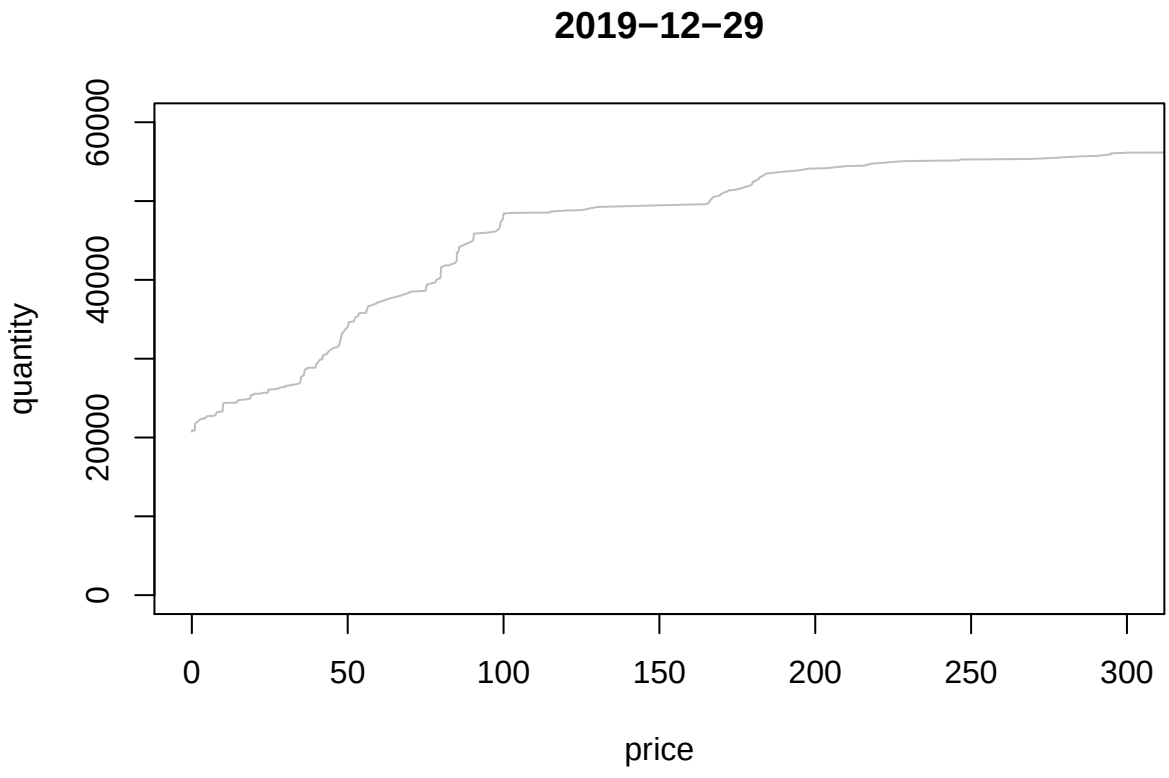
Viene creato un dataframe composto dalle quantità cumulate dell'offerta di elettricità giornaliera per le ore 15.

```
load(file = "OffertePubblicheFiltrate.Rdata")
msd<- df_off_15
df<- list()

for (day in unique(msd$BID_OFFER_DATE_DT))
```

```
{
  temp<- msd%>%filter(BID_OFFER_DATE_DT == day) %>%
    arrange(ENERGY_PRICE_NO)
  temp$sum<- cumsum(temp$QUANTITY_NO)
  temp$ENERGY_PRICE_NO<-as.factor(temp$ENERGY_PRICE_NO)
    temp<-temp %>%
    group_by(ENERGY_PRICE_NO)%>%
    summarise(sum =max(sum))
  temp$ENERGY_PRICE_NO<-as.numeric(as.character(temp$ENERGY_PRICE_NO))
  temp$Date<- day
  df<- append(df, list(temp),0)

}
```

Di seguito un esempio:

```
plot(df[[3]]$ENERGY_PRICE_NO,df[[3]]$sum, col="grey", type = 'l',
main= ymd(df[[3]][1, 'Date']), xlab = 'price', ylab = 'quantity',
xlim = c(0,300), ylim=c(0,60000))
```



Sviluppo dei modelli per approssimare le curve

Dato che le curve cumulate presentano un andamento a "scalini", si decide di utilizzare un modello di regressione polinomiale di terzo grado e delle splines con knots scelti a priori relativamente pari a 10, 100, 200 e 300 per cercare di levigare la curva.

```
modelli<-lapply(df, function(x) lm( sum ~ ENERGY_PRICE_NO +
              I(ENERGY_PRICE_NO^2)+ I(ENERGY_PRICE_NO^3), x))

splines<-lapply(df, function(x) lm(sum~bs(ENERGY_PRICE_NO,knots=c(10,100,200,300)),x))

r_splines<- as.data.frame(unlist((lapply(splines, function(x) summary(x)$r.squared))))
summary(r_splines)
```

```
##  unlist((lapply(splines, function(x) summary(x)$r.squared)))
##  Min.   :0.9257
##  1st Qu.:0.9786
##  Median :0.9836
##  Mean   :0.9822
##  3rd Qu.:0.9878
##  Max.   :0.9959
```

```
r_modelli<- as.data.frame(unlist((lapply(modelli, function(x) summary(x)$r.squared))))
summary(r_modelli)
```

```
##  unlist((lapply(modelli, function(x) summary(x)$r.squared)))
##  Min.   :0.7766
##  1st Qu.:0.8717
##  Median :0.9130
##  Mean   :0.9044
##  3rd Qu.:0.9404
##  Max.   :0.9920
```

Come emerge dal summary, i modelli polinomiali presentano generalmente un R^2 minore (0.913), quindi si decide di proseguire con le splines (0.982).

A seguito di verifiche la spline non risulta essere monotona crescente (come invece una cumulata dovrebbe essere), per cui si opta per imporre questa condizione:

```
gam<- lapply(df, function(x) gam(sum ~ s(ENERGY_PRICE_NO),data= x,
                                 family = gaussian(link='identity')))
```

Per ricostruire le curve dopo l'approssimazione, si utilizzano i valori delle quantità previste dalle splines in 26 punti, i quali vanno da 0 a 250 ogni 10 unità di prezzo.

```
curves<- data.frame()

for (i in 1:length(gam))
{
  temp<- data.frame(Date= NaN)
  temp$Date<- df[[i]][[1, "Date"]]
  prediction<- predict(gam[[i]], data.frame(ENERGY_PRICE_NO=seq(0, 250, by=10)))
  pred<- t(data.frame(prediction))
  temp<-cbind(temp, pred)
  curves<- rbind(curves, temp)
}

names<- paste(rep('pred', 26), seq(0, 250, by=10), sep = '_')
colnames(curves)[2:27]<- c(names)
```

Le date e le previsioni vengono inserite nel dataframe curves.

Analisi dei gruppi attraverso il metodo delle k-medie

Per identificare delle curve simili, si procede con un'analisi dei gruppi attraverso il metodo di clustering k-medie. Vengono considerate le partizioni per K = 3,4,5,6,7,8 gruppi utilizzando come criterio di scelta gli indici R^2 e $Pseudo - F$.

```
list_r_squared<- list()
list_pseudo_f<- list()

for (i in c(3,4,5,6,7,8)) {
  set.seed(123)
  temp <- assign(paste0('kmeans_',i),kmeans(curves[,2:27], i))
  r_squared <- (temp$betweenss) / (temp$totss);
  list_r_squared <- append(list_r_squared, r_squared)
  pseudo_f <- (temp$betweenss/(i-1)) / (temp$tot.withinss/(730-i))
  list_pseudo_f <- append(list_pseudo_f, pseudo_f)
}
```

```
unlist(list_pseudo_f)
```

```
## [1] 1327.5602 1191.9652 1020.7377  884.0993  761.7781  899.5549
```

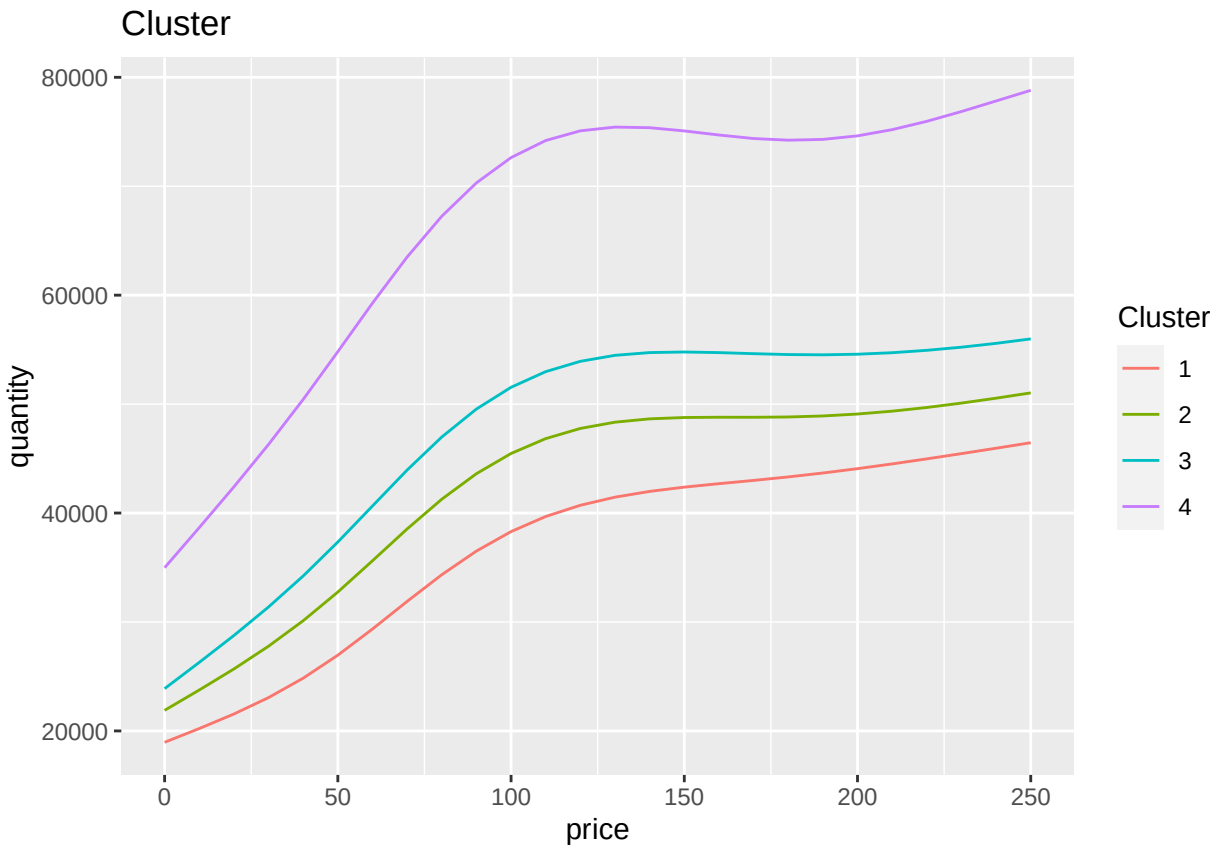
```
unlist(list_r_squared)
```

```
## [1] 0.7850461 0.8312372 0.8492081 0.8592671 0.8634218 0.8971347
```

La soluzione vincente, per le metriche appena citate risulta essere la divisione per 4 gruppi con valori delle due metriche prese in considerazione di (0.831, 1191.965). Pur non essendo i valori più elevati, si decide di scegliere questa disposizione perchè presenta buone performance su entrambe le metriche e favorisce l'interpretabilità dei cluster.

Di seguito la rappresentazione dei cluster:

```
centroidi<- as.data.frame(t(kmeans_4$centers))
centroidi$X<- seq(0, 250, by=10)

centroidi_tidy <- centroidi%>%
  gather(key = "Cluster", value = "Y", `1`:`4`)

ggplot(centroidi_tidy, aes(x=X, y= Y, colour= Cluster))+
  geom_line()+
  labs(x="price",
       y="quantity",
       title = "Cluster")
```



Si nota una curva molto distante dalle altre con valori sull'asse delle ascisse più elevati; dopo una breve analisi esplorativa si vede che a questo cluster appartengono i giorni dei primi tre mesi del 2018 (ciò è giustificato da un aumento dei prezzi iniziato a fine 2017 e terminato appunto a marzo 2018).

Al cluster 1, invece, appartengono principalmente i giorni del weekend. È ragionevole pensare che questo sia dovuto ad una limitata richiesta di elettricità nei giorni non lavorativi.

Classificazione dei nuovi giorni in base ai cluster ottenuti

Per prevedere l'appartenenza ad un cluster, si creano delle nuove variabili:

1. giorno, inerente il giorno della settimana;
2. month, inerente il mese;
3. cluster_prec, inerente il cluster di appartenenza del giorno precedente;
4. cluster_week_prec, inerente il cluster di appartenenza dello stesso giorno nella settimana precedente.

```
curves<- cbind(curves, kmeans_4$cluster)
curves$Date<- ymd(curves$Date)
curves$giorno <- weekdays(curves$Date)
curves$month<- month(curves$Date)

colnames(curves)[28]<- "Cluster"
```

```
cluster_prec<- curves$Cluster[2:nrow(curves)]
curves_con_prec <-cbind(curves[1:729,], cluster_prec)

cluster_week_prec<- curves$Cluster[8:nrow(curves)]
curves_con_prec <-cbind(curves_con_prec[1:723,], cluster_week_prec)

curves_con_prec$Cluster<- as.factor(curves_con_prec$Cluster)
curves_con_prec$cluster_prec<- as.factor(curves_con_prec$cluster_prec)
curves_con_prec$cluster_week_prec<- as.factor(curves_con_prec$cluster_week_prec)
curves_con_prec$month<- as.factor(curves_con_prec$month)
```

Dopo aver creato il dataset per la classificazione, si procede con la divisione in train e test e all'applicazione dei seguenti modelli: Random Forest, Decision Tree e Regressione Logistica.

```
set.seed(123)
sample = sample.split(curves_con_prec$Cluster, SplitRatio = .75)
train = subset(curves_con_prec, sample == TRUE)
test  = subset(curves_con_prec, sample == FALSE)
```

Random Forest

```
rf<- randomForest(Cluster~ giorno + month + cluster_prec + cluster_week_prec , train)
pred = predict(rf, newdata=test[-28])
cm = table(test[,28], pred)
cm
```

```
##    pred
##      1  2  3  4
##   1 27 12  1  0
##   2  4 65  8  0
##   3  1 20 28  1
##   4  0  0  2 12
```

```
sum(diag(cm))/nrow(test)
```

```
## [1] 0.7292818
```

Decision Tree

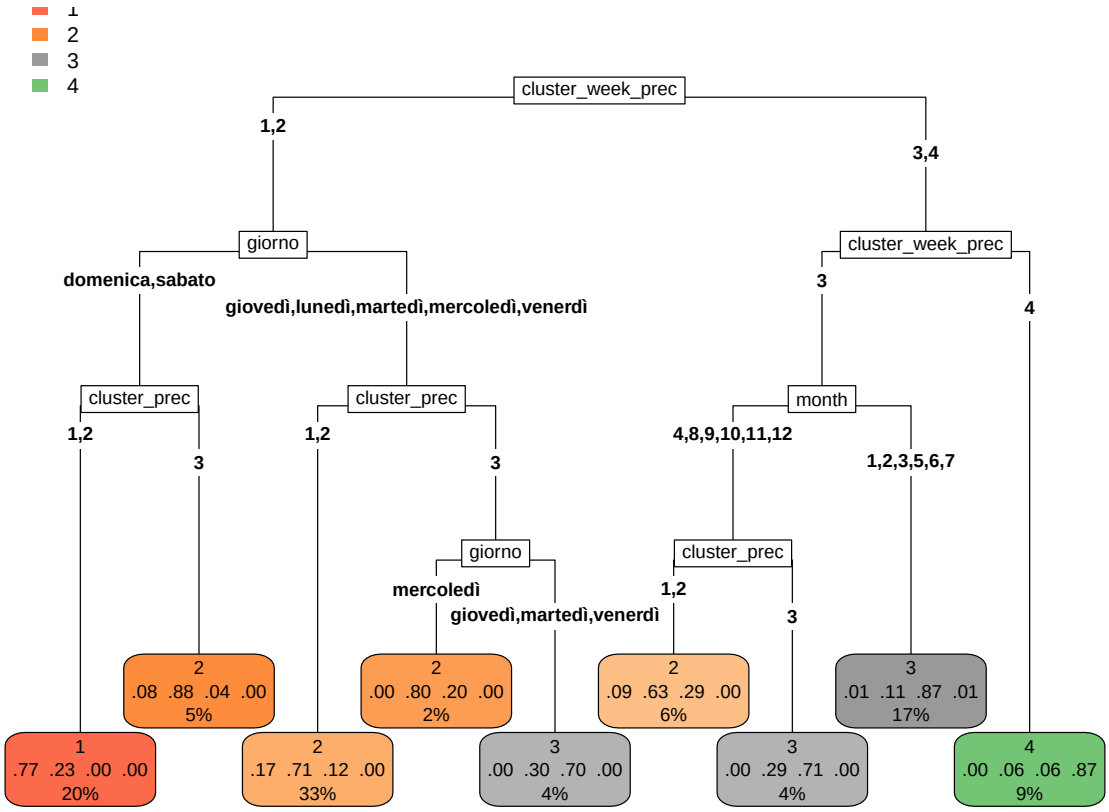
```
tree<- rpart(Cluster~ giorno + month + cluster_prec + cluster_week_prec , train)
pred = predict(tree, newdata=test[-28], type = 'class')
cm = table(test[,28], pred)
cm
```

```
##    pred
##      1  2  3  4
##   1 31  8  1  0
##   2  4 63 10  0
##   3  0 14 35  1
##   4  0  0  2 12
```

```
sum(diag(cm))/nrow(test)
```

```
## [1] 0.7790055
```

```
rpart.plot(tree, type = 5)
```



L'albero mostra la caratterizzazione dei cluster e conferma le interpretazioni precedenti.

Regressione Logistica

```
logistic<- nnet::multinom(Cluster~ giorno + month + cluster_prec +
                            cluster_week_prec , train)
```

```
## # weights:  100 (72 variable)
## initial  value 751.371544
## iter  10 value 311.448914
## iter  20 value 275.551085
## iter  30 value 265.974704
## iter  40 value 259.764823
## iter  50 value 258.722639
## iter  60 value 258.519535
## iter  70 value 258.505155
```

```
## iter  80 value 258.504792
## iter  80 value 258.504790
## iter  80 value 258.504790
## final  value 258.504790
## converged
```

```
pred = predict(logistic, newdata=test[-28], type = 'class')
cm = table(test[,28], pred)
cm
```

```
##    pred
##      1  2  3  4
##   1 30  9  1  0
##   2  4 62 10  1
##   3  0 11 37  2
##   4  0  0  0 14
```

```
sum(diag(cm))/nrow(test)
```

```
## [1] 0.7900552
```

I risultati dell'accuracy sono molto simili (intorno allo 0,75) e i cluster che vengono confusi sono generalmente il 2 e il 3 che si è visto avere le curve più simili. Si sceglie di conseguenza il Decision Tree per l'interpretabilità.

Previsione sui dati del 2020

Si organizza il dataset contenente i nuovi giorni (Gennaio-Febbraio 2020), creato in modo analogo a quanto fatto in precedenza, per poter effettuare la previsione di quale sia il cluster a cui appartengono.

```
load("to_predict.RData")

df<- list()
for (day in unique(msd$BID_OFFER_DATE_DT))
{
  temp<- msd%>%filter(BID_OFFER_DATE_DT == day) %>%
    arrange(ENERGY_PRICE_NO)
  temp$sum<- cumsum(temp$QUANTITY_NO)
  temp$ENERGY_PRICE_NO<-as.factor(temp$ENERGY_PRICE_NO)
  temp<-temp %>%
    group_by(ENERGY_PRICE_NO)%>%
    summarise(sum =max(sum))
  temp$ENERGY_PRICE_NO<-as.numeric(as.character(temp$ENERGY_PRICE_NO))
  temp$Date<- day
  df<- append(df, list(temp),0)

}

mon_splines<- lapply(df, function(x) lm(sum ~ mSpline(ENERGY_PRICE_NO, degree = 3,
                                                      Boundary.knots = c(0,300)), x))

r_mon_splines<- as.data.frame(unlist((lapply(mon_splines,
                                             function(x) summary(x)$r.squared))))
summary(r_mon_splines)
```

```
##  unlist((lapply(mon_splines, function(x) summary(x)$r.squared)))
##  Min.   :0.7631
##  1st Qu.:0.7966
##  Median :0.8363
##  Mean   :0.8391
##  3rd Qu.:0.8748
##  Max.   :0.9518
```

```
curves<- data.frame()
for (i in 1:length(mon_splines))
{
  temp<- data.frame(Date= NaN)
  temp$Date<- df[[i]][[1, "Date"]]
  prediction<- predict(mon_splines[[i]], data.frame(ENERGY_PRICE_NO=seq(0, 250, by=10)))
  pred<- t(data.frame(prediction))
  temp<-cbind(temp, pred)
  curves<- rbind(curves, temp)
}

names<- paste(rep('pred', 26), seq(0, 250, by=10), sep = '_')
colnames(curves)[2:27]<- c(names)

new_curves<- curves

#aggiungo valore cluster settimana precedente e giorno precedente
last_7<- curves_con_prec[1:7, 28]
pr <- predict_KMeans(new_curves[,2:27], kmeans_4$centers)
cluster_week_prec <- append(pr[8:60], last_7)
new_curves$cluster_week_prec<- cluster_week_prec

last<- curves_con_prec[1, 28]
cluster_prec <- append(pr[2:60], last)
new_curves$cluster_prec<- cluster_prec

#come sopra aggiungo giorno della settimana e mese
new_curves$Date<- ymd(new_curves$Date)
new_curves$giorno <- weekdays(new_curves$Date)
new_curves$month<- month(new_curves$Date)
```

Si utilizza il Decision Tree per assegnare ai nuovi punti il cluster di appartenenza previsto.

```
#trasformazione in factor e predizione cluster su dati 2020
new_curves$cluster_prec<- as.factor(new_curves$cluster_prec)
new_curves$cluster_week_prec<- as.factor(new_curves$cluster_week_prec)
new_curves$month<- as.factor(new_curves$month)
new_curves$predicted_cluster<- predict(tree, new_curves[,28:31], type = 'class')
```

Si aggiunge il cluster predetto al dataset iniziale.

```
new_days<- new_curves[,c(1,28:32)]
for (i in c(1:length(df)))
{
  df[[i]]$Date <- ymd(df[[i]]$Date)
```

```
  df[[i]]<- merge(df[[i]], new_days[, c(1,6)])
  }
```

Si stima la curva tipo del cluster come spline che passa attraverso i centroidi.

```
#modello per ogni cluster
cluster_model<- list()
for (i in c(1:4))
{
  cluster_model[[i]]<- gam(centroidi[,i] ~ s(X),data= centroidi,
                           family=gaussian(link='identity'))
}
```

In base all'appartenenza al cluster si utilizza il rispettivo modello per le stime della cumulata.

```
for (i in c(1:length(df)))
{
df[[i]]$prediction <- predict(cluster_model[[df[[i]][1, "predicted_cluster"]]],
                              newdata=data.frame(X=df[[i]]$ENERGY_PRICE_NO))
}
```

Si calcola l'MSE delle previsioni.

```
mse<-c()
for (i in c(1:length(df)))
{
mse[i]<- MSE(df[[i]]$prediction, df[[i]]$sum)
}
```

Di seguito MSE e RMSE medio per i primi due mesi del 2020:

```
mean(mse)
```

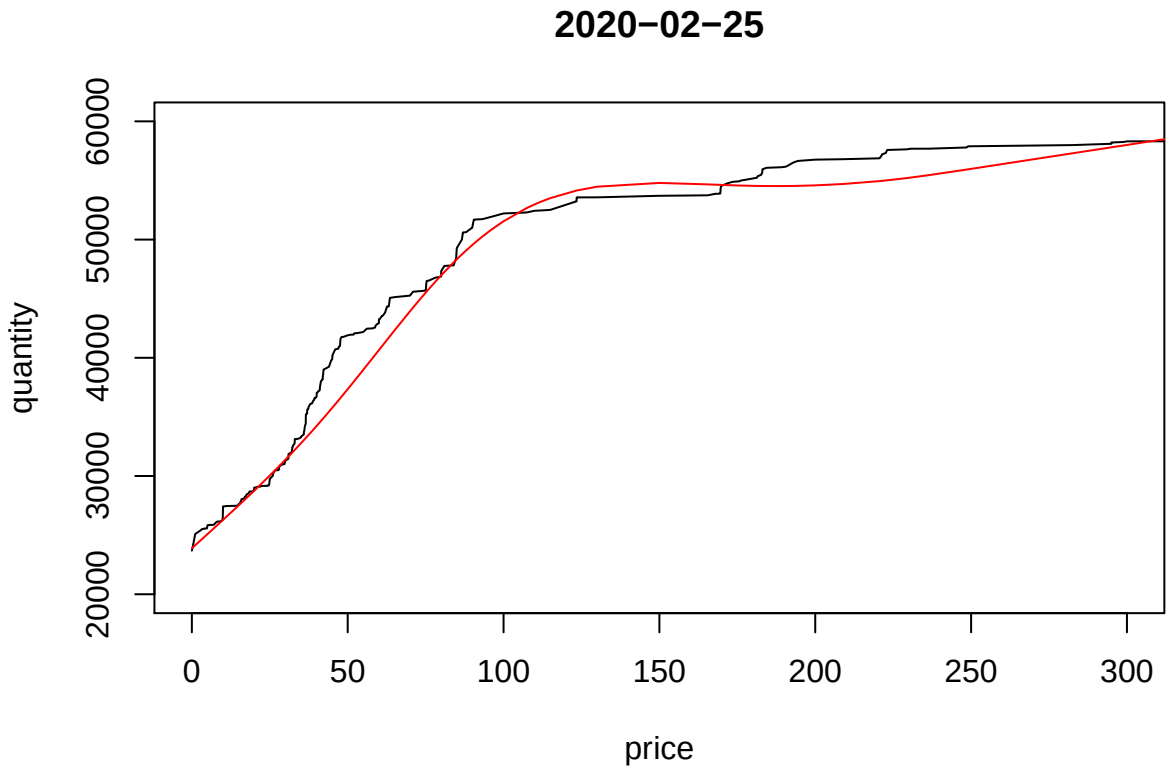
```
## [1] 32676025
```

```
sqrt(mean(mse))
```

```
## [1] 5716.295
```

Un esempio di previsione per il 25/02/2020:

```
plot(df[[5]]$ENERGY_PRICE_NO, df[[5]]$sum, type= 'l', xlim= c(0,300),
     ylim = c(20000, 60000), main = df[[5]][1,1], ylab = 'quantity', xlab= 'price')
lines(df[[5]]$ENERGY_PRICE_NO, df[[5]]$prediction, col= 'red')
```



Si calcolano gli stessi indici per l'intervallo di prezzo 0-300: questa è la zona più significativa della cumulata in quanto il punto di equilibrio si trova solitamente entro questo intervallo.

```
mse<-c()
df_filtered<-list()
for (i in c(1:length(df)))
{
df_filtered[[i]]<- df[[i]]%>%filter(ENERGY_PRICE_NO<300)
mse[i]<- MSE(df_filtered[[i]]$prediction, df_filtered[[i]]$sum)
}
```

```
mean(mse)
```

```
## [1] 17414137
```

```
sqrt(mean(mse))
```

```
## [1] 4173.025
```

APPROCCIO TIME SERIES

Introduzione

Il secondo approccio consiste nello stimare attraverso le spline le curve dei due anni così da ottenere la serie storica dei coefficienti. Questa viene utilizzata per fare la stima del giorno succesivo. Una volta fatto ciò si

costruisce una nuova serie storica che comprende i dati effettivi del giorno per la stima di quello seguente.

Script

Si carica e si preprocessa il dataset come fatto in precedenza.

```
load(file = "OffertePubblicheFiltrate.Rdata")
msd<- df_off_15
df<- list()

for (day in unique(msd$BID_OFFER_DATE_DT))
{
  temp<- msd%>%filter(BID_OFFER_DATE_DT == day) %>%
    arrange(ENERGY_PRICE_NO)
  temp$sum<- cumsum(temp$QUANTITY_NO)
  temp$ENERGY_PRICE_NO<-as.factor(temp$ENERGY_PRICE_NO)
    temp<-temp %>%
    group_by(ENERGY_PRICE_NO)%>%
    summarise(sum =max(sum))
  temp$ENERGY_PRICE_NO<-as.numeric(as.character(temp$ENERGY_PRICE_NO))
  temp$Date<- day
  df<- append(df, list(temp))

}
```

Analogo per i giorni da stimare.

```
load("to_predict.RData")

df_new<- list()
for (day in unique(msd$BID_OFFER_DATE_DT))
{
  temp<- msd%>%filter(BID_OFFER_DATE_DT == day) %>%
    arrange(ENERGY_PRICE_NO)
  temp$sum<- cumsum(temp$QUANTITY_NO)
  temp$ENERGY_PRICE_NO<-as.factor(temp$ENERGY_PRICE_NO)
  temp<-temp %>%
    group_by(ENERGY_PRICE_NO)%>%
    summarise(sum =max(sum))
  temp$ENERGY_PRICE_NO<-as.numeric(as.character(temp$ENERGY_PRICE_NO))
  temp$Date<- day
  df_new<- append(df_new, list(temp))
}
```

```
next_day<- list() #lista vuota per i risultati
for (j in c(1:length(df_new)))
{
splines<- lapply(df, function(x)
  lm(sum ~ bs(ENERGY_PRICE_NO,knots = c(10,100,200,300) ), x))
#stimiamo le spline per i giorni noti
  coef <- data.frame('intercept'= 1, 'coef_1'= 1, 'coef_2'= 1,
          'coef_3'= 1, 'coef_4'= 1, 'coef_5'= 1, 'coef_6'= 1, 'coef_7'= 1 )
```

```
for (i in c(1:length(splines))){
  temp<- splines[[i]]$coefficients
  coef<- rbind(coef, temp)
  }
coef<- coef[-1,] #mettiamo i coefficienti dentro un dataset

for (i in c(1:length(df))){
  coef[i, "Date"]<- df[[i]][1, "Date"] #cambiamo formato alla data
  }
coef$Date<- ymd(coef$Date)

coef_tidy <- coef%>%
  gather(key = "coefficients", value = "Value", `intercept`:`coef_7`)#dataset tidy

series<- list()
i<- 1
for (coeff in unique(coef_tidy$coefficients)){
  series[[i]]<- coef_tidy%>%
    filter(coefficients== coeff)%>%
    arrange(Date)
  i<- i+1
}#serie storica per ogni coefficiente

for (i in c(1:length(series))){
  name<- c('Date', series[[i]][1, 'coefficients'])
  series[[i]]<- series[[i]][, c('Date','Value')]
  colnames(series[[i]])<- name
}

arima<- lapply(series, function(x) auto.arima(x[j:(729+j),2]%>%
                                                ts(frequency = 365))) #modello arima

prev<-c()
for (i in c(1:length(arima))){
  prev<- append(prev, forecast::forecast(arima[[i]], h = 1)$mean)
}#stima coefficienti per il giorno dopo

next_day[[j]] <-  prev[1]+ bs(df_new[[j]]$ENERGY_PRICE_NO,
    knots = c(10,100,200,300))%*% prev[2:8] #riempiamo la lista con le previsioni

df<- append(df, df_new[j],0)
#aggiungiamo il giorno a quelli noti per la stima del successivo
}
```

Si uniscono previsioni e dati effettivi.

```
for (i in c(1:length(next_day)))
{next_day[[i]]<- cbind(next_day[[i]], df_new[[i]])
colnames(next_day[[i]])[1]<- 'prev'}
```

Calcolo dell'MSE su tutto l'intervallo.

```
mse<-c()
for (i in c(1:length(next_day)))
{
  mse[i]<- MSE(next_day[[i]]$prev, df_new[[i]]$sum)
}
```

```
mean(mse)
```

```
## [1] 8490232
```

```
sqrt(mean(mse))
```

```
## [1] 2913.8
```

Calcolo dell'MSE sull'intervallo 0-300.

```
mse<-c()
for (i in c(1:length(next_day)))
{next_day[[i]]<- next_day[[i]]%>%
  filter(ENERGY_PRICE_NO< 301)
  mse[i]<- MSE(next_day[[i]]$prev, next_day[[i]]$sum)
}
```

```
mean(mse)
```

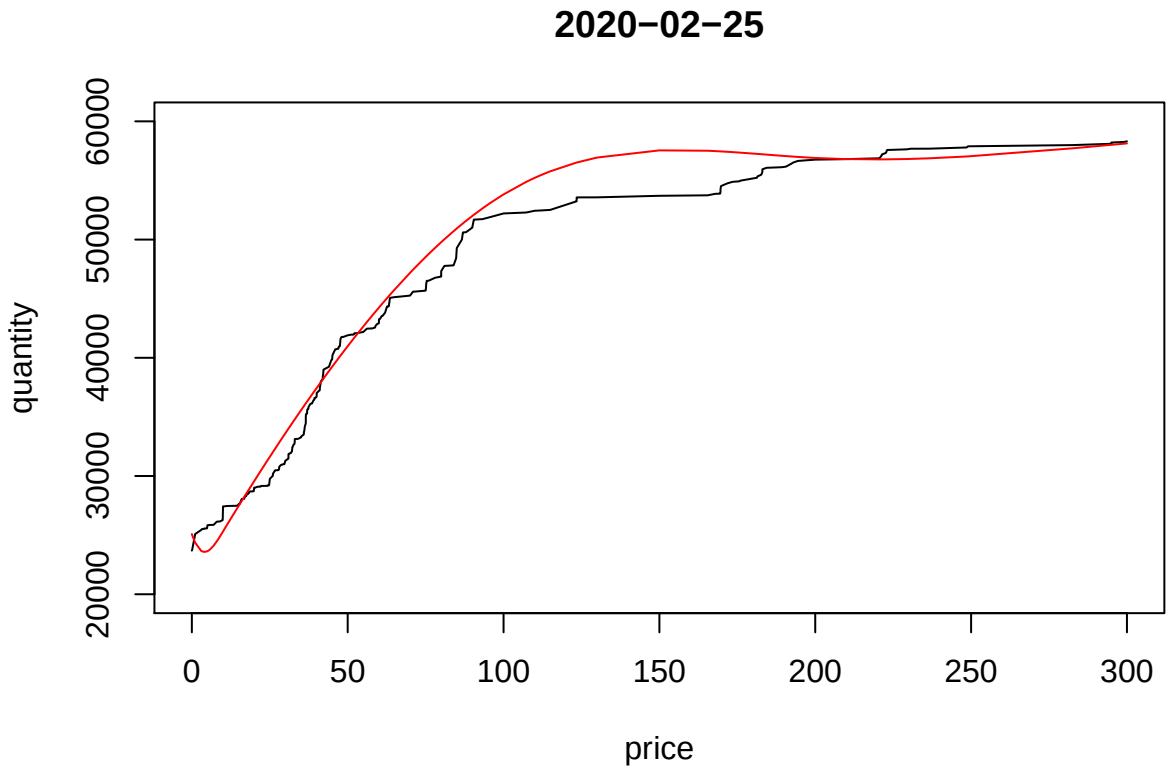
```
## [1] 8742265
```

```
sqrt(mean(mse))
```

```
## [1] 2956.732
```

Un esempio di stima per il 25/02/2020.

```
plot(next_day[[56]]$ENERGY_PRICE_NO, next_day[[56]]$sum, type= 'l', xlim= c(0,300),
     ylim = c(20000, 60000), main = ymd(next_day[[56]][1,'Date']),
     ylab = 'quantity', xlab= 'price')
lines(next_day[[56]]$ENERGY_PRICE_NO, next_day[[56]]$prev, col= 'red')
```



CONCLUSIONI

In conclusione, il secondo approccio è risultato essere più accurato secondo la metrica RMSE (4173.025, 2956.732 sull'intervallo 0-300). Il primo approccio si potrebbe preferire, invece, in un'ottica di interpretabilità.