Universitatea “Alexandru Ioan Cuza” din Iași
Facultatea de Economie și Administrarea Afacerilor
Specializarea Data Mining

PROIECT

Walmart Recruiting - Store Sales Forecasting

Big Data – Spark & R

Profesor coordonator:
Talabă George

Studenta:
Bargan Diana Georgiana

Cuprins

Introducere

În cadrul acestui proiect, mi-am propus să explorez și să analizez setul de date **"Walmart Recruiting - Store Sales Forecasting"** disponibil pe site-ul Kaggle, care cuprinde informații detaliate despre magazinele Walmart, vânzările lor săptămânale și caracteristicile asociate. Setul de date este structurat în trei fișiere principale: **stores.csv, train.csv, features.csv**.

Principalul obiectiv a fost să identific și să evaluez factorii care au o influență semnificativă asupra vânzărilor săptămânale la nivel de departament în cadrul fiecărui magazin Walmart. Mi-am propus să construiesc un model predictiv care să ajute la estimarea acestor vânzări, bazându-mă pe variabilele identificate ca fiind relevante.

Pentru a atinge acest obiectiv, am implementat și comparat mai multe modele de regresie prin aplicarea tehnicilor avansate de analiză de date în mediul Spark. Am început cu un **model de regresie liniar multiplu (MRLM)**, urmând să extind analiza la modele mai avansate, precum **Random Forest (RF)**, **Gradient Boosted Trees (GBT)** și **Decision Tree**. Voi compara aceste modele atât în scenarii fără validare încrucișată, cât și în scenarii cu validare încrucișată, pentru a determina care dintre ele oferă cea mai bună interpretare și predicție a setului de date.

1. Descrierea seturilor de date

Setul de date **„stores.csv"** include informații despre cele 45 de magazine, indicând tipul și dimensiunea magazinului.

Setul de date **„train.csv"** conține datele istorice ale vânzărilor săptămânale ale fiecărui departament.

Descrierea caracteristicilor:

Store	- 1–45 numărul magazinului.
Dept	- One of 1–99 numărul departamentului.
Date	- săptămâna
Weekly_Sales	- vânzări pentru departamentul respectiv din magazinul dat.
IsHoliday	- dacă săptămâna este o săptămână specială de vacanță.

Setul de date **„features.csv"** conține caracteristici suplimentare referitoare la magazine.

Descrierea caracteristicilor:

Store	numărul magazinului.
Date	săptămâna
Temperature	temperatura medie din regiune
Fuel_Price	prețul combustibilului din regiune în timpul săptămânii respective
MarkDown1-5	reprezintă tipul de reducere și cantitatea disponibilă în cursul acelei săptămâni. Este disponibil numai după noiembrie 2011 și nu este disponibil pentru toate magazinele tot timpul.
IPC	indicele prețurilor de consum în cursul săptămânii respective.
Unemployment	rata șomajului în cursul acelei săptămâni în regiunea magazinului
IsHoliday	dacă săptămâna este o săptămână specială de vacanță.

După etapa de curățare si prelucrare a seturilor de date, vom crea un set de date numit **spark_merge_df,** prin îmbinarea celor 3 seturi de date menționate pe coloane comune.

2. Curățarea și preprocesarea seturilor de date

Pentru început, cele 3 seturi de date au fost citite și salvate în dataframe-uri de tip **Spark**, utilizând funcția **sdf_copy()** din pachetul **sparklyr.** Fiecare set de date a fost analizat și curățat separat. Ulterior, s-a creat un set de date numit **spark_merge_df** prin îmbinarea celor 3 seturi de date menționate, pe coloane comune.

Curățarea setului de date spark_sales și afișarea statisticilor descriptive:

```
> glimpse(spark_sales)
Rows: ??
Columns: 5
Database: spark_connection
$ Store        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
$ Dept         <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
$ Date         <chr> "2010-02-05", "2010-02-12", "2010-02-19", "2010-02-26", "2010-03-0…
$ Weekly_Sales <dbl> 24924.50, 46039.49, 41595.55, 19403.54, 21827.90, 21043.39, 22136.…
$ IsHoliday    <lgl> FALSE, TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALS…
```

```
> spark_sales <- spark_sales %>%
+   mutate(Date = as.Date(Date))
> #Verificam daca avem valori NA in spark_sales dataframe:
> spark_sales %>%
+   summarise(across(everything(), ~ sum(as.integer(is.na(.))))) %>%
+   collect()
# A tibble: 1 × 5
  Store  Dept  Date Weekly_Sales IsHoliday
  <dbl> <dbl> <dbl>        <dbl>      <dbl>
1     0     0     0            0          0
```

```
> #Descriptive statistics for numerical variables from the sales dataframe
> sdf_describe(spark_sales) %>%
+   collect() %>%
+   pivot_longer(cols = -summary, names_to = "column", values_to = "value") %>%
+   pivot_wider(names_from = summary, values_from = value)
# A tibble: 3 × 6
  column       count  mean               stddev             min      max
  <chr>        <chr>  <chr>              <chr>              <chr>    <chr>
1 Store        421570 22.200545579619043 12.785297389902755 1        45
2 Dept         421570 44.26031738501317  30.492054015785932 1        99
3 Weekly_Sales 421570 15981.258123467243 22711.18351916329  -4988.94 693099.36
>
> ###Nu exista valori NA in setul de date, dar la o privire mai atenta,
> ##valorile negative sunt vizibile in vanzarile saptamanale !!
```

```
> spark_sales %>%
+   summarise(count = sum(ifelse(Weekly_Sales < 0, 1, 0)),
+             total_rows = n() ) %>%
+   mutate(percentage_negative = (count / total_rows) * 100) %>%
+   collect()
# A tibble: 1 × 3
  count total_rows percentage_negative
  <dbl>      <dbl>               <dbl>
1  1285     421570               0.305
>
> #Exista 1285 de rânduri cu valori negative din setul de date de 42.1570 de randuri,
> #ceea ce reprezinta doar 0,3% din intregul setul si, prin urmare, le vom elimina.
> # Eliminam valorile negative ale vanzarilor saptamanale:
> spark_sales <- spark_sales %>%
+   filter(Weekly_Sales > 0)
```

```
> #Summary vanzari saptamanale:
> spark_sales %>%
+   summarise(
+     count_sales = n(),
+     min_sales = min(Weekly_Sales),
+     mean_sales = mean(Weekly_Sales),
+     median_sales = median(Weekly_Sales),
+     max_sales = max(Weekly_Sales),
+     sd_sales = sd(Weekly_Sales) ) %>%
+   collect() %>%
+   print()
# A tibble: 1 × 6
  count_sales min_sales mean_sales median_sales max_sales sd_sales
        <dbl>     <dbl>      <dbl>        <dbl>     <dbl>    <dbl>
1      420212      0.01     16033.         7662.   693099.   22729.
>
> #Dimensiunea setului de date dupa preprocesare este:
> sdf_dim(spark_sales)
[1] 420212      5
```

Curățarea setului de date spark_features și afișarea statisticilor descriptive:

```
> glimpse(spark_features)
Rows: ??
Columns: 12
Database: spark_connection
$ Store        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
$ Date         <chr> "2010-02-05", "2010-02-12", "2010-02-19", "2010-02-26", "…
$ Temperature  <dbl> 42.31, 38.51, 39.93, 46.63, 46.50, 57.79, 54.58, 51.45, 6…
$ Fuel_Price   <dbl> 2.572, 2.548, 2.514, 2.561, 2.625, 2.667, 2.720, 2.732, 2…
$ MarkDown1    <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ MarkDown2    <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ MarkDown3    <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ MarkDown4    <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ MarkDown5    <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ CPI          <dbl> 211.0964, 211.2422, 211.2891, 211.3196, 211.3501, 211.380…
$ Unemployment <dbl> 8.106, 8.106, 8.106, 8.106, 8.106, 8.106, 8.106, 8.106, 7…
$ IsHoliday    <lgl> FALSE, TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FA…
> spark_features <- spark_features %>%
+   mutate(Date = as.Date(Date))
> #Verificam daca avem valori NA in sales dataframe:
> spark_features %>%
+   mutate(across(everything(), ~na_if(., "NA"))) %>%
+   summarise(across(everything(), ~ sum(as.integer(is.na(.))))) %>%
+   collect() %>%
+   print()
# A tibble: 1 × 12
  Store  Date Temperature Fuel_Price MarkDown1 MarkDown2 MarkDown3 MarkDown4
  <dbl> <dbl>       <dbl>       <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
1     0     0           0           0      4158      5269      4577      4726
# i 4 more variables: MarkDown5 <dbl>, CPI <dbl>, Unemployment <dbl>,
#   IsHoliday <dbl>
> #Deoarece nu avem prea multe informatii despre reduceri si cea mai mare
> #parte a acestora sunt NA, nu putem sa inlocuim valorile lipsa cu o
> #medie/mediana. In schimb, inlocuim NA cu valoarea 0, astfel incat sa putem
> #utiliza reducerile pentru ulterioarele analize.
> spark_features <- spark_features %>%
+   mutate(across(everything(), ~na_if(., "NA"))) %>%
+   mutate_at(vars(MarkDown1:MarkDown5), ~ ifelse(is.na(.), 0, .))
>
> glimpse(spark_features)
Rows: ??
Columns: 12
Database: spark_connection
$ Store        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
$ Date         <date> 2010-02-05, 2010-02-12, 2010-02-19, 2010-02-26, 2010-03-05,…
$ Temperature  <dbl> 42.31, 38.51, 39.93, 46.63, 46.50, 57.79, 54.58, 51.45, 62.2…
$ Fuel_Price   <dbl> 2.572, 2.548, 2.514, 2.561, 2.625, 2.667, 2.720, 2.732, 2.71…
$ MarkDown1    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ MarkDown2    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ MarkDown3    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ MarkDown4    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ MarkDown5    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ CPI          <dbl> 211.0964, 211.2422, 211.2891, 211.3196, 211.3501, 211.3806, …
$ Unemployment <dbl> 8.106, 8.106, 8.106, 8.106, 8.106, 8.106, 8.106, 8.106, 7.80…
$ IsHoliday    <lgl> FALSE, TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE…
> #Descriptive statistics features dataframe:
> sdf_describe(spark_features) %>%
+   collect() %>%
+   pivot_longer(cols = -summary, names_to = "column", values_to = "value") %>%
+   pivot_wider(names_from = summary, values_from = value) %>%
+   print()
# A tibble: 10 × 6
   column       count mean               stddev              min      max
   <chr>        <chr> <chr>              <chr>               <chr>    <chr>
 1 Store        8190  23.0               12.987966099514     1        45
 2 Temperature  8190  59.35619780219781  18.6786068489072    -7.29    101.95
 3 Fuel_Price   8190  3.4059918192918217 0.43133657110071383 2.472    4.468
 4 MarkDown1    8190  3462.090725274715  7388.916285890706   -2781.45 103184.98
 5 MarkDown2    8190  1206.9816642246678 5495.556014713521   -265.76  104519.54
 6 MarkDown3    8190  776.4642185592169  7539.953758202285   -179.26  149483.31
 7 MarkDown4    8190  1392.763114774119  4707.111488269868   0.0      67474.85
 8 MarkDown5    8190  2043.4037252747307 9431.22321501505    -185.17  771448.1
 9 CPI          7605  172.46080918276078 39.73834609860842   126.064  228.9764563
10 Unemployment 7605  7.826821038790305  1.877258593917429   3.684    14.313
>
> #Dimensiunea setului de date este:
> sdf_dim(spark_features)
[1] 8190   12
```

Analiza setului de date spark_features :

```
> #Dimensiunea setului de date este:
> sdf_dim(spark_stores)
[1] 45  3
> glimpse(spark_stores)
Rows: ??
Columns: 3
Database: spark_connection
$ Store <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, …
$ Type  <chr> "A", "A", "B", "A", "B", "A", "B", "A", "B", "B", "A", "B", "A", "A…
$ Size  <int> 151315, 202307, 37392, 205863, 34875, 202505, 70713, 155078, 125833…
> # Obtinerea valorilor unice din coloana Type
> spark_stores %>%
+   group_by(Type) %>%
+   summarise(count = n()) %>%
+   arrange(desc(count))%>%
+   collect()
# A tibble: 3 × 2
  Type  count
  <chr> <dbl>
1 A        22
2 B        17
3 C         6
```

Îmbinarea seturilor de date **spark_sales și spark_stores** pe baza atributelor magazinului, iar setul de date rezultat este îmbinat cu setul **spark_features** pe baza atributelor ***Store, Date și IsHoliday.***

```
spark_merge_df <- spark_sales %>%
  inner_join(spark_stores, by = "Store") %>%
  inner_join(spark_features, by = c("Store", "Date", "IsHoliday")) %>%
  sdf_register("merge_df")

#Convertim IsHoliday in variabila numerica cu 0 = False si 1 = True
#Cream si o noua variabila Type numeric unde A = 1, B=2, C = 3
spark_merge_df <- spark_merge_df %>%
  mutate(IsHoliday = as.integer(IsHoliday),
         # Convertim direct variabila booleana in 0 sau 1
         Type_numeric = case_when(
           Type == "A" ~ 1,
           Type == "B" ~ 2,
           Type == "C" ~ 3 ),
         #Adăugarea coloanelor „An”, „Lună” si „Saptamana” in merge_df
         Year = year(Date),
         Month = month(Date),
         WeekOfYear = weekofyear(Date)
  )
```

Setul de date **spark_merge_df** final are **20 de variabile și 420212 de observații**.

```
> glimpse(spark_merge_df)
Rows: ??
Columns: 20
Database: spark_connection
$ Store        <int> 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, …
$ Dept         <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
$ Date         <date> 2010-02-05, 2010-02-12, 2010-02-19, 2010-02-26, 2010-03-05,…
$ Weekly_Sales <dbl> 17426.75, 37734.82, 22135.29, 14942.21, 17098.49, 15535.51, …
$ IsHoliday    <int> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ Type         <chr> "B", "B", "B", "B", "B", "B", "B", "B", "B", "B", "B", "B", …
$ Size         <int> 112238, 112238, 112238, 112238, 112238, 112238, 112238, 1122…
$ Temperature  <dbl> 49.47, 47.87, 54.83, 50.23, 53.77, 50.11, 59.57, 60.06, 59.8…
$ Fuel_Price   <dbl> 2.962, 2.946, 2.915, 2.825, 2.987, 2.925, 3.054, 3.083, 3.08…
$ MarkDown1    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ MarkDown2    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ MarkDown3    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ MarkDown4    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ MarkDown5    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ CPI          <dbl> 126.4421, 126.4963, 126.5263, 126.5523, 126.5783, 126.6043, …
$ Unemployment <dbl> 13.975, 13.975, 13.975, 13.975, 13.975, 13.975, 13.975, 13.9…
$ Type_numeric <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, …
$ Year         <int> 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, …
$ Month        <int> 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 6, 6, 6, …
$ WeekOfYear   <int> 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 2…
> sdf_dim(spark_merge_df)
[1] 420212     20
```

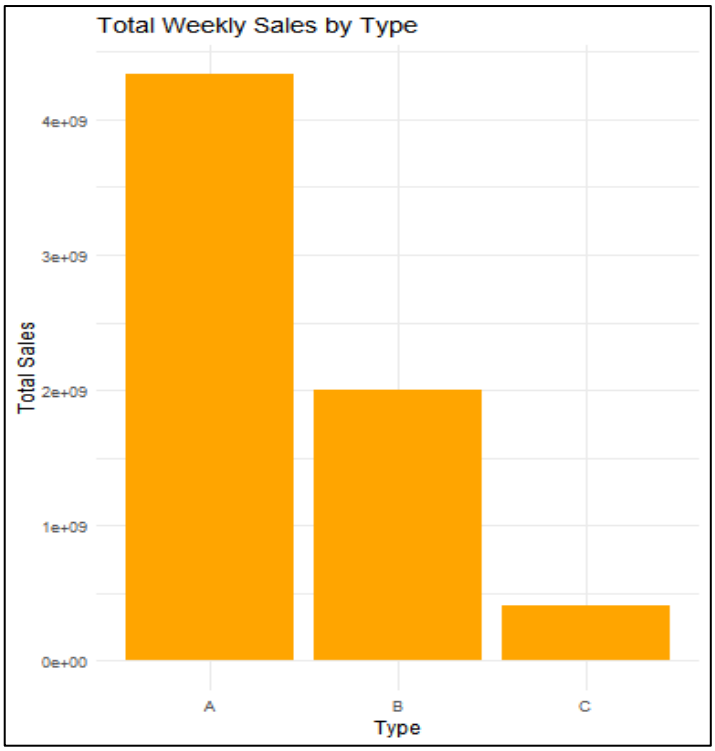
```
#Colectam datele in R pentru a putea realiza graficele si alte analize.
#Au fost salvate dataframe-urile curatate in R pentru a le folosi ulterior
#la incarcare mult mai usor.

r_merge_df <- collect(spark_merge_df)
r_features <- collect(spark_features)
r_sales<- collect(spark_sales)
r_stores<-collect(spark_stores)

#Salvare:
save.image(file = "all_dataframes_proiectDM1105.RData")
```

3. Analiza exploratorie a datelor (EDA)

3.1. Grafice

1) Calculul vânzărilor pentru fiecare tip de magazin

Se poate observa faptul că magazinele de tip A înregistrează cele mai mari vânzări, urmate de magazinele de tip B și în final magazinele de tip C.



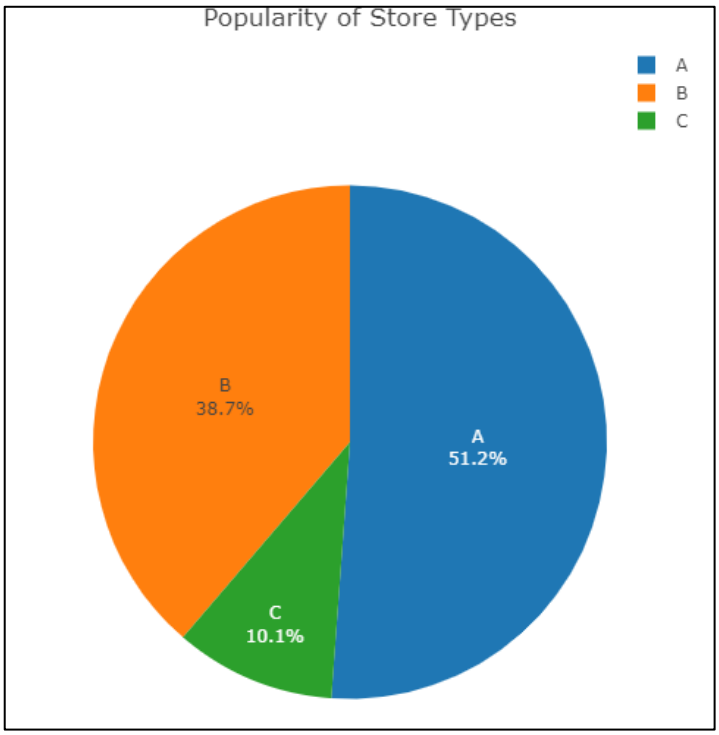
2) Calculul skewness si kurtosis pentru variabila Weekly_Sales

```
> skewness_value <- skewness(r_merge_df$Weekly_Sales)
> kurtosis_value <- kurtosis(r_merge_df$Weekly_Sales)
>
> print(paste("Skewness: ", skewness_value))
[1] "Skewness:  3.25893005854513"
> print(paste("Kurtosis: ", kurtosis_value))
[1] "Kurtosis:  24.4605238423249"
```

Distribuția are un skewness egal cu 3, ceea ce indica o asimetrie la dreapta, iar kurtosis=21 indică o distribuție puternic leptocurtică.

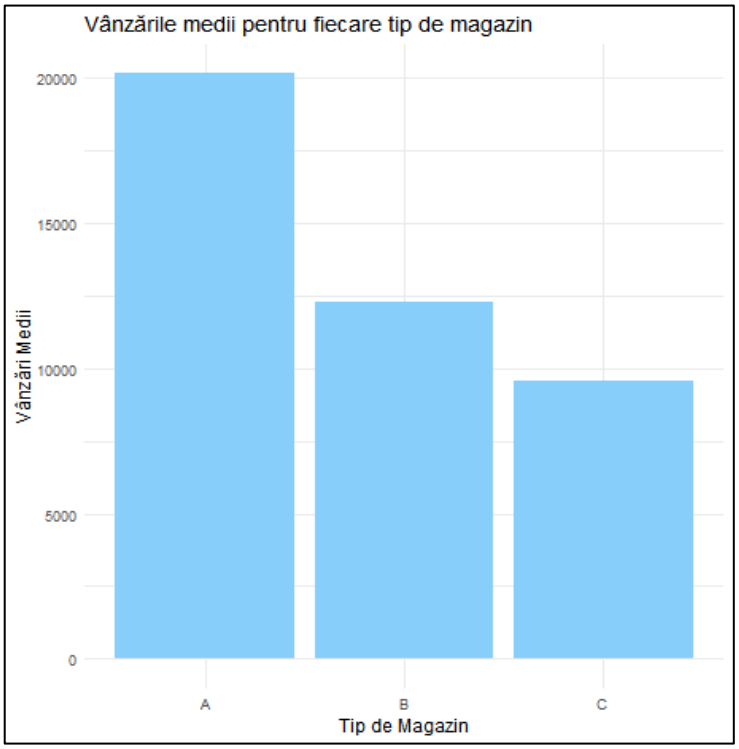
3) Popularitatea tipurilor de magazine

Magazinele de tip A sunt mai populare decât cele de tip B si C.



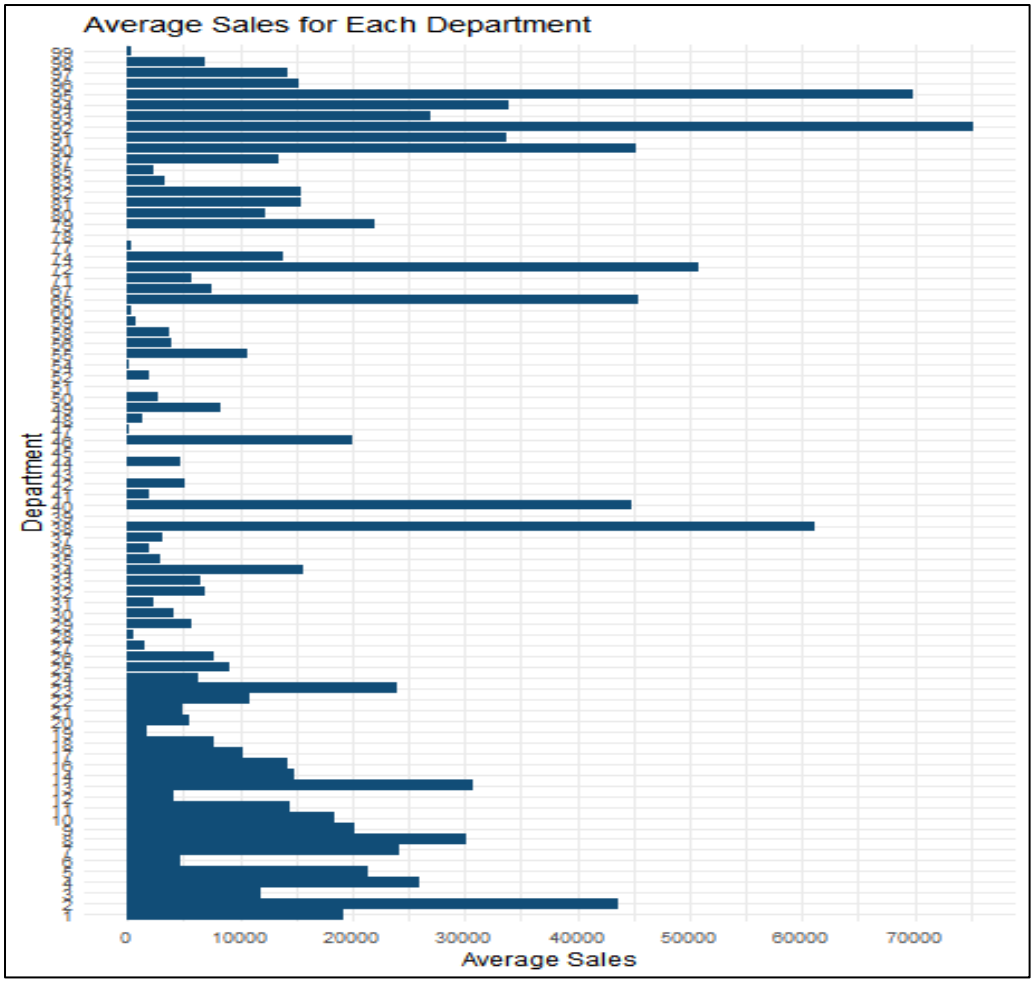
4) Vânzările medii în funcție de tipul magazinului

Pe baza graficului alăturat, se observă că magazinele de tip „A” au cele mai mari vânzări în comparație cu celelalte două magazine. Acest lucru ne spune că există o relație directă între dimensiunea magazinului și vânzările lor corespunzătoare. Magazinele de tip „B” ocupă locul al doilea ca mărime, numărul de magazine precum și vânzările medii, dovedind astfel această ipoteză.

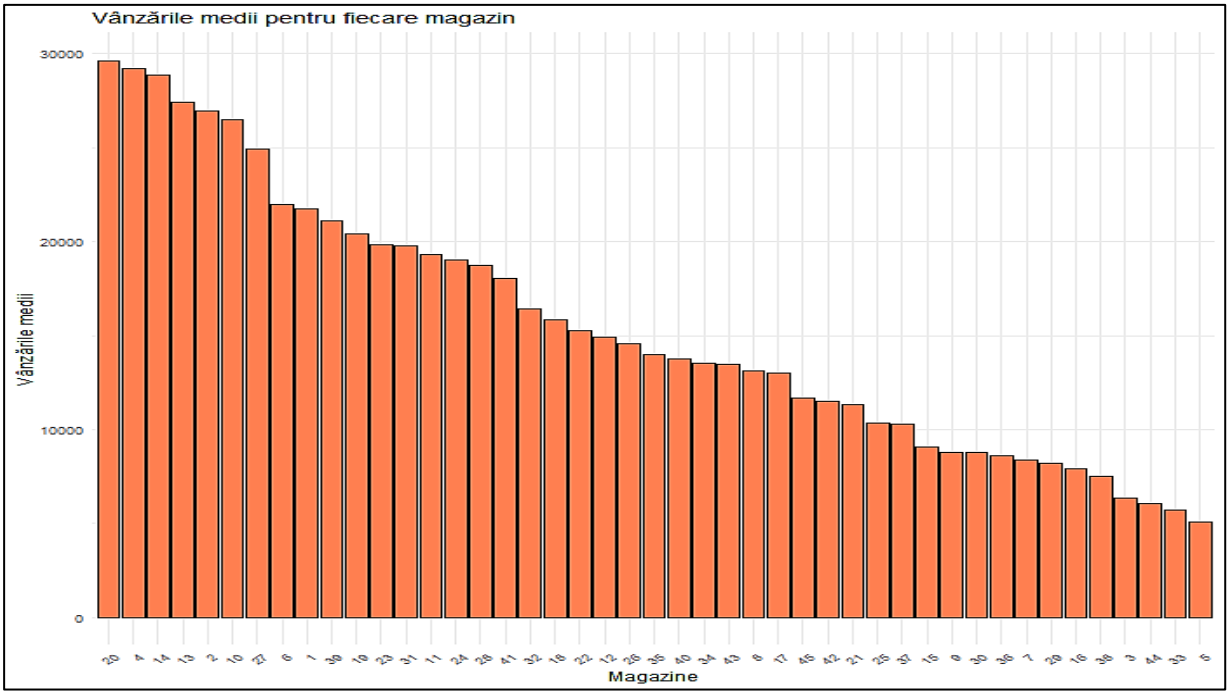


5) Vânzările medii pentru fiecare departament

Din imagine de mai jos, se observă că diferite departamente au prezentat niveluri diferite de vânzări medii. Departamentele cu numerele **92,95,38,72,65** prezintă cel mai mare număr de vânzări medii.

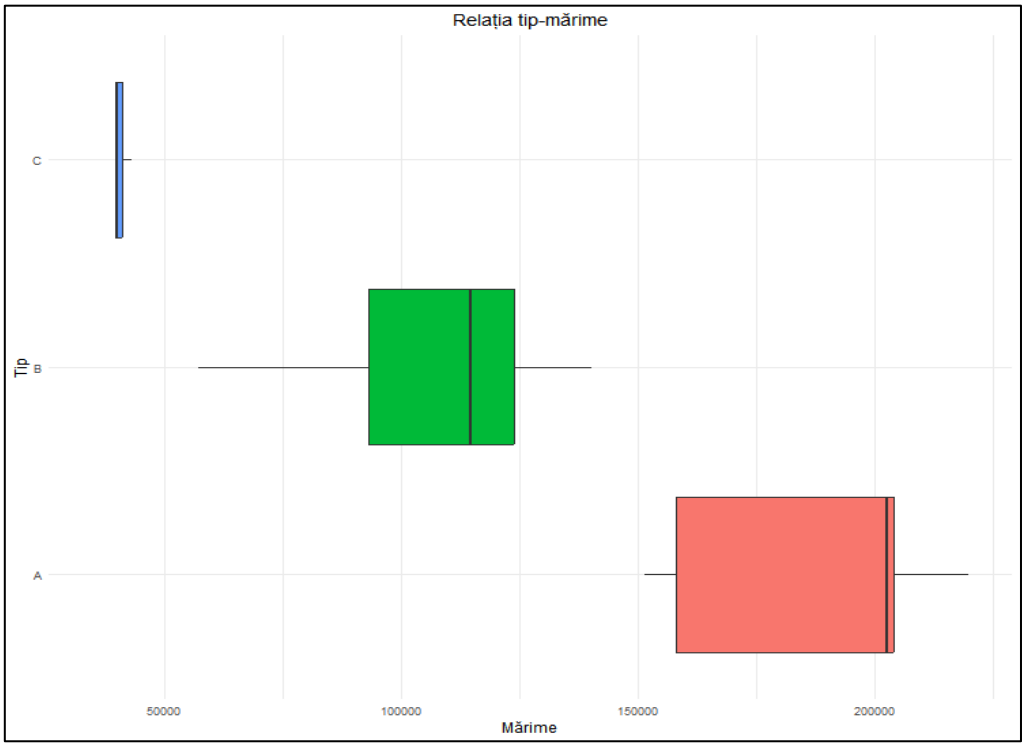


6) Vânzările medii pentru fiecare magazin



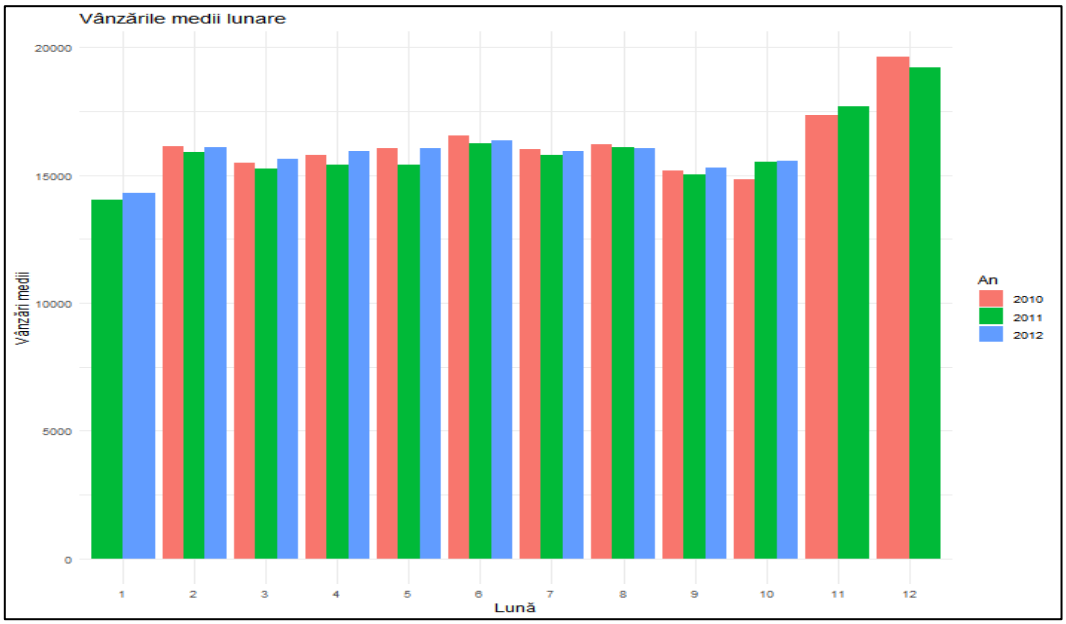
Din graficul de mai sus, putem observa că magazinele **20,4,14,13,2** au cele mai mari vânzări medii în toți cei trei ani. De asemenea, merită menționat că există o diferență foarte mare în ceea ce privește vânzările medii între magazine; în timp ce unele magazine înregistrează vânzări uriașe, altele rămân mult în urmă. Acest lucru poate depinde de factori precum tipurile de produse vândute, locația geografică, temperatura, rata șomajului etc. Un studiu suplimentar arată că toate aceste trei magazine aparțin magazinului de tip A care adună cele mai mari vânzări din toate cele trei tipuri de magazine.

7) Creare boxplot Type Size Store



Dimensiunea tipului de magazin este în concordanță cu vânzările, așa cum era de așteptat. Magazinele mai mari au vânzări mai mari. Walmart clasifică magazinele în funcție de dimensiunile lor, așa cum se arată în graficul de mai sus. După cea mai mică valoare a dimensiunii pentru tipul A, începe tipul B, urmat de tipul C.

8) Vânzările medii lunare pentru fiecare an



Observații din grafic:

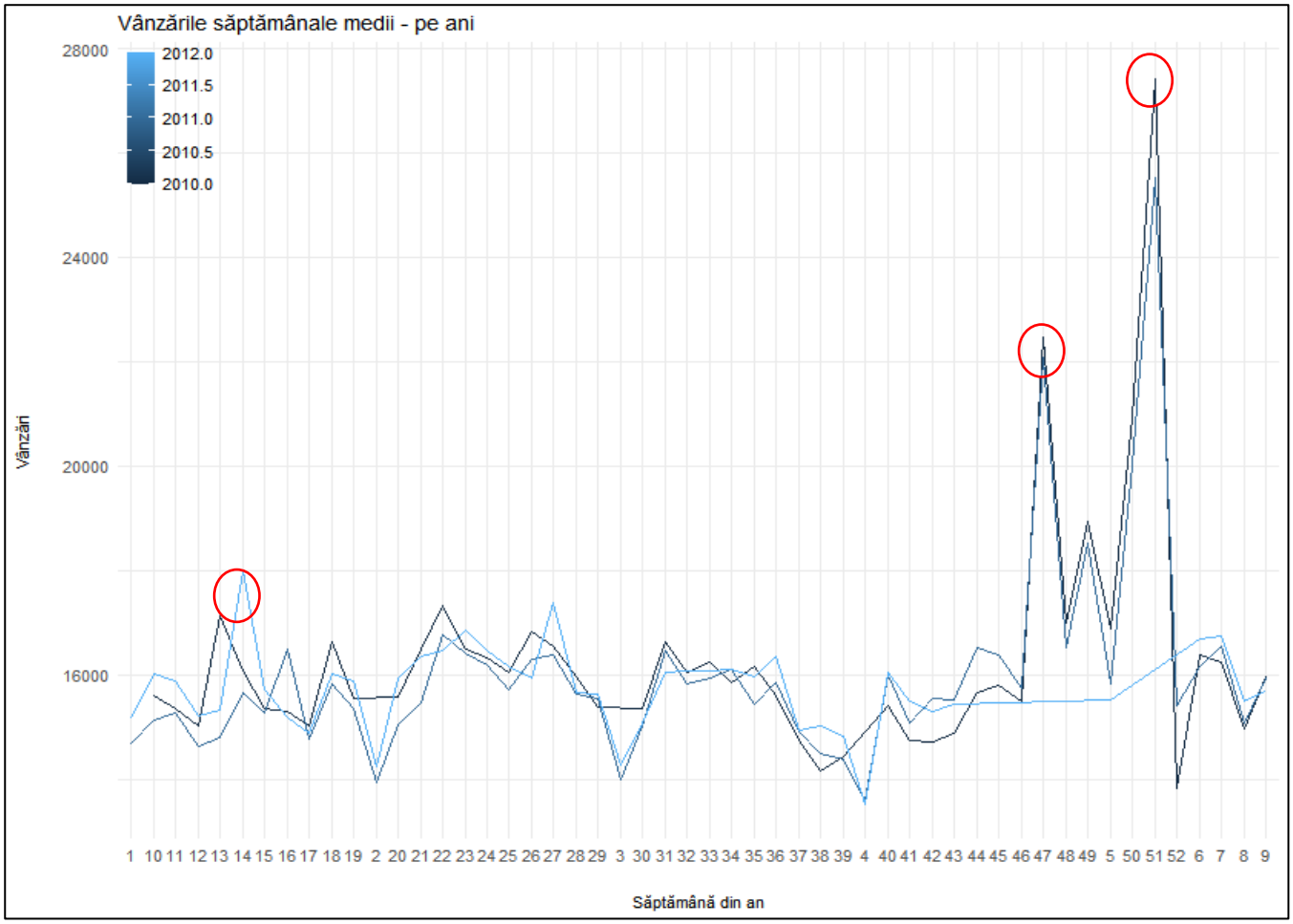
1. Luna ianuarie a avut cele mai scăzute vânzări pentru anii 2011 și 2012, în timp ce pentru 2010 nu s-au înregistrat vânzări săptămânale.

2. Din luna Februarie până în Octombrie, vânzările săptămânale au rămas aproape constante la aproximativ 15.000$ pentru toți cei trei ani.

3. Lunile Noiembrie și Decembrie au înregistrat cele mai mari vânzări pentru 2010 și 2011, în timp ce pentru 2012 nu au fost furnizate date privind vânzările.

9) Vânzări săptămânale medii - pe an

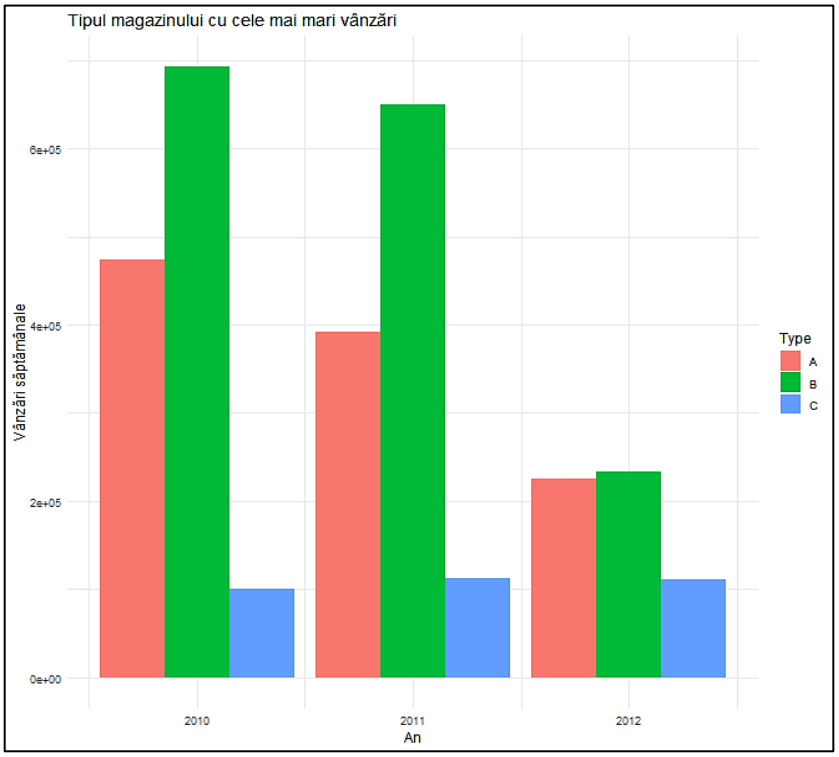
- Săptămânal, în săptămâna sărbătorii de Ziua Recunoștinței și cu o săptămână înainte de Crăciun, cele mai mari vânzări au fost înregistrate în 2010 și 2011. În 2012 săptămâna nr. 14 a înregistrat cele mai mari vânzări comparativ cu alte săptămâni ale anului, dar acest lucru nu corespunde vreunei sărbători sau evenimentului special.



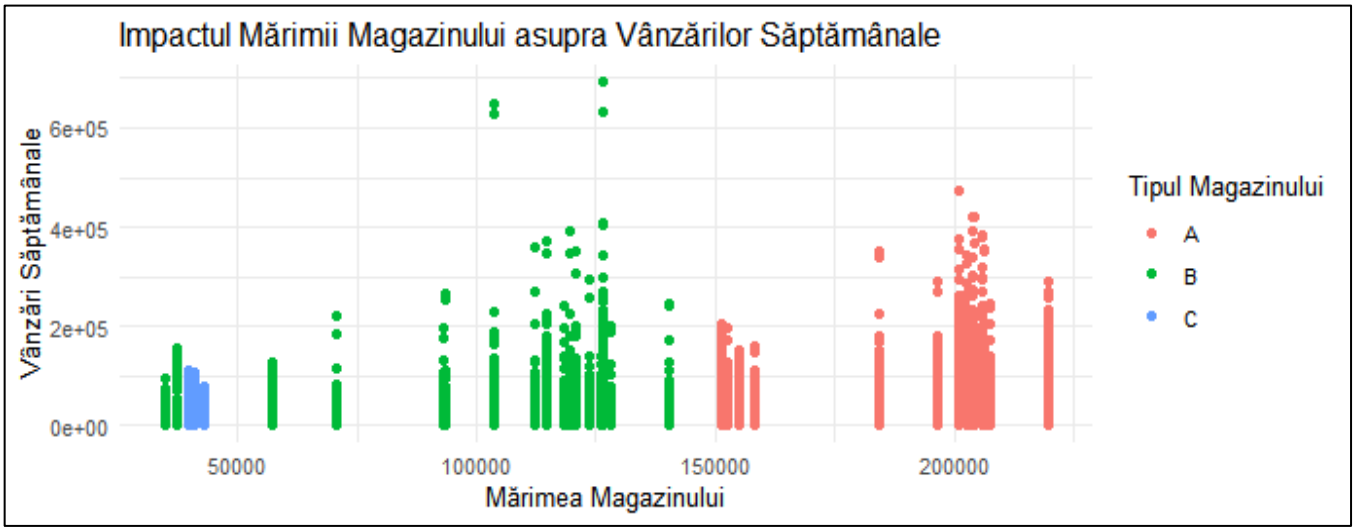
10) Tipul magazinului cu cele mai mari vânzări

Observăm faptul că în 2010, cele mai multe vânzări au fost înregistrate de magazinul tip B, urmat de magazinul tip A, și în final, de magazinul de tip C. Această structură menținându-se și în anul 2011.

În anul 2012, deși structura se menține, vânzările magazinului B au devenit aproape egale cu cele ale magazinului A. Ambele au scăzut ca volum față de anii precedenți, în timp ce vânzările magazinului C au crescut față de anii anteriori, însă au rămas tot sub nivelul celor doua magazine A și B.

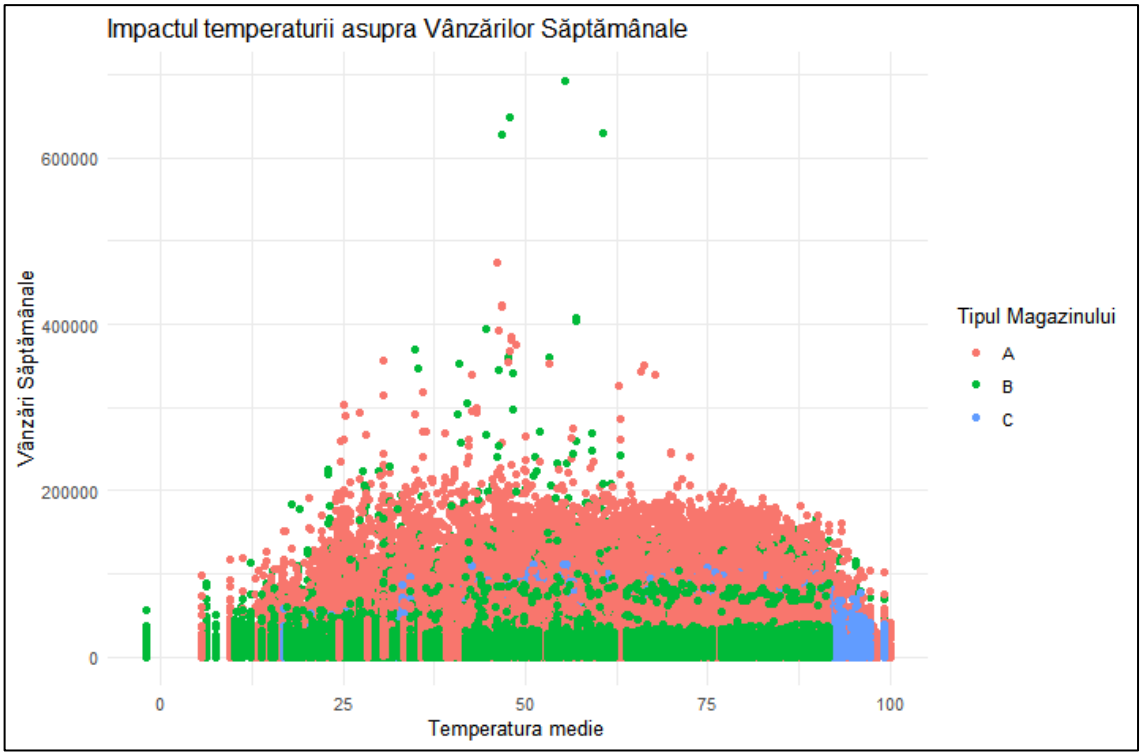


11) Impactul dimensiunii magazinului asupra vânzărilor



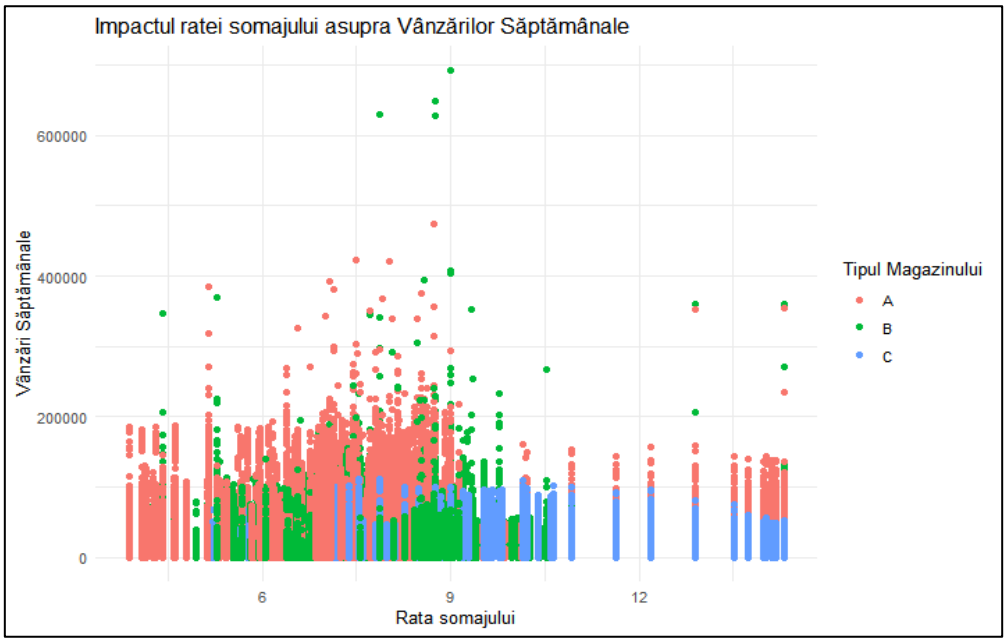
- Dintr-o vizualizare anterioară, s-a observat că tipul „A” este cel mai mare tip de magazin, urmat de tipurile „B” și „C”.
- Unele magazine de tip „B”, așa cum s-a descris mai sus, înregistrează vânzări medii mai mari decât magazinele de tip „A”, ceea ce contrazice ideea generală că, cu cât magazinul este mai mare, cu atât vânzările sunt mai mari. Cu toate acestea, în general, magazinele de tip „A” prezintă în continuare un volum mare de vânzări, în timp ce magazinele de tip „C” au un volum de vânzări semnificativ mai mic. În concluzie, vânzările cresc, în general, odată cu creșterea dimensiunii magazinului, cu câteva excepții minore.

12) Impactul temperaturii asupra vânzărilor



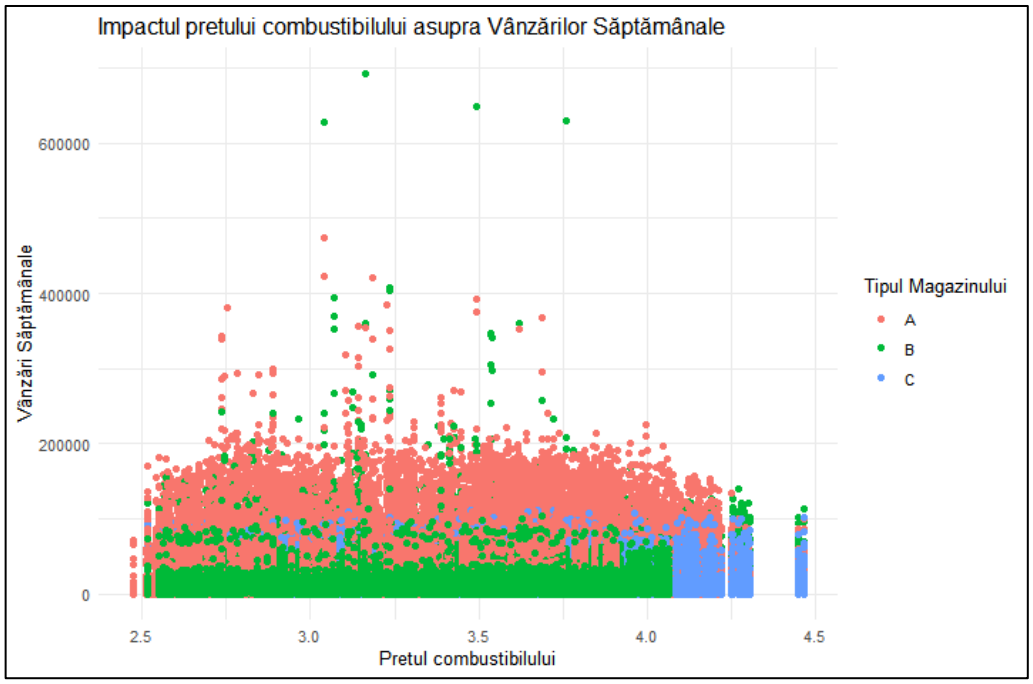
- După cum se vede in graficul de mai sus, cele mai mari vânzări pentru majoritatea tipurilor de magazine au loc atunci când temperaturile variază între 40 și 80 de grade Fahrenheit, ceea ce susține ideea că vremea plăcută tinde să stimuleze vânzările. Vânzările sunt relativ mai mici la temperaturi foarte scăzute sau extrem de ridicate, dar rămân destul de puternice în condiții climatice favorabile.

13) Impactul șomajului asupra vânzărilor



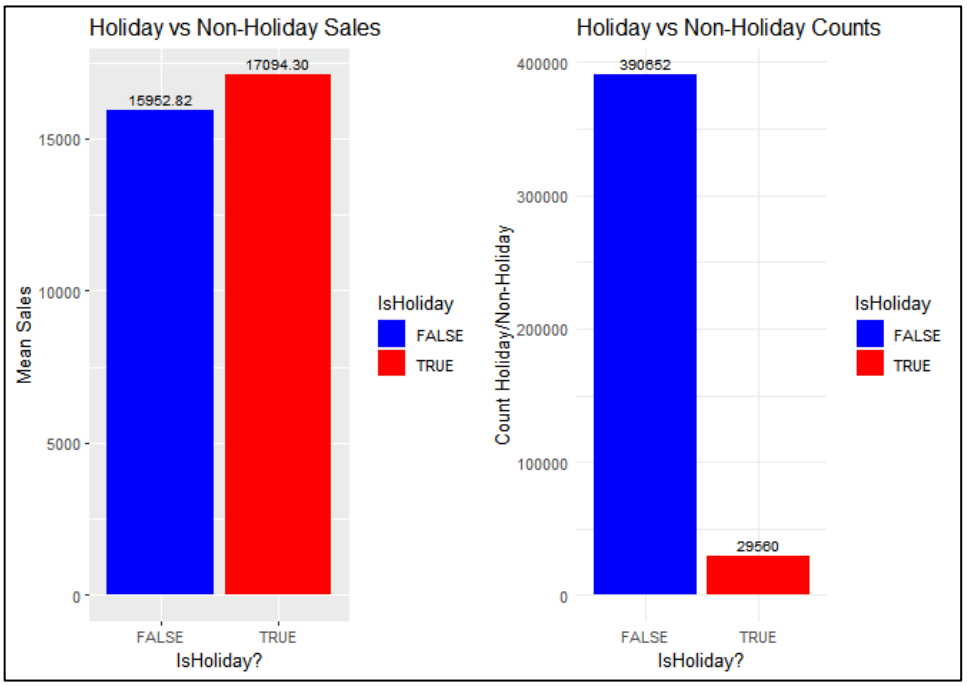
- Pentru tipurile de magazine date, se pare că există o scădere vizibilă a vânzărilor atunci când rata șomajului depășește valoarea 11. Chiar și atunci când indicele de șomaj este mai mare de 11, nu există nicio schimbare semnificativă a vânzările medii pentru magazinele de tip C în comparație cu vânzările totale.
- Se pare că există o scădere semnificativă a vânzărilor pentru magazinele de tip A și B pe măsura ce rata șomajului crește.
- Cele mai mari vânzări înregistrate pentru magazinele de tip A și B apar în jurul unei rate a șomajului de 8-10, ceea ce oferă perspective ambigue cu privire la impactul șomajului asupra vânzărilor pentru fiecare tip de magazin.

14) Impactul prețului combustibilului asupra vânzărilor



- Deși pare să existe o scădere a vânzărilor atunci când prețurile combustibililor sunt peste 4,25 USD, vânzările sunt mai mari atunci când prețurile combustibililor se încadrează în intervalul 2,75 USD - 3,75 USD. Unele dintre cele mai mari vânzări pentru magazinele de tip A și B au loc în acest interval. Deși nu există un model clar care să dovedească acest lucru, unele observații susțin teoria conform căreia prețurile mai mici la combustibili încurajează creșterea vânzărilor.

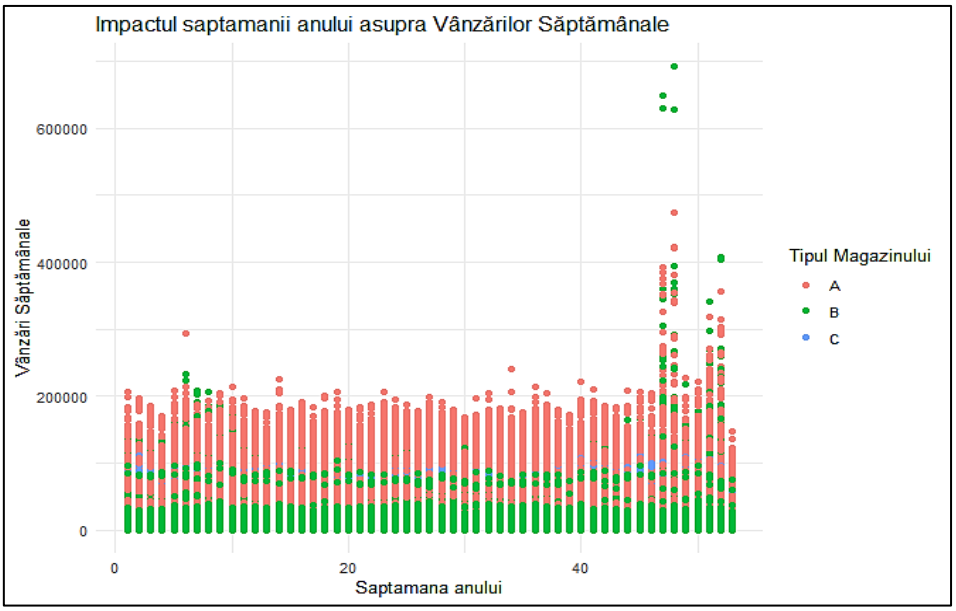
15) Vânzări zile Holiday vs non-holiday



- Setul de date furnizat conține date despre vânzările săptămânale ale Walmart în diferite perioade ale anului, inclusiv date despre vânzări din perioadele de sărbători, cum ar fi Ziua Recunoștinței, Crăciunul, Super Bowl și Ziua Muncii.
- Este esențial să se compare diferența dintre vânzările din timpul sărbătorilor și cele din săptămânile obișnuite pentru a înțelege dacă perioada de sărbători generează vânzări mai mari. Pentru această comparație, am numărat mai întâi numărul de sărbători dintr-un an și am comparat vânzările din timpul datelor de sărbătoare cu zilele obișnuite. Deși datele de sărbătoare au reprezentat doar aproximativ 7% din zilele unui an, acestea au înregistrat totuși vânzări săptămânale mai mari decât restul anului combinat (așa cum se vede în imaginea de mai jos). Vânzările și numărătoarea în care **„IsHoliday" = TRUE** pot fi observate în grafic de mai sus și evidențiază faptul că, deși numărul de date de sărbătoare este mult mai mic decât cel al datelor obișnuite din setul de date, acestea generează totuși vânzări mai mari.

16) Vânzările din săptămâna anului în funcție de tipul de magazin

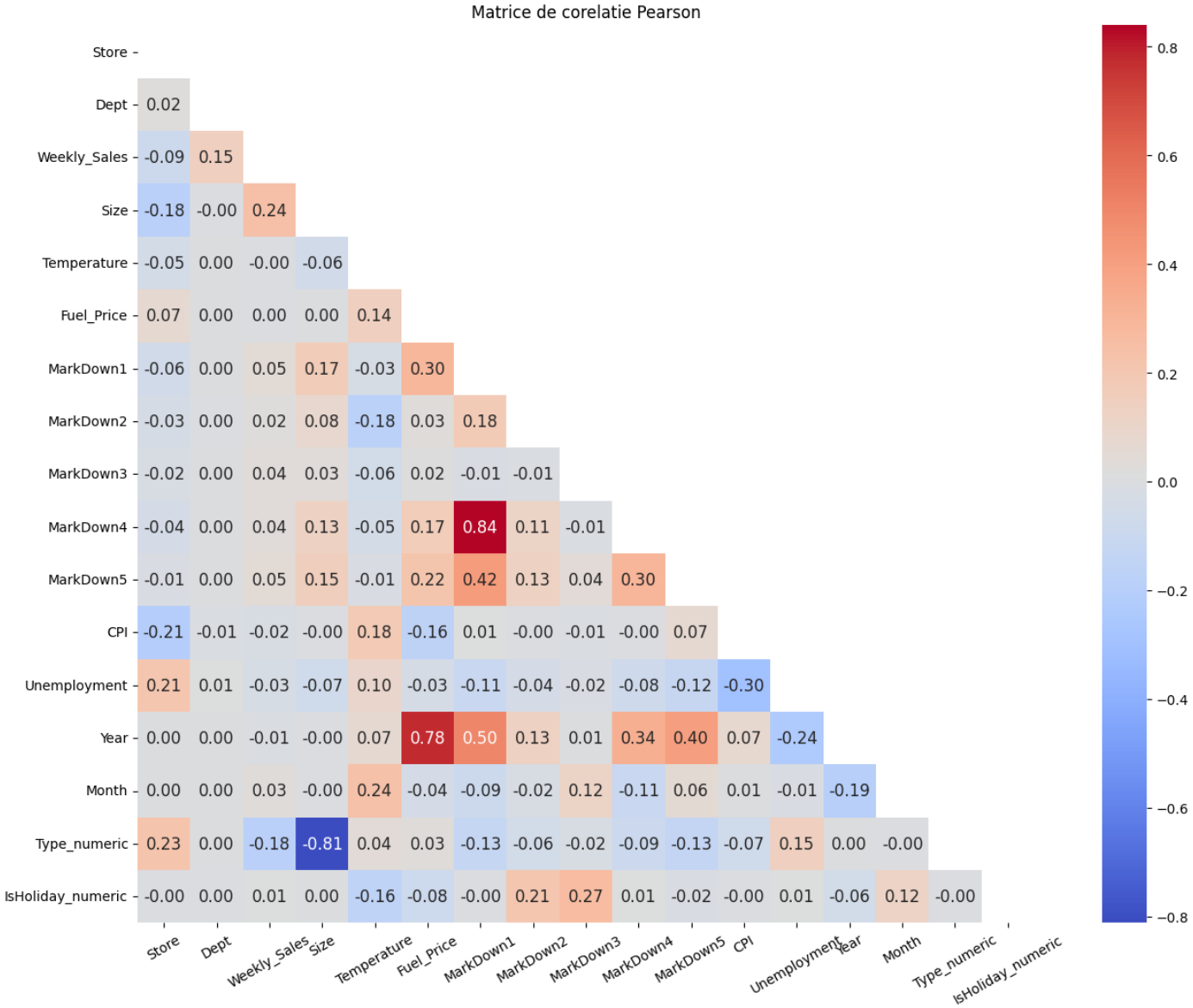
- Din graficul de mai jos, este clar că, spre deosebire de magazinele de tip A și B, vânzările medii pentru magazinele de tip C nu cresc neapărat la sfârșitul anului, în preajma Zilei Recunoștinței și a Crăciunului. În plus, graficul arată că magazinele de tip A au, în general, vânzări săptămânale mai mari în comparație cu celelalte două tipuri de magazine, demonstrând în continuare că o dimensiune mai mare a magazinelor.



3.2. Matricea de corelație

Observații din matricea de corelație:

- Există o corelație moderată între vânzările săptămânale și dimensiunea, tipul și departamentul magazinului.
- Se poate observa că există o corelație negativă între vânzările săptămânale și temperatură, respectiv rata șomajului, IPC și prețul combustibilului. Acest lucru ar putea sugera că vânzările nu sunt afectate semnificativ de modificările acestor factori.
- Reducerile de preț de la 1 la 5 nu par, de asemenea, să aibă o corelație distinctă cu vânzările săptămânale, ceea ce sugerează că acestea nu sunt un factor major în analiza lor.
- Cele mai mari corelații sunt între vânzările săptămânale și: departament (0,15), magazin (0,24) și tipul magazinului (0,18).



4. Construirea modelelor

4.1. Selecția eșantionului pentru construirea modelor ML

Din setul de date inițial, s-a realizat o selecție a datelor doar pentru **vânzările săptămânale din anul 2011.** În urma unei analize exploratorii și a studiului de corelație, au fost eliminate acele caracteristici care prezentau o corelație slabă și care nu influențează semnificativ variabila dependentă **Weekly_Sales**.

Caracteristicile eliminate sunt:

→ cele 5 tipuri de reduceri (Markdown(1:5)) deoarece prezintă o corelație slabă cu variabila obiectiv

→ temperatura aerului, prețul combustibilului, tipul de magazin, CPI, rata șomajului, și data la care s-au înregistrat vânzările.

```
s_mergedf_2011 <- spark_merge_df %>%
  filter(Year == 2011) %>%
  select(-Date, -Temperature, -Fuel_Price, -Type, -starts_with("MarkDown"), -CPI, -Month, -Unemployment) %>%
  sdf_register("s_mergedf_2011")

> glimpse(s_mergedf_2011)
Rows: ??
Columns: 8
Database: spark_connection
$ Store        <int> 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 1…
$ Dept         <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
$ Weekly_Sales <dbl> 11800.72, 11463.13, 12132.72, 13022.67, 14550.06, 23821.56, 44096.30,…
$ IsHoliday    <int> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ Size         <int> 112238, 112238, 112238, 112238, 112238, 112238, 112238, 112238, 11223…
$ Type_numeric <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, …
$ Year         <int> 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 201…
$ WeekOfYear   <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20…

> sdf_dim(s_mergedf_2011)
[1] 152940      8
```

Setul de date pe care îl vom utiliza în crearea modelelor ML **are 8 coloane si 152940 de observații.**

4.2. Împărțirea setului de date

Împărțirea setului de date inițial este realizată cu ajutorul funcției **sdf_random_split().** Această funcție împarte setul **s_mergedf_2011** în două subseturi: unul pentru antrenament, care conține 75% din date, și unul pentru test, care conține restul de 25%. Observăm că setul de antrenare obținut conține **114920** înregistrări, în timp ce setul de testare conține **38020** înregistrări. Cele două subseturi de date au fost încărcate în Spark.

```
model_partition_tbl <- s_mergedf_2011 %>%
  sdf_random_split(train = 0.75, test = 0.25, seed = 1234)

glimpse(model_partition_tbl)

# Creare referinte tabele si incarcarea datelor in Spark
s_mergedf2011_train <- sdf_register(model_partition_tbl$train, "s_mergedf2011_train")
s_mergedf2011_test <- sdf_register(model_partition_tbl$test, "s_mergedf2011_test")

sdf_dim(s_mergedf2011_train)
#[1] 114920      8
sdf_dim(s_mergedf2011_test)
#[1] 38020     8
```

4.3. Construirea modelelor fără validare încrucișata

În **Sparklyr**, construirea modelelor ML fără utilizarea validării încrucișate presupune antrenarea modelului pe întregul set de date de antrenament, evitând împărțirea acestuia în subseturi pentru evaluarea performanței pe setul de testare. Această abordare poate fi mai rapidă și mai puțin solicitantă din punct de vedere computațional, deoarece evită repetarea procesului de antrenare si evaluare pe mai multe subseturi. Cu toate acestea, lipsa validării încrucișate poate duce la o estimare mai puțin robustă a performanței modelului, deoarece nu beneficiază de avantajele generalizării pe mai multe subseturi de date.

În **sparklyr**, acest lucru se realizează prin utilizarea funcțiilor precum **ml_linear_regression** pentru regresie liniară, **ml_random_forest** pentru modelul Random Forest, **ml_gradient_boosted_trees** pentru Gradient Bosted Tree și **ml_decision_tree_regressor** pentru modelul Decison Tree, ce permit aplicarea directă a modelului pe setul de antrenament și evaluarea performanței pe setul de testare.

Pentru a crea un model de ML în Sparklyr, un prim pas esențial este definirea formulei care specifică relația dintre variabilele independente (predictori) și variabila dependentă (Weekly_Sales). Aceasta se realizează cu ajutorul funcției **formula(),** unde **Weekly_Sales** este desemnată ca variabilă dependentă, iar simbolul **~ .** indică utilizarea tuturor celorlalte coloane din setul de date ca predictori. Odată ce formula este creată, aceasta poate fi utilizată în funcțiile de modelare din Sparklyr pentru construirea modelelor predictive.

```
##Creare formula
ml_formula <- formula(Weekly_Sales ~ .)
```

Pentru evaluarea performanței modelelor, am construit o funcție numită **„compute_metrics()”** pentru automatizarea procesului de calcul a unor metrici esențiale, cum ar fi rădăcina erorii medie pătratice (RMSE) și coeficientul determinării (R-squared), pe baza predicțiilor modelului pe un set de date de testare.

```
compute_metrics <- function(model, test_data, response_var) {
  # Calculam predictiile pe setul de testare
  predictions <- ml_predict(model, test_data)

  # Calculam radacina erorii medie patratica (RMSE)
  rmse <- ml_regression_evaluator(predictions, label_col = response_var, metric_name = "rmse")

  # Calculam R-squared
  r_squared <- ml_regression_evaluator(predictions, label_col = response_var, metric_name = "r2")

  # Afisam RMSE si R-squared
  return(list(RMSE = rmse, R_squared = r_squared))
}
```

4.3.1. Regresia liniară multiplă (MRLM)

După cum se poate observa, modelul MRLM prezintă un **R-pătrat multiplu de 0.08834**, ceea ce indică faptul că aproximativ **8.83%** din variația vânzărilor săptămânale este explicată de variabilele independente selectate în model. Root Mean Squared Error (RMSE), care măsoară dispersia reziduurilor, ne oferă o perspectivă asupra acurateței modelului. O valoare RMSE de aproximativ **21,780** sugerează că, în medie, predicțiile modelului pe setul de antrenament diferă de valorile reale ale vânzărilor săptămânale cu aproximativ **21,780** unități.”

```
> #Model de regresie liniara multiplu
> model_lr <- ml_linear_regression(s_mergedf2011_train, formula = ml_formula)
> summary(model_lr)
Deviance Residuals (approximate):
   Min     1Q Median     3Q    Max
-32634 -13057  -5689   5590 608989

Coefficients:
  (Intercept)         Store          Dept     IsHoliday          Size  Type_numeric          Year    WeekOfYear
-6313.5762814   -88.6393332   115.1614099   868.7517318     0.1058962  1967.9394055     0.0000000    57.7956110

R-Squared: 0.08834
Root Mean Squared Error: 21780
```

```
> print(compute_metrics(model_lr, s_mergedf2011_test, "Weekly_Sales"))
$RMSE
[1] 22073.73

$R_squared
[1] 0.08704692
```

Comparând rezultatele între seturile de date de antrenament și de testare, observăm că există o ușoară diferență în performanța modelului între cele două seturi de date. RMSE-ul pe setul de testare (22073.73) este ușor mai mare decât pe setul de antrenament (21780) ceea ce indică o ușoară scădere a performanței modelului atunci când este evaluat pe setul de testare. Totuși, diferența este relativ mică, sugerând că modelul are o performanță consistentă pe ambele seturi de date. În ceea ce privește R-squared, valorile sunt apropriate pentru ambele seturi, indicând o capacitate stabilă a modelului de a explica variația vânzărilor săptămânale.

4.3.2. Modelul Random Forest

```
> model_rf <- ml_random_forest(s_mergedf2011_train, formula = ml_formula)
> summary(model_rf)
> print(compute_metrics(model_rf, s_mergedf2011_test, "Weekly_Sales"))
$RMSE
[1] 17884.36

$R_squared
[1] 0.4007009
```

4.3.3. Modelul Gradient Bosted Tree

```
> model_gbt <- ml_gradient_boosted_trees(s_mergedf2011_train, formula = ml_formula)
> summary(model_gbt)
> print(compute_metrics(model_gbt, s_mergedf2011_test, "Weekly_Sales"))
$RMSE
[1] 15013.78

$R_squared
[1] 0.5776456
```

4.3.4. Modelul Decision Tree Regression

```
# Model de regresie cu arbori de decizie
> model_dt <- ml_decision_tree_regressor(s_mergedf2011_train, formula = ml_formula)
> summary(model_dt)

# Calculare metrici pentru modelul de regresie cu arbori de decizie
> metrics_dt <- compute_metrics(model_dt, s_mergedf2011_test, "Weekly_Sales")
> print(metrics_dt)
$RMSE
[1] 18058.37

$R_squared
[1] 0.3871539
```

4.3.5 Comparație performanță modele fără validare încrucișată

Performanța modelelor predictive este determinată prin calculul celor două metrici esențiale **rădăcina pătrată a erorii medii pătratice (RMSE)** și **coeficientul de determinare (R-squared)** pe setul de testare, folosind funcția **compute_metrics().** Rezultatele sunt apoi prezentate atât într-un tabel pentru comparații directe, dar și prin intermediul unei reprezentări grafice pentru observarea tendințelor și distribuției performanței într-un mod mai dinamic și accesibil.

```
#### Calculare metrici pentru fiecare model
metrics_lr <- compute_metrics(model_lr, s_mergedf2011_test, "Weekly_Sales")
metrics_rf <- compute_metrics(model_rf, s_mergedf2011_test, "Weekly_Sales")
metrics_gbt <- compute_metrics(model_gbt, s_mergedf2011_test, "Weekly_Sales")
metrics_dt <- compute_metrics(model_dt, s_mergedf2011_test, "Weekly_Sales")

###Afisare metrici pentru fiecare model sub forma de tabel
all_metrics <- bind_rows(
  list(metrics_lr, metrics_rf, metrics_gbt, metrics_dt), .id = "Model") %>%
  mutate(Model = c("MLR",
                   "Random Forest",
                   "Gradient Boosted Tree",
                   "Decision Tree")[as.integer(Model)])
```

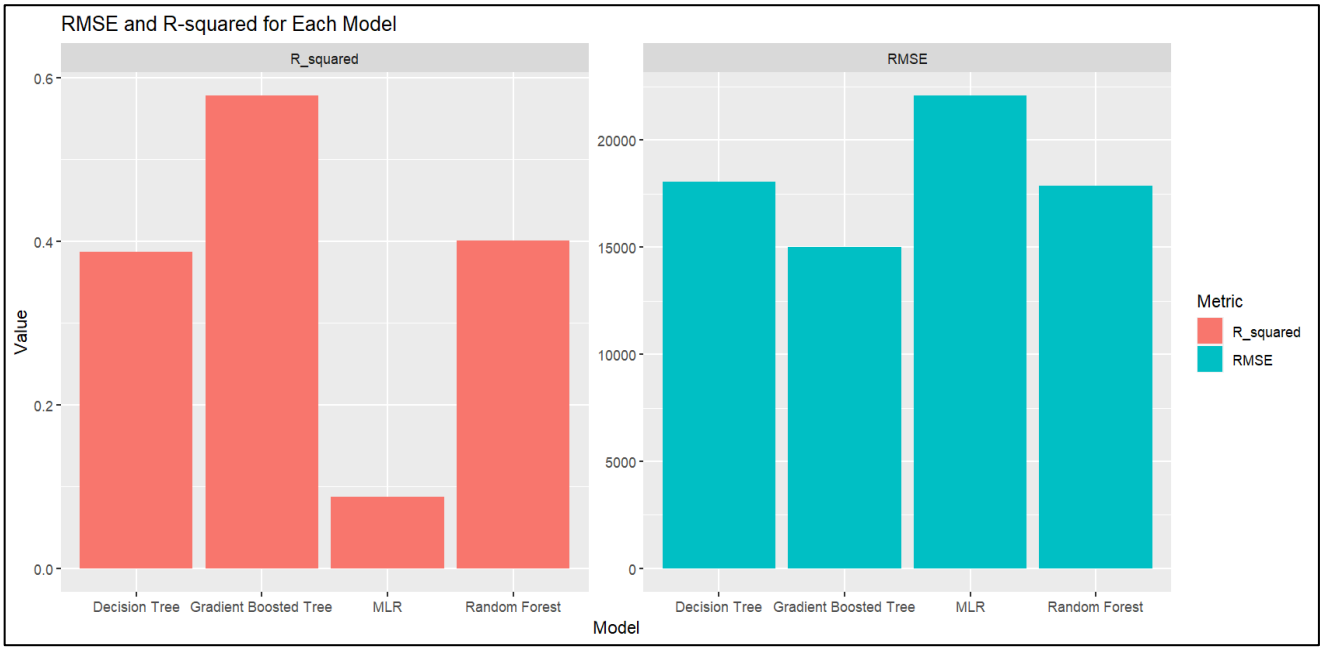
Tabelul metricilor de performanță a modelelor fără validare încrucișată :

```
> all_metrics
      RMSE  R_squared                     Model
1 22073.73 0.08704692                       MLR
2 17884.36 0.40070090             Random Forest
3 15013.78 0.57764557 Gradient Boosted Tree
4 18058.37 0.38715390             Decision Tree
```

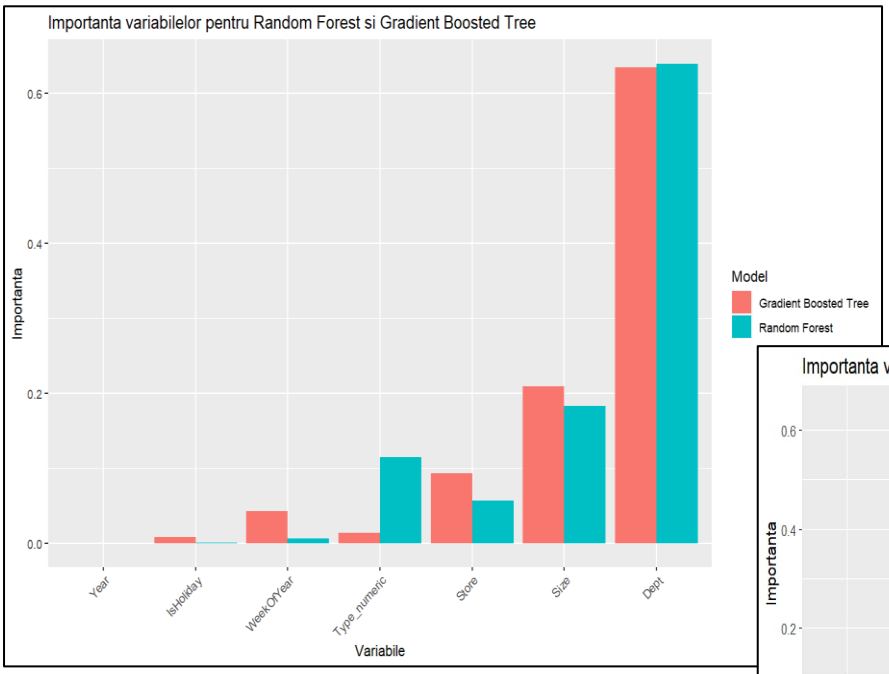
Analizând atât tabelul cu metricile de performanță cât și reprezentarea grafică de mai jos, putem observa că Random Forest, Gradient Boosted Tree și Decision Tree prezintă valori mai scăzute pentru metrica RMSE (Root Mean Squared Error) în comparație cu modelul de regresie liniar multiplu. Acest lucru indică faptul că modelele bazate pe arbori de decizie au o capacitate mai bună de a estima valorile reale ale vânzărilor săptămânale în comparație cu modelul liniar.

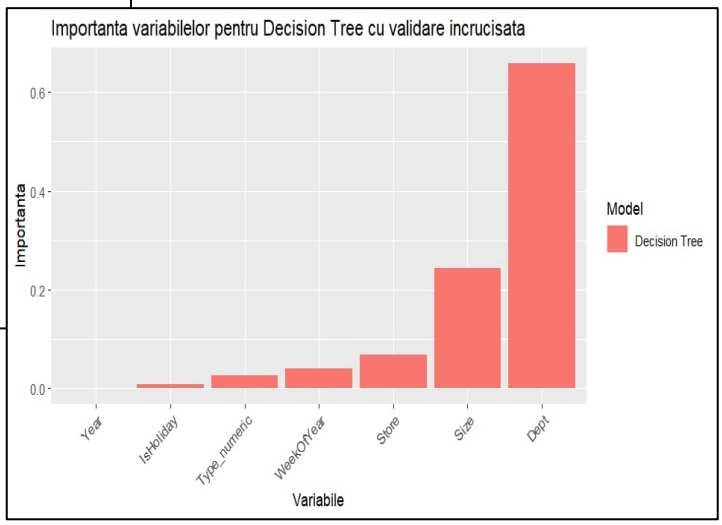
Cu toate acestea, **Gradient Boosted Tree se remarcă ca fiind cel mai performant model dintre cele analizate.** Acesta nu doar că depășește modelul MLR, dar și modelele Random Forest și Decision Tree oferind cea mai mică valoare a RMSE = 15014 și un coeficient mai mare a R-squared = 0.578 sugerând că aproximativ **57.8%** din variabilitatea datelor este explicată de model.

Plot comparația metricilor de performanță ale celor 4 modele:



Plot importanța variabilelor pentru modelele RF, GBT și DT :





În toate cele 3 modele obținute, departamentul (Dept) este cel mai important predictor ce are o influență covârșitoare asupra vânzărilor săptămânale. Pe locul doi este dimensiunea magazinului (Size), pe locul trei este magazinul (Store) și pe locul patru este tipul de magazin (Type_numeric) pentru modelele RF si GBT, in timp ce pentru modelul DT locul patru este ocupat de săptămâna anului. Restul predictorilor din toate modele au o influență extrem de mică sau aproape deloc asupra vânzărilor săptămânale.

4.4. Construirea modelelor cu validare încrucișată

În vederea construirii modelelor de regresie cu validare încrucișată, s-a creat o funcție numită **ml_regression_cv()** pentru a automatiza procesul de validare încrucișată și reglare a hiperparametrilor. Această funcție permite evaluarea performanței modelelor pe mai multe seturi de date și identificarea celor mai buni hiperparametri pentru model. Funcția primește ca input un dataframe Spark care conține datele de antrenament, coloana care conține valorile de predicție (variabila dependentă), un vector cu numele variabilelor independente, funcția de modelare, numărul de folduri pentru validare încrucișată și un grid de hiperparametri pentru tuning.

```r
ml_regression_cv<- function(
    data,              # input spark dataframe
    response,          # Predicted column
    features = NULL, # character vector or else all the columns except response
    model_fun,         # the modelling function to use (unquoted)
    k = 5,             # number of folds
    param_grid = NULL # hyperparameter grid for tuning
) {

  # Crearea ponderilor pentru partitionarea datelor
  weights <- rep(1 / k, times = k)
  names(weights) <- paste0("fold", as.character(1:k))

  # Partitionarea datelor folosind ponderile
  data_cv <- sdf_random_split(data, weights = weights)

  # Obtinerea indicilor pentru diferitele seturi de antrenament
  K <- 1:k
  indices <- purrr::map(K, ~ K[-.x])

  # Crearea seturilor de antrenament prin combinarea partitiilor din data_cv
  data_splits <- purrr::map(indices, ~ sdf_bind_rows(data_cv[.x]))

  # Daca nu a fost specificat un vector de nume de caracteristici
  if (is.null(features)) {
    columns <- colnames(data_splits[[1]])
    features <- columns[columns != response]
  }
  # Reglare de hiperparametri
  fits <- purrr::map(data_splits, function(train_data) {
    if (!is.null(param_grid)) {
      best_model <- NULL
      best_metric <- Inf
      best_params <- NULL
      for (params in param_grid) {
        model <- do.call(model_fun, c(list(train_data, response = response, features = features), params))
        predictions <- ml_predict(model, train_data)
        metric <- ml_regression_evaluator(predictions, label_col = response, metric_name = "rmse")
        if (metric < best_metric) {
          best_metric <- metric
          best_model <- model
          best_params <- params
        }
      }
      return(list(model = best_model, params = best_params))
    } else {
      model <- model_fun(train_data, response = response, features = features)
      return(list(model = model, params = NULL))
    }
  })

  # Generarea predictiilor pentru fiecare fold
  preds <- purrr::map2(fits, K, ~ ml_predict(.x$model, data_cv[[.y]]))

  # Evaluarea modelelor folosind evaluatori de regresie
  evals <- purrr::map2_df(preds, fits, ~ {
    rmse <- ml_regression_evaluator(.x, label_col = response, metric_name = "rmse")
    r2 <- ml_regression_evaluator(.x, label_col = response, metric_name = "r2")
    params <- .y$params
    if (is.null(params)) params <- list(num_trees = NA, max_depth = NA)
    tibble(
      RMSE = rmse,
      R_squared = r2,
      num_trees = params$num_trees,
      max_depth = params$max_depth
    )
  })

  # Returnarea rezultatelor
  list(fits = fits, predictions = preds, evals = evals)
}
```

Pașii principali ai funcției sunt:

1) **Partiționarea datelor:** funcția împarte dataframe-ul de intrare în „k” folduri folosind ponderi egale, asigurând o distribuție echilibrată a datelor pentru validarea încrucișată, prin intermediul funcției **„sdf_random_split().**
2) **Generarea seturilor de antrenament și testare:** Pentru fiecare fold, funcția generează indicii corespunzători seturilor de antrenament, excluzând unul dintre folduri pentru a-l folosi ca set de testare, în timp ce combină restul foldurilor pentru a forma seturile de antrenament.
3) **Reglarea hiperparametrilor (dacă este necesar) și evaluarea performantei:** Dacă este furnizat un grid de hiperparametri, funcția parcurge toate combinațiile de hiperparametri specificate în grid, antrenând modelele pe seturile de antrenament și evaluându-le pe baza metricilor **RMSE și R-squared**. Modelul cu cea mai bună performanță (cel mai mic RMSE) este selectat pentru fiecare set de antrenament. Dacă nu este furnizat niciun grid de hiperparametri, funcția antrenează un model simplu folosind funcția de model specificată.

Rezultatul final al funcției este o listă care conține modelele antrenate, predicțiile realizate și evaluările performanței pentru fiecare fold.

4.4.1. Modelul Random Forest

Implementarea modelului de regresie Random Forest (RF) prin tehnica validării încrucișate implică utilizarea unui set de hiperparametri predefiniți pentru a determina cea mai bună configurație a modelului. Dupa cum se poate observa in imaginea de mai jos, s-a definit o grilă de hiperparametri specifică, care include două combinații distincte de valori pentru numărul de arbori și adâncimea maximă a acestora. Prima combinație testează un model cu 20 de arbori, fiecare având o adâncime de 5 niveluri, în timp ce a doua crește complexitatea modelului la 50 de arbori cu o adâncime de 10 niveluri. Această abordare permite evaluarea și compararea performanței modelului sub diferite condiții de complexitate, asigurând alegerea celei mai potrivite configurații pentru predicția datelor de interes, in cazul nostru predicția vânzărilor săptămânale (Weekly_Sales).

Utilizând funcția **ml_regression_cv**(), modelul RF este instruit pe setul de date **s_mergedf2011_train**, cu **Weekly_Sales** ca variabilă țintă. Se aplică validarea încrucișată cu **k = 5 folduri,** pentru a asigura că subset de date este folosit atât pentru antrenament cât și pentru testare.

```
# Combinatii de hiperparametri pentru Random Forest
param_grid_rf <- list(
  list(num_trees = 20, max_depth = 5),
  list(num_trees = 50, max_depth = 10)
)

# Validare incrucisata pentru Random Forest
rf_cv3 <- ml_regression_cv(
  s_mergedf2011_train,
  response = "Weekly_Sales",
  model_fun = ml_random_forest,
  k = 5,
  param_grid = param_grid_rf
)
```

4.4.2. Modelul Gradient Bosted Tree

La fel ca în cazul modelului Random Forest, pentru Gradient Boosted Trees (GBT) s-au urmat pași similari de validare încrucișată. Diferența principală între cele două modele constă în setările specifice ale hiperparametrilor și în algoritmul utilizat. Pentru GBT, grila de hiperparametri include combinații de **max_iter** (numărul maxim de iterații) și **max_depth** (adâncimea maximă), care sunt esențiale în ajustarea și optimizarea performanței modelului GBT.

În timp ce RF construiește mai mulți arbori de decizie în mod independent și apoi le combină rezultatele, GBT construiește arborii secvențial, fiecare nou arbore îmbunătățind erorile celui anterior. Acest lucru face ca GBT să fie adesea mai complex și costisitor în timp de antrenare față de RF. Validarea încrucișată cu 5 foldere asigură că modelul este robust și capabil să generalizeze bine pe date noi.

```
# Combinatii de hiperparametri pentru Gradient Boosted Trees
param_grid_gbt <- list(
  list(max_iter = 20, max_depth = 5),
  list(max_iter = 50, max_depth = 10)
)

# Validare incrucisata pentru Gradient Boosted Trees
gbt_cv3 <- ml_regression_cv(
  s_mergedf2011_train,
  response = "Weekly_Sales",
  model_fun = ml_gradient_boosted_trees,
  k = 5,
  param_grid = param_grid_gbt
)
```

4.4.3. Modelul Decision Tree Regression

În ceea ce privește modelul Decision Tree Regression, acesta a fost construit urmând o metodologie similară cu cea aplicată pentru celelalte două modele. Cu toate acestea, ceea ce diferențiază modelul Decision Tree de celelalte este abordarea simplificată a grilei de hiperparametri, limitându-se la explorarea adâncimii maxime a arborilor (**max_depth**). Prin limitarea hiperparametrilor la adâncimea maximă, am putut observa mai clar modul în care structura arborelui influențează precizia predicțiilor vânzărilor săptămânale. Acest lucru ne permite să comparăm eficiența unui model mai simplu, Decision Tree, cu complexitatea adăugată în modelele RF și GBT, unde numărul de arbori și iterații contribuie la o capacitate predictivă sporită

```
# Combinatii de hiperparametri pentru Decision Tree
param_grid_dt <- list(
  list(max_depth = 5),
  list(max_depth = 10)
)

# Validare incrucisata pentru Decision Tree Regression
dt_cv3 <- ml_regression_cv(
  s_mergedf2011_train,
  response = "Weekly_Sales",
  model_fun = ml_decision_tree_regressor,
  k = 5,
  param_grid = param_grid_dt
)
```

4.4.4. Comparație performanță modele cu validare încrucișată

Pentru evaluarea și compararea performanței modelor RF, GBT și DT, au fost extrase rezultatele evaluării fiecărui model și apoi combinate într-un tabel, permițând o comparație directă între modele. Acest tabel include metricile RMSE (Root Mean Square Error) și R-squared obținute pe fiecare fold creat dar și cea mai bună combinație de hiperparametri selectată.

```
# Extragem evaluarile intr-un dataframe
rf_results <- rf_cv3$evals
gbt_results<- gbt_cv3$evals

# Combinam rezultatele intr-un singur tabel
all_results_mod_csv <- bind_rows(
  rf_results %>% mutate(Model = "RF", Fold = rep(1:5, each = nrow(rf_results) / 5)),
  gbt_results %>% mutate(Model = "GBT", Fold = rep(1:5, each = nrow(gbt_results) / 5))
)
```

```
#Rezultatele extrase intr-un dataframe pentru DT:
dt_results <- dt_cv3$evals
all_results_mod_csv <- bind_rows(
  dt_results %>% mutate(Model = "DT", Fold = rep(1:5, each = nrow(dt_results) / 5))
)
```

Tabelele metricilor de performanță cu cea mai bună combinație de hiperparametri pentru modelele cu validare încrucișată:

Tabel 1 : Metricile pentru RF si GBT

```
> all_results_mod_csv
# A tibble: 10 × 6
    RMSE R_squared num_trees max_depth Model  Fold
   <dbl>     <dbl>     <dbl>     <dbl> <chr>  <int>
 1 15779.    0.532        50        10 RF         1
 2 14962.    0.558        50        10 RF         2
 3 14958.    0.584        50        10 RF         3
 4 15185.    0.553        50        10 RF         4
 5 14489.    0.589        50        10 RF         5
 6 16511.    0.480        50        10 GBT        1
 7 16076.    0.527        50        10 GBT        2
 8 16586.    0.452        50        10 GBT        3
 9 16570.    0.461        50        10 GBT        4
10 17226.    0.431        50        10 GBT        5
```

Tabel 2 : Metricile pentru DT

	RMSE	R_squared	max_depth
1	13640.97	0.6279198	10
2	14925.40	0.5895601	10
3	14366.03	0.5991257	10
4	14411.83	0.5979964	10
5	14987.05	0.5741229	10

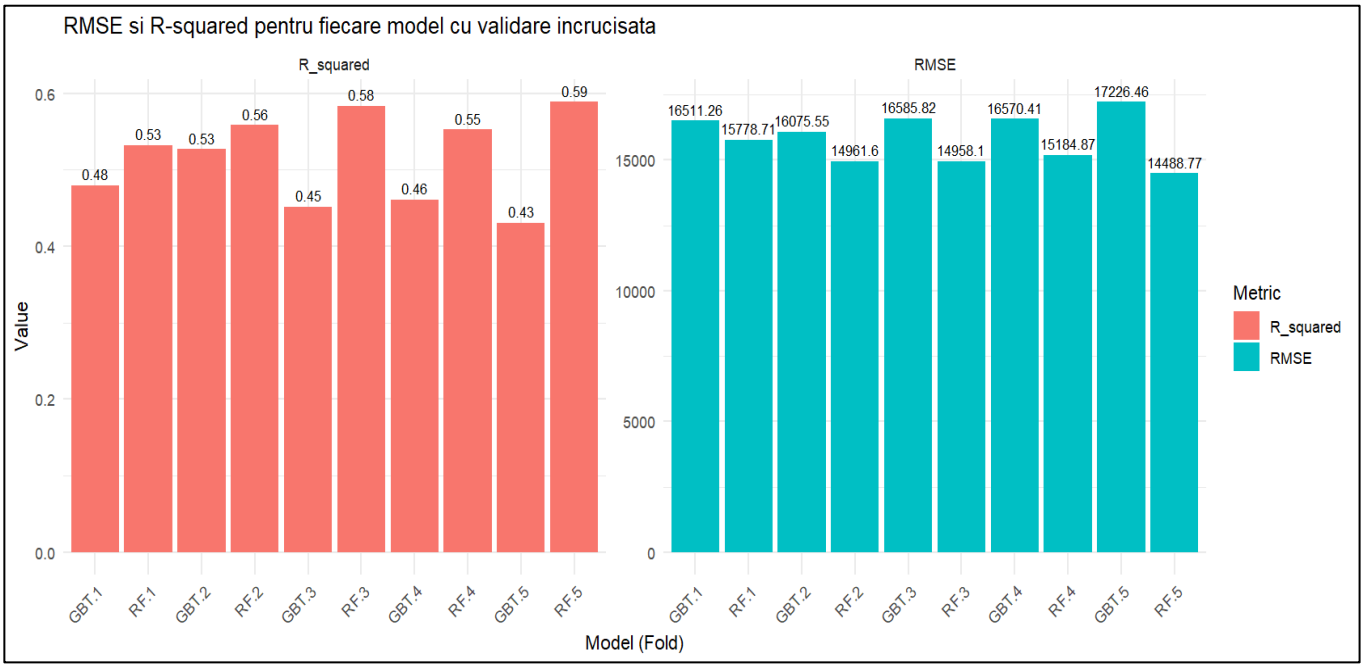
După combinarea datelor, s-a realizat o reprezentare grafică pentru a vizualiza și a interpreta mai ușor diferențele dintre modele. Graficul creat ilustrează variația metricilor RMSE și R-squared pe cele 5 folduri de validare încrucișată pentru ambele modele. Această abordare vizuală facilitează identificarea modelului care prezintă cea mai bună performanță generală și cea mai mare consistență pe diferite subseturi de date.

Analizând atât tabelele cu metricile de performanță cât și graficul putem observa următoarele lucruri:

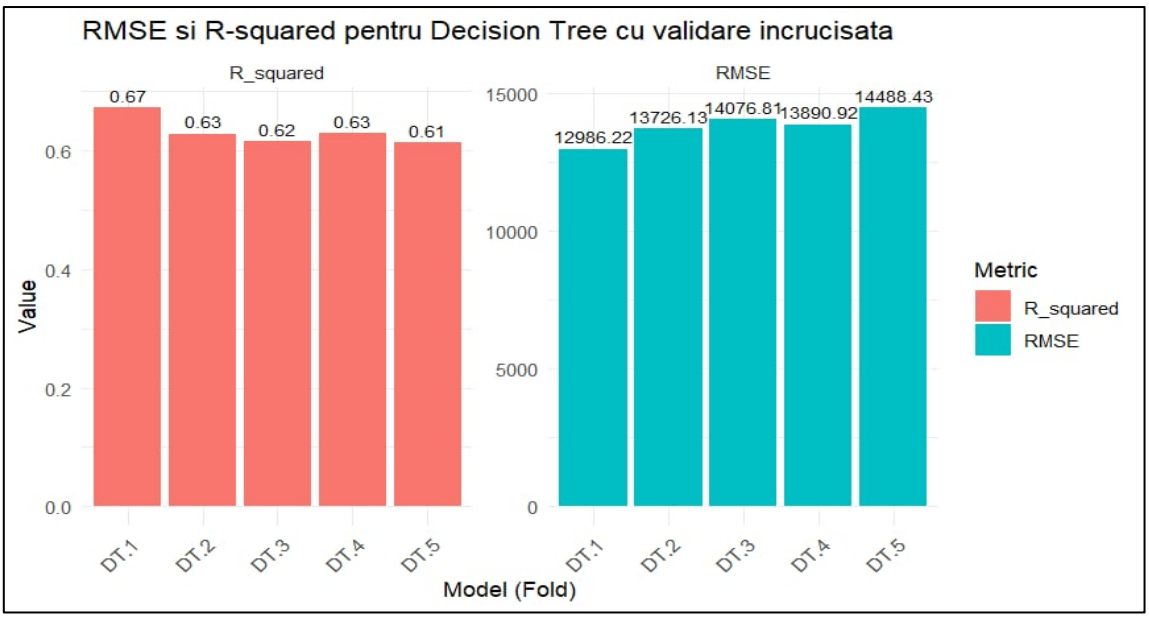
- **Modelul Random Forest (RF) :**
  - RMSE variază între **14,489** și **15,779**.
  - R-squared variază între **0.532** și **0.589**.
- **Gradient Boosted Trees (GBT):**
  - RMSE variază între **16,076** și **17,226**.
  - R-squared variază între **0.431** și **0.527**.
- **Decision Tree Regression (DT):**
  - RMSE variază între **13,640** și **14,987**.
  - R-squared variază între **0.574** și **0.627**.

Pe baza acestor metrici, modelul **Decision Tree Regression pare să aibă o performanță mai bună** decât modelul Random Forest și Gradient Boosted Trees. Acest lucru este indicat de valorile mai mici ale RMSE și valorile mai mari ale R-squared pentru Decision Tree pe toate cele 5 folduri. RMSE mai mic sugerează că erorile de predicție ale modelului RF sunt, în medie, mai mici, iar un R-squared mai mare indică o proporție mai mare a variației răspunsului care este explicată de model.

Plot comparația metricilor de performanță ale primelor două modele:



Plot comparația metricilor de performanță ale ultimului model:



Plot importanța variabilelor pentru modelele RF, GBT și DT cu validare încrucișată: