PREDICTING RETAIL STORE SALES WITH XGBOOST

Author: Manenimabasi Martin Udoh

Institution: Covenant University

Abstract:

This project focuses on forecasting weekly sales for a retail store by leveraging historical data and machine learning techniques. The dataset includes sales records from 99 departments across 45 distinct stores, along with details about holidays and promotional markdowns. These markdowns play a pivotal role in boosting sales during significant events like the Super Bowl, Christmas, and Thanksgiving.

The primary objective of this study is to develop accurate predictive models that can inform decision-making processes and offer valuable recommendations for enhancing future business operations. The project employs XGBoost, a powerful ensemble learning algorithm, and utilizes the computational capabilities of Amazon SageMaker to build and evaluate the models.

The project's findings showcase the effectiveness of XGBoost in achieving accurate sales predictions. The model achieved impressive performance with key metrics such as Root Mean Squared Error (RMSE), Mean Squared Error (MSE), and Mean Absolute Error (MAE), demonstrating its precision in forecasting sales. Furthermore, the high R-squared (R2) value of 0.9002 and adjusted R-squared (Adjusted R2) value of 0.8997 underscore the model's ability to explain a significant portion of the variance in sales.

These results have profound implications for the retail industry, offering opportunities for optimized inventory management, staffing decisions, and marketing strategies. Furthermore, the project underscores the significance of ensemble learning, regularization techniques, and cloud-based machine learning in addressing real-world sales forecasting challenges.

In conclusion, this project exemplifies the potential of data-driven decision-making in the retail sector, providing actionable insights into enhancing business processes and driving revenue growth.

Keywords: XGBoost, Sales Prediction, Comparative Analysis, Machine Learning, Regression

---

TABLE OF CONTENTS

Contact Information:

- **Email:** manenimabasi.udoh@stu.cu.edu.ng
- **LinkedIn:** manenimabasi-udoh
- **GitHub:** manenim

INTRODUCTION

Retail businesses thrive on the ability to anticipate customer demand and optimize their operations accordingly. Accurate sales forecasting plays a pivotal role in achieving this goal. In this project, we delve into the realm of retail analytics to develop predictive models for forecasting weekly sales in a retail store.

Our dataset comprises comprehensive sales records from 99 departments across 45 diverse stores. Beyond sales figures, the dataset includes vital information about holidays and promotional markdowns. These markdowns are strategically implemented to drive sales, especially during key events such as the Super Bowl, Christmas, and Thanksgiving.

The significance of this project lies in its potential to inform data-driven decision-making within the retail industry. Accurate sales predictions empower businesses to make informed choices about inventory management, staffing levels, and marketing strategies. This, in turn, can lead to increased operational efficiency and revenue generation.

Our approach combines cutting-edge machine learning techniques, specifically XGBoost, with the computational capabilities of Amazon SageMaker. Through this synergy, we aim to demonstrate the power of ensemble learning, regularization methods, and cloud-based machine learning in addressing real-world sales forecasting challenges.

In the following sections, we will delve into the project's methodology, key findings, and implications for the retail industry, showcasing how data-driven insights can drive success and enhance business operations.

IMPORTS AND SETUP

In [9]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import zipfile
```

In [10]:

```
# importing the csv files using pandas
feature = pd.read_csv('Features_data_set.csv')
sales = pd.read_csv('sales_data_set.csv')
stores = pd.read_csv('stores_data_set.csv')
```

In [11]:

```
# exploring the data
# "stores" dataframe contains information related to the 45 stores such as t

stores
```

Out[11]:

	Store	Type	Size
0	1	A	151315
1	2	A	202307
2	3	B	37392
3	4	A	205863
4	5	B	34875
5	6	A	202505
6	7	B	70713
7	8	A	155078
8	9	B	125833
9	10	B	126512
10	11	A	207499
11	12	B	112238
12	13	A	219622
13	14	A	200898
14	15	B	123737
15	16	B	57197
16	17	B	93188
17	18	B	120653
18	19	A	203819
19	20	A	203742
20	21	B	140167
21	22	B	119557
22	23	B	114533
23	24	A	203819
24	25	B	128107
25	26	A	152513
26	27	A	204184
27	28	A	206302
28	29	B	93638
29	30	C	42988
30	31	A	203750
31	32	A	203007
32	33	A	39690
33	34	A	158114
34	35	B	103681

	Store	Type	Size
35	36	A	39910
36	37	C	39910
37	38	C	39690
38	39	A	184109
39	40	A	155083
40	41	A	196321
41	42	C	39690
42	43	C	41062
43	44	C	39910
44	45	B	118221

In [12]:

```
# Features dataframe contains additional data related to the store, departme
# Store: store number
# Date: week
# Temperature: average temperature in the region
# Fuel_Price: cost of fuel in the region
# MarkDown1-5: anonymized data related to promotional markdowns.
# CPI: consumer price index
# Unemployment: unemployment rate
# IsHoliday: whether the week is a special holiday week or not

feature
```

Out[12]:

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDo
0	1	05/02/2010	42.31	2.572	NaN	NaN	NaN	
1	1	12/02/2010	38.51	2.548	NaN	NaN	NaN	
2	1	19/02/2010	39.93	2.514	NaN	NaN	NaN	
3	1	26/02/2010	46.63	2.561	NaN	NaN	NaN	
4	1	05/03/2010	46.50	2.625	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	
8185	45	28/06/2013	76.05	3.639	4842.29	975.03	3.00	244
8186	45	05/07/2013	77.50	3.614	9090.48	2268.58	582.74	579
8187	45	12/07/2013	79.37	3.614	3789.94	1827.31	85.72	74
8188	45	19/07/2013	82.84	3.737	2961.49	1047.07	204.19	36
8189	45	26/07/2013	76.06	3.804	212.02	851.73	2.06	1

8190 rows × 12 columns

In [13]:

```
# exploring the "sales" dataframe
# "Sales" dataframe contains historical sales data, which covers 2010-02-05
# Store: store number
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
# Date: the week
# Weekly_Sales: sales for the given department in the given store
# IsHoliday: whether the week is a special holiday week

sales
```

Out[13]:

	Store	Dept	Date	Weekly_Sales	IsHoliday
0	1	1	05/02/2010	24924.50	False
1	1	1	12/02/2010	46039.49	True
2	1	1	19/02/2010	41595.55	False
3	1	1	26/02/2010	19403.54	False
4	1	1	05/03/2010	21827.90	False
...	...	...	...	...	...
421565	45	98	28/09/2012	508.37	False
421566	45	98	05/10/2012	628.10	False
421567	45	98	12/10/2012	1061.02	False
421568	45	98	19/10/2012	760.01	False
421569	45	98	26/10/2012	1076.80	False

421570 rows × 5 columns

DATA EXPLORATION

In [14]:

```
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 421570 entries, 0 to 421569
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   Store         421570 non-null  int64  
 1   Dept          421570 non-null  int64  
 2   Date          421570 non-null  object 
 3   Weekly_Sales  421570 non-null  float64
 4   IsHoliday     421570 non-null  bool   
dtypes: bool(1), float64(1), int64(2), object(1)
memory usage: 13.3+ MB
```

In [15]:

```
sales.describe()
```

Out[15]:

	Store	Dept	Weekly_Sales
count	421570.000000	421570.000000	421570.000000
mean	22.200546	44.260317	15981.258123
std	12.785297	30.492054	22711.183519
min	1.000000	1.000000	-4988.940000
25%	11.000000	18.000000	2079.650000
50%	22.000000	37.000000	7612.030000
75%	33.000000	74.000000	20205.852500
max	45.000000	99.000000	693099.360000

In [16]:
```
feature.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8190 entries, 0 to 8189
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Store         8190 non-null   int64  
 1   Date          8190 non-null   object 
 2   Temperature   8190 non-null   float64
 3   Fuel_Price    8190 non-null   float64
 4   MarkDown1     4032 non-null   float64
 5   MarkDown2     2921 non-null   float64
 6   MarkDown3     3613 non-null   float64
 7   MarkDown4     3464 non-null   float64
 8   MarkDown5     4050 non-null   float64
 9   CPI           7605 non-null   float64
 10  Unemployment  7605 non-null   float64
 11  IsHoliday     8190 non-null   bool   
dtypes: bool(1), float64(9), int64(1), object(1)
memory usage: 712.0+ KB
```

In [17]:
```
feature.describe()
```

Out[17]:

	Store	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	M
count	8190.000000	8190.000000	8190.000000	4032.000000	2921.000000	3613.000000	3
mean	23.000000	59.356198	3.405992	7032.371786	3384.176594	1760.100180	3
std	12.987966	18.678607	0.431337	9262.747448	8793.583016	11276.462208	6
min	1.000000	-7.290000	2.472000	-2781.450000	-265.760000	-179.260000	
25%	12.000000	45.902500	3.041000	1577.532500	68.880000	6.600000	
50%	23.000000	60.710000	3.513000	4743.580000	364.570000	36.260000	1
75%	34.000000	73.880000	3.743000	8923.310000	2153.350000	163.150000	3
max	45.000000	101.950000	4.468000	103184.980000	104519.540000	149483.310000	67

In [18]:
```
stores.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45 entries, 0 to 44
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Store   45 non-null     int64
 1   Type    45 non-null     object
 2   Size    45 non-null     int64
dtypes: int64(2), object(1)
memory usage: 1.2+ KB
```

In [19]:
```
stores.describe()
```

Out[19]:

	Store	Size
count	45.000000	45.000000
mean	23.000000	130287.600000
std	13.133926	63825.271991
min	1.000000	34875.000000
25%	12.000000	70713.000000
50%	23.000000	126512.000000
75%	34.000000	202307.000000
max	45.000000	219622.000000

In [20]:
```
# Change the datatype of 'date' column
```

In [21]:
```
feature['Date'] = pd.to_datetime(feature['Date'], format='mixed')
sales['Date'] = pd.to_datetime(sales['Date'], format='mixed')
```

In [22]:
```
feature
```

Out[22]:

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4
0	1	2010-05-02	42.31	2.572	NaN	NaN	NaN	NaN
1	1	2010-12-02	38.51	2.548	NaN	NaN	NaN	NaN
2	1	2010-02-19	39.93	2.514	NaN	NaN	NaN	NaN
3	1	2010-02-26	46.63	2.561	NaN	NaN	NaN	NaN
4	1	2010-05-03	46.50	2.625	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...
8185	45	2013-06-28	76.05	3.639	4842.29	975.03	3.00	2449.97
8186	45	2013-05-07	77.50	3.614	9090.48	2268.58	582.74	5797.47
8187	45	2013-12-07	79.37	3.614	3789.94	1827.31	85.72	744.84
8188	45	2013-07-19	82.84	3.737	2961.49	1047.07	204.19	363.00
8189	45	2013-07-26	76.06	3.804	212.02	851.73	2.06	10.88

8190 rows × 12 columns

In [23]:

```
sales
```

Out[23]:

	Store	Dept	Date	Weekly_Sales	IsHoliday
0	1	1	2010-05-02	24924.50	False
1	1	1	2010-12-02	46039.49	True
2	1	1	2010-02-19	41595.55	False
3	1	1	2010-02-26	19403.54	False
4	1	1	2010-05-03	21827.90	False
...	...	...	...	...	...
421565	45	98	2012-09-28	508.37	False
421566	45	98	2012-05-10	628.10	False
421567	45	98	2012-12-10	1061.02	False
421568	45	98	2012-10-19	760.01	False
421569	45	98	2012-10-26	1076.80	False

421570 rows × 5 columns

MERGING DATASETS

In [24]:
```
sales.head()
```

Out[24]:

	Store	Dept	Date	Weekly_Sales	IsHoliday
0	1	1	2010-05-02	24924.50	False
1	1	1	2010-12-02	46039.49	True
2	1	1	2010-02-19	41595.55	False
3	1	1	2010-02-26	19403.54	False
4	1	1	2010-05-03	21827.90	False

In [25]:
```
feature.head()
```

Out[25]:

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	Ma
0	1	2010-05-02	42.31	2.572	NaN	NaN	NaN	NaN	
1	1	2010-12-02	38.51	2.548	NaN	NaN	NaN	NaN	
2	1	2010-02-19	39.93	2.514	NaN	NaN	NaN	NaN	
3	1	2010-02-26	46.63	2.561	NaN	NaN	NaN	NaN	
4	1	2010-05-03	46.50	2.625	NaN	NaN	NaN	NaN	

In [26]:
```
df = pd.merge(sales, feature, on = ['Store','Date','IsHoliday'])
```

In [27]:
```
df
```

Out[27]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	Marl
0	1	1	2010-05-02	24924.50	False	42.31	2.572	NaN	
1	1	2	2010-05-02	50605.27	False	42.31	2.572	NaN	
2	1	3	2010-05-02	13740.12	False	42.31	2.572	NaN	
3	1	4	2010-05-02	39954.04	False	42.31	2.572	NaN	
4	1	5	2010-05-02	32229.38	False	42.31	2.572	NaN	
...	...	...	...	...	...	...	...	...	
421565	45	93	2012-10-26	2487.80	False	58.85	3.882	4018.91	
421566	45	94	2012-10-26	5203.31	False	58.85	3.882	4018.91	
421567	45	95	2012-10-26	56017.47	False	58.85	3.882	4018.91	
421568	45	97	2012-10-26	6817.48	False	58.85	3.882	4018.91	
421569	45	98	2012-10-26	1076.80	False	58.85	3.882	4018.91	

421570 rows × 14 columns

In [28]:

```
df.head()
```

Out[28]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDowr
0	1	1	2010-05-02	24924.50	False	42.31	2.572	NaN	Na
1	1	2	2010-05-02	50605.27	False	42.31	2.572	NaN	Na
2	1	3	2010-05-02	13740.12	False	42.31	2.572	NaN	Na
3	1	4	2010-05-02	39954.04	False	42.31	2.572	NaN	Na
4	1	5	2010-05-02	32229.38	False	42.31	2.572	NaN	Na

In [29]:

```
stores.head()
```

Out[29]:

	Store	Type	Size
0	1	A	151315
1	2	A	202307
2	3	B	37392
3	4	A	205863
4	5	B	34875

In [30]:
```
df = pd.merge(df, stores, on = ['Store'], how = 'left')
```

In [31]:
```
df.head()
```

Out[31]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDowr
0	1	1	2010-05-02	24924.50	False	42.31	2.572	NaN	Na
1	1	2	2010-05-02	50605.27	False	42.31	2.572	NaN	Na
2	1	3	2010-05-02	13740.12	False	42.31	2.572	NaN	Na
3	1	4	2010-05-02	39954.04	False	42.31	2.572	NaN	Na
4	1	5	2010-05-02	32229.38	False	42.31	2.572	NaN	Na

- Define a function to extract the month information from the dataframe column "Date"
- Apply the function to the entire column "Date" in the merged dataframe "df" and write the output in a column entitled "month"

In [32]:
```
def get_month(x):
    return int(str(x).split('-')[1])
```

In [33]:
```
df['Month'] = df['Date'].apply(get_month)
```

In [34]:
```
df
```

Out[34]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	Marl
0	1	1	2010-05-02	24924.50	False	42.31	2.572	NaN	
1	1	2	2010-05-02	50605.27	False	42.31	2.572	NaN	
2	1	3	2010-05-02	13740.12	False	42.31	2.572	NaN	
3	1	4	2010-05-02	39954.04	False	42.31	2.572	NaN	
4	1	5	2010-05-02	32229.38	False	42.31	2.572	NaN	
...	...	...	...	...	...	...	...	...	
421565	45	93	2012-10-26	2487.80	False	58.85	3.882	4018.91	
421566	45	94	2012-10-26	5203.31	False	58.85	3.882	4018.91	
421567	45	95	2012-10-26	56017.47	False	58.85	3.882	4018.91	
421568	45	97	2012-10-26	6817.48	False	58.85	3.882	4018.91	
421569	45	98	2012-10-26	1076.80	False	58.85	3.882	4018.91	

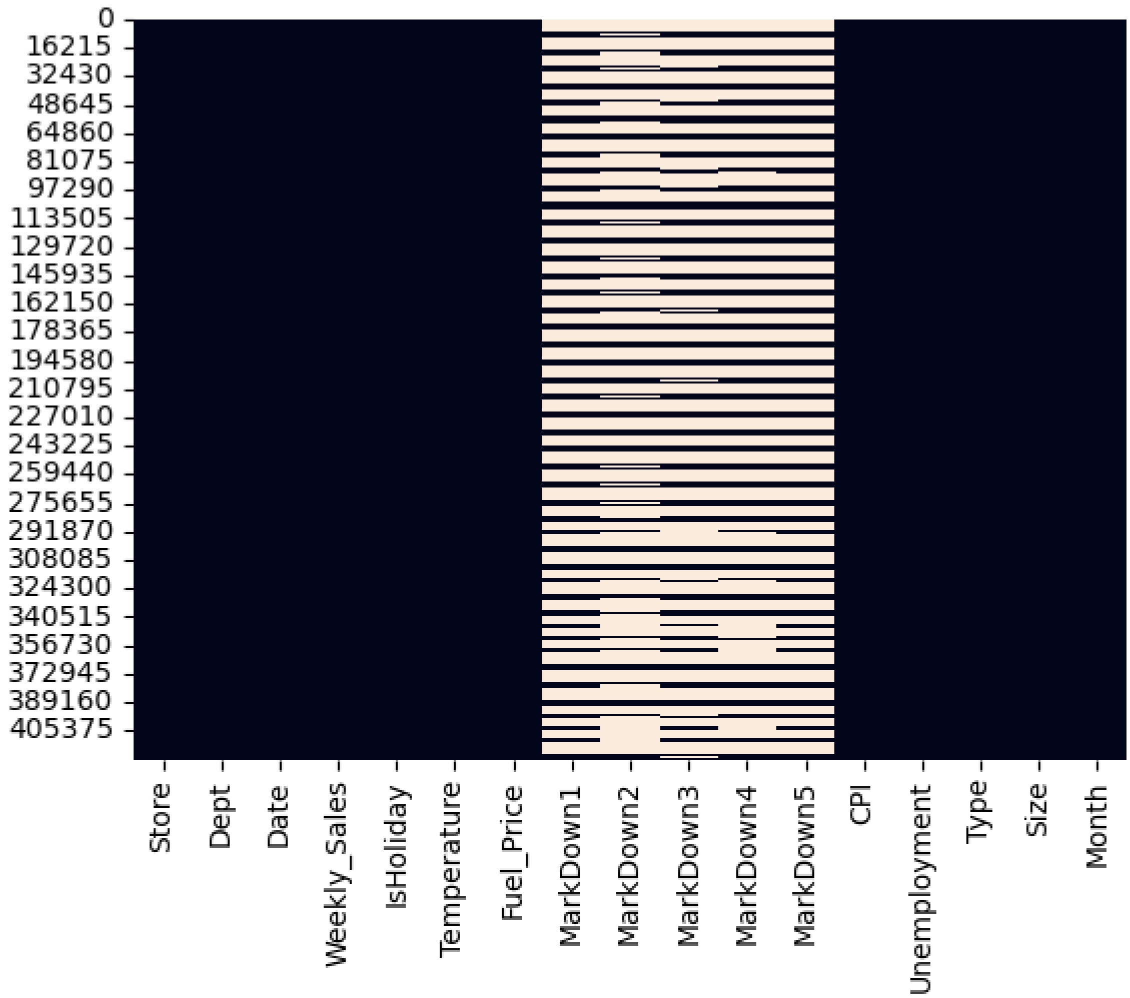
421570 rows × 17 columns

EXPLORING MERGED DATASET

In [35]:

```
sns.heatmap(df.isnull(), cbar = False)
```

Out[35]: <Axes: >



In [36]:
```
# check the number of non-null values in the dataframe
df.isnull().sum()
```

Out[36]:
```
Store                0
Dept                 0
Date                 0
Weekly_Sales         0
IsHoliday            0
Temperature          0
Fuel_Price           0
MarkDown1       270889
MarkDown2       310322
MarkDown3       284479
MarkDown4       286603
MarkDown5       270138
CPI                  0
Unemployment         0
Type                 0
Size                 0
Month                0
dtype: int64
```

In [37]:
```
# Fill up NaN elements with zeros
df = df.fillna(0)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [38]:
```
df
```

Out[38]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	Marl
0	1	1	2010-05-02	24924.50	False	42.31	2.572	0.00	
1	1	2	2010-05-02	50605.27	False	42.31	2.572	0.00	
2	1	3	2010-05-02	13740.12	False	42.31	2.572	0.00	
3	1	4	2010-05-02	39954.04	False	42.31	2.572	0.00	
4	1	5	2010-05-02	32229.38	False	42.31	2.572	0.00	
...	...	...	...	...	...	...	...	...	
421565	45	93	2012-10-26	2487.80	False	58.85	3.882	4018.91	
421566	45	94	2012-10-26	5203.31	False	58.85	3.882	4018.91	
421567	45	95	2012-10-26	56017.47	False	58.85	3.882	4018.91	
421568	45	97	2012-10-26	6817.48	False	58.85	3.882	4018.91	
421569	45	98	2012-10-26	1076.80	False	58.85	3.882	4018.91	

421570 rows × 17 columns

In [39]:
```
# Statistical summary of the combined dataframe
df.describe()
```

Out[39]:

	Store	Dept	Date	Weekly_Sales	Temperature	Fue
count	421570.000000	421570.000000	421570	421570.000000	421570.000000	421570.
mean	22.200546	44.260317	2011-06-19 05:35:51.733757440	15981.258123	60.090059	3.
min	1.000000	1.000000	2010-01-10 00:00:00	-4988.940000	-2.060000	2.
25%	11.000000	18.000000	2010-10-12 00:00:00	2079.650000	46.680000	2.
50%	22.000000	37.000000	2011-06-17 00:00:00	7612.030000	62.090000	3.
75%	33.000000	74.000000	2012-03-02 00:00:00	20205.852500	74.280000	3.
max	45.000000	99.000000	2012-12-10 00:00:00	693099.360000	100.140000	4.
std	12.785297	30.492054	NaN	22711.183519	18.447931	0.

In [40]:
```python
# check the number of duplicated entries in the dataframe
df.duplicated().sum()
```

Out[40]:
```
0
```

In [41]:
```python
df['Type'].value_counts()
```

Out[41]:
```
Type
A    215478
B    163495
C     42597
Name: count, dtype: int64
```

In [ ]:
```python
# Replace the "IsHoliday" with ones and zeros instead of True and False (cha
df['IsHoliday'] = df['IsHoliday'].replace({True:1, False:0})
```

In [43]:
```python
df
```

Out[43]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	Marl
0	1	1	2010-05-02	24924.50	0	42.31	2.572	0.00	
1	1	2	2010-05-02	50605.27	0	42.31	2.572	0.00	
2	1	3	2010-05-02	13740.12	0	42.31	2.572	0.00	
3	1	4	2010-05-02	39954.04	0	42.31	2.572	0.00	
4	1	5	2010-05-02	32229.38	0	42.31	2.572	0.00	
...	...	...	...	...	...	...	...	...	
421565	45	93	2012-10-26	2487.80	0	58.85	3.882	4018.91	
421566	45	94	2012-10-26	5203.31	0	58.85	3.882	4018.91	
421567	45	95	2012-10-26	56017.47	0	58.85	3.882	4018.91	
421568	45	97	2012-10-26	6817.48	0	58.85	3.882	4018.91	
421569	45	98	2012-10-26	1076.80	0	58.85	3.882	4018.91	

421570 rows × 17 columns

EXPLORATORY DATA ANALYSIS

In [44]:
```python
# Creating a pivot table to understand the relationship in the data
```

```
result = pd.pivot_table(df, values = 'Weekly_Sales', columns = ['Type'], inc
                     aggfunc= np.mean)
```

In [45]:
```
result
```

Out[45]:

		Type	A	B	C
Date	**Store**	**Dept**			
2010-01-10	**1**	**1**	20094.19	NaN	NaN
		2	45829.02	NaN	NaN
		3	9775.17	NaN	NaN
		4	34912.45	NaN	NaN
		5	23381.38	NaN	NaN
...	**...**	**...**	...	...	...
2012-12-10	**45**	**93**	NaN	2644.24	NaN
		94	NaN	4041.28	NaN
		95	NaN	49334.77	NaN
		97	NaN	6463.32	NaN
		98	NaN	1061.02	NaN

421570 rows × 3 columns

In [46]:
```
result.describe()
# It can be seen that Type A stores have much higher sales than Type B and T
```

Out[46]:

Type	A	B	C
count	215478.000000	163495.000000	42597.000000
mean	20099.568043	12237.075977	9519.532538
std	26423.457227	17203.668989	15985.351612
min	-4988.940000	-3924.000000	-379.000000
25%	3315.090000	1927.055000	131.990000
50%	10105.170000	6187.870000	1149.670000
75%	26357.180000	15353.740000	12695.010000
max	474330.100000	693099.360000	112152.350000

In [47]:
```
# creating another pivot table to check the impact of holidays on markdown1
result_md = pd.pivot_table(df, values = ['MarkDown1','MarkDown2','MarkDown3'
                    aggfunc={'MarkDown1' : np.mean,'MarkDown2' : np.mean, 'M
```

In [48]:
```
result_md
```

Out[48]:

		IsHoliday	MarkDown1 0	MarkDown1 1	MarkDown2 0	MarkDown2 1	MarkDown3 0	MarkDown3 1	MarkDown4 0	MarkDown4 1	MarkDown5 0	MarkDown5 1
Date	**Store**	**Dept**										
2010-01-10	**1**	**1**	0.00	NaN	0.0	NaN	0.00	NaN	0.00	NaN	0.00	NaN
		2	0.00	NaN	0.0	NaN	0.00	NaN	0.00	NaN	0.00	NaN
		3	0.00	NaN	0.0	NaN	0.00	NaN	0.00	NaN	0.00	NaN
		4	0.00	NaN	0.0	NaN	0.00	NaN	0.00	NaN	0.00	NaN
		5	0.00	NaN	0.0	NaN	0.00	NaN	0.00	NaN	0.00	NaN
...	**...**	**...**	...	...	...	...	...	...	...	...	...	...
2012-12-10	**45**	**93**	1956.28	NaN	0.0	NaN	7.89	NaN	599.32	NaN	3990.54	NaN
		94	1956.28	NaN	0.0	NaN	7.89	NaN	599.32	NaN	3990.54	NaN
		95	1956.28	NaN	0.0	NaN	7.89	NaN	599.32	NaN	3990.54	NaN
		97	1956.28	NaN	0.0	NaN	7.89	NaN	599.32	NaN	3990.54	NaN
		98	1956.28	NaN	0.0	NaN	7.89	NaN	599.32	NaN	3990.54	NaN

421570 rows × 10 columns

In [49]:

```
result_md.sum()
```

Out[49]:

```
           IsHoliday
MarkDown1  0            1.017371e+09
           1            7.452684e+07
MarkDown2  0            2.310619e+08
           1            1.399088e+08
MarkDown3  0            2.460332e+07
           1            1.727284e+08
MarkDown4  0            4.196331e+08
           1            3.698298e+07
MarkDown5  0            6.585670e+08
           1            4.240793e+07
dtype: float64
```

In [50]:

```
result_md.describe()
# we can conclude that MarkDown2 and MarkDown3 have higher volume on holiday
# while other MarkDowns don't show significant changes relating to holiday.
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Out[50]:

	MarkDown1		MarkDown2		MarkDc
IsHoliday	0	1	0	1	0
count	391909.000000	29661.000000	391909.000000	29661.000000	391909.000000	29661.00
mean	2595.936803	2512.620778	589.580546	4716.929394	62.778142	5823.41
std	6123.402037	5020.047408	2984.163111	15295.329993	630.704594	19959.30
min	0.000000	0.000000	-265.760000	-9.980000	-29.100000	0.00
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
75%	2826.570000	2463.160000	0.500000	65.000000	3.840000	66.08
max	88646.760000	36778.650000	45971.430000	104519.540000	25959.980000	141630.61

In [51]:
```
df
```

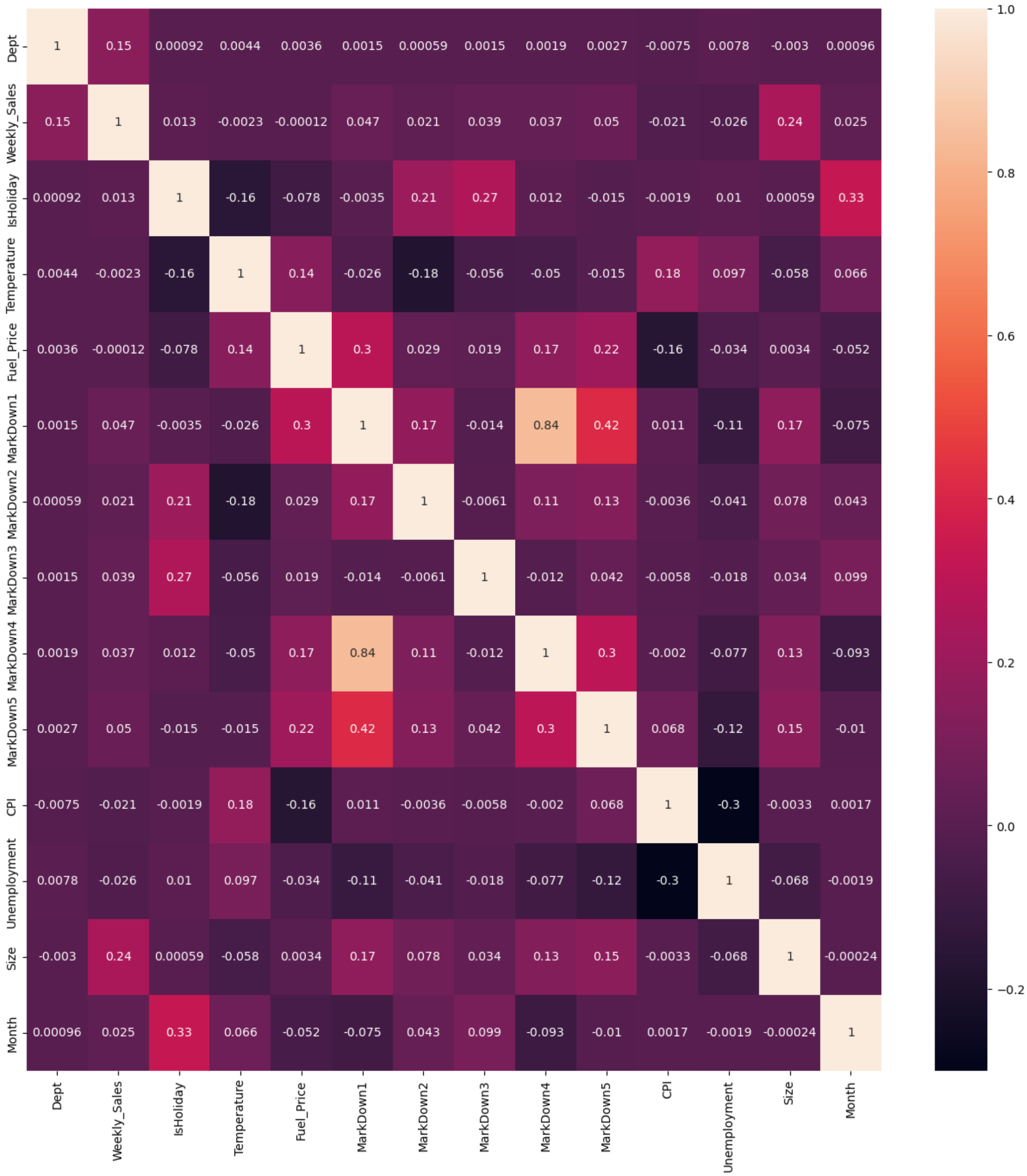
Out[51]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	Marl
0	1	1	2010-05-02	24924.50	0	42.31	2.572	0.00	
1	1	2	2010-05-02	50605.27	0	42.31	2.572	0.00	
2	1	3	2010-05-02	13740.12	0	42.31	2.572	0.00	
3	1	4	2010-05-02	39954.04	0	42.31	2.572	0.00	
4	1	5	2010-05-02	32229.38	0	42.31	2.572	0.00	
...	...	...	...	...	...	...	...	...	
421565	45	93	2012-10-26	2487.80	0	58.85	3.882	4018.91	
421566	45	94	2012-10-26	5203.31	0	58.85	3.882	4018.91	
421567	45	95	2012-10-26	56017.47	0	58.85	3.882	4018.91	
421568	45	97	2012-10-26	6817.48	0	58.85	3.882	4018.91	
421569	45	98	2012-10-26	1076.80	0	58.85	3.882	4018.91	

421570 rows × 17 columns

In [52]:
```
numeric_columns = df.select_dtypes(include=['number'])
corr_matrix = numeric_columns.drop(columns = ['Store']).corr()
```

In [53]:
```
plt.figure(figsize = (16,16))
sns.heatmap(corr_matrix, annot = True)
plt.show()
```



DATA VISUALIZATION

In [54]:
```
df
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Out[54]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	Marl
0	1	1	2010-05-02	24924.50	0	42.31	2.572	0.00	
1	1	2	2010-05-02	50605.27	0	42.31	2.572	0.00	
2	1	3	2010-05-02	13740.12	0	42.31	2.572	0.00	
3	1	4	2010-05-02	39954.04	0	42.31	2.572	0.00	
4	1	5	2010-05-02	32229.38	0	42.31	2.572	0.00	
...	...	...	...	...	...	...	...	...	
421565	45	93	2012-10-26	2487.80	0	58.85	3.882	4018.91	
421566	45	94	2012-10-26	5203.31	0	58.85	3.882	4018.91	
421567	45	95	2012-10-26	56017.47	0	58.85	3.882	4018.91	
421568	45	97	2012-10-26	6817.48	0	58.85	3.882	4018.91	
421569	45	98	2012-10-26	1076.80	0	58.85	3.882	4018.91	

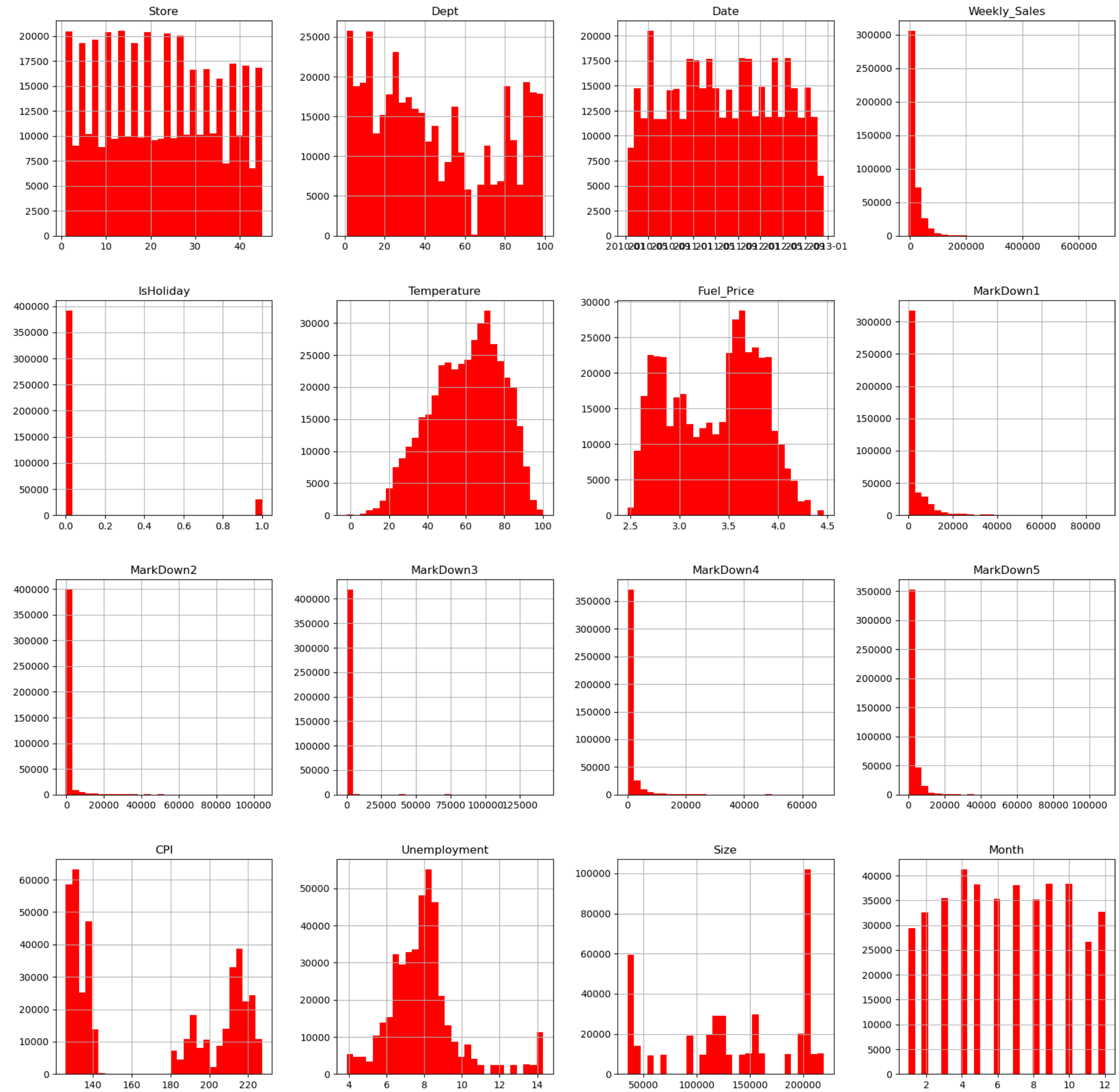
421570 rows × 17 columns

In [55]:

```
df.hist(bins = 30, figsize = (20,20), color = 'r')
```

Out[55]:

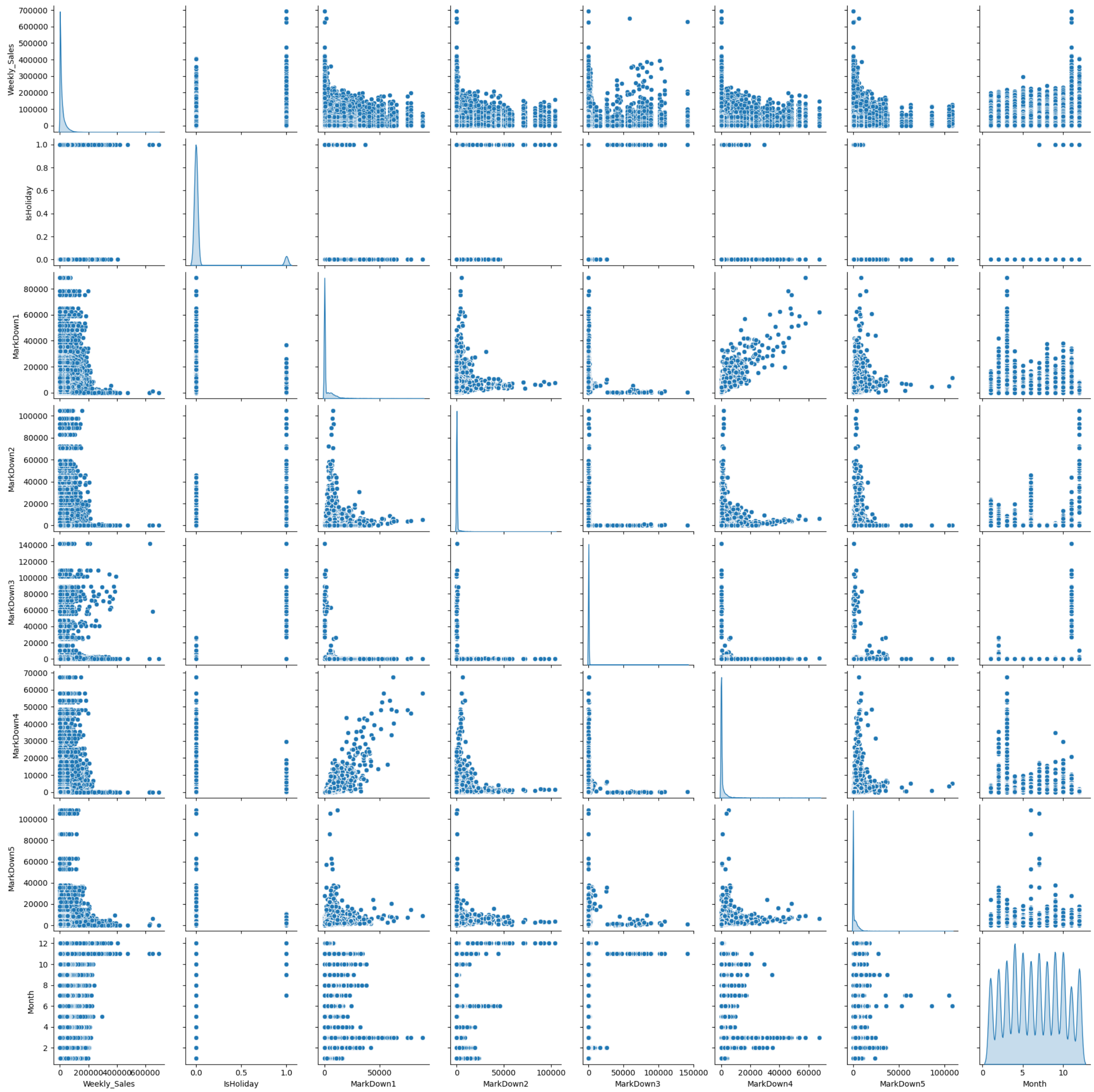
```
array([[<Axes: title={'center': 'Store'}>,
        <Axes: title={'center': 'Dept'}>,
        <Axes: title={'center': 'Date'}>,
        <Axes: title={'center': 'Weekly_Sales'}>],
       [<Axes: title={'center': 'IsHoliday'}>,
        <Axes: title={'center': 'Temperature'}>,
        <Axes: title={'center': 'Fuel_Price'}>,
        <Axes: title={'center': 'MarkDown1'}>],
       [<Axes: title={'center': 'MarkDown2'}>,
        <Axes: title={'center': 'MarkDown3'}>,
        <Axes: title={'center': 'MarkDown4'}>,
        <Axes: title={'center': 'MarkDown5'}>],
       [<Axes: title={'center': 'CPI'}>,
        <Axes: title={'center': 'Unemployment'}>,
        <Axes: title={'center': 'Size'}>,
        <Axes: title={'center': 'Month'}>]], dtype=object)
```



In [56]:
```
# visualizing the relationship using pairplots
# there is a relationship between markdown #1 and Markdown #4
# holiday and sales
# Weekly sales and markdown #3
sns.pairplot(df[["Weekly_Sales","IsHoliday","MarkDown1","MarkDown2","MarkDow
```

```
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/seaborn/
axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[56]: <seaborn.axisgrid.PairGrid at 0x7ff0a80060b0>



In [57]:
```
df_type = df.groupby('Type').mean()
```

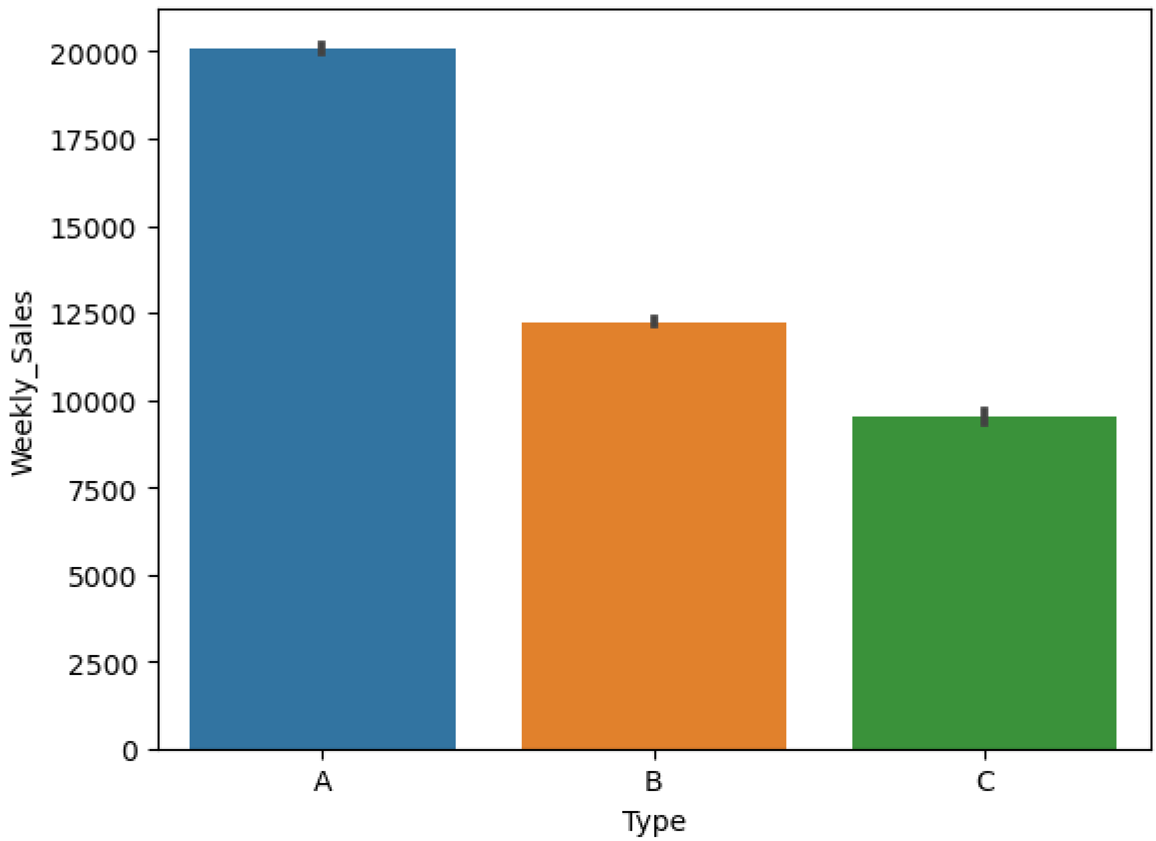
In [58]:
```
df_type
```

Out[58]:

Type	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Pric
A	21.736419	44.622156	2011-06-18 13:48:22.514409984	20099.568043	0.070471	60.531945	3.34399
B	18.450417	43.112273	2011-06-18 22:58:32.430349568	12237.075977	0.070412	57.562951	3.38252
C	38.942015	46.836350	2011-06-23 14:53:44.508768256	9519.532538	0.069582	67.554266	3.36465

In [59]:
```
sns.barplot(x = df['Type'], y = df['Weekly_Sales'], data = df)
```

Out[59]: <Axes: xlabel='Type', ylabel='Weekly_Sales'>



In [60]:

```python
# df_dept = df.drop(columns = ['Store','Type','IsHoliday','Temperature','Fue
numeric_columns = df.select_dtypes(include=['number'])
df_dept = numeric_columns.groupby('Dept').mean()
df_dept
```

Out[60]:

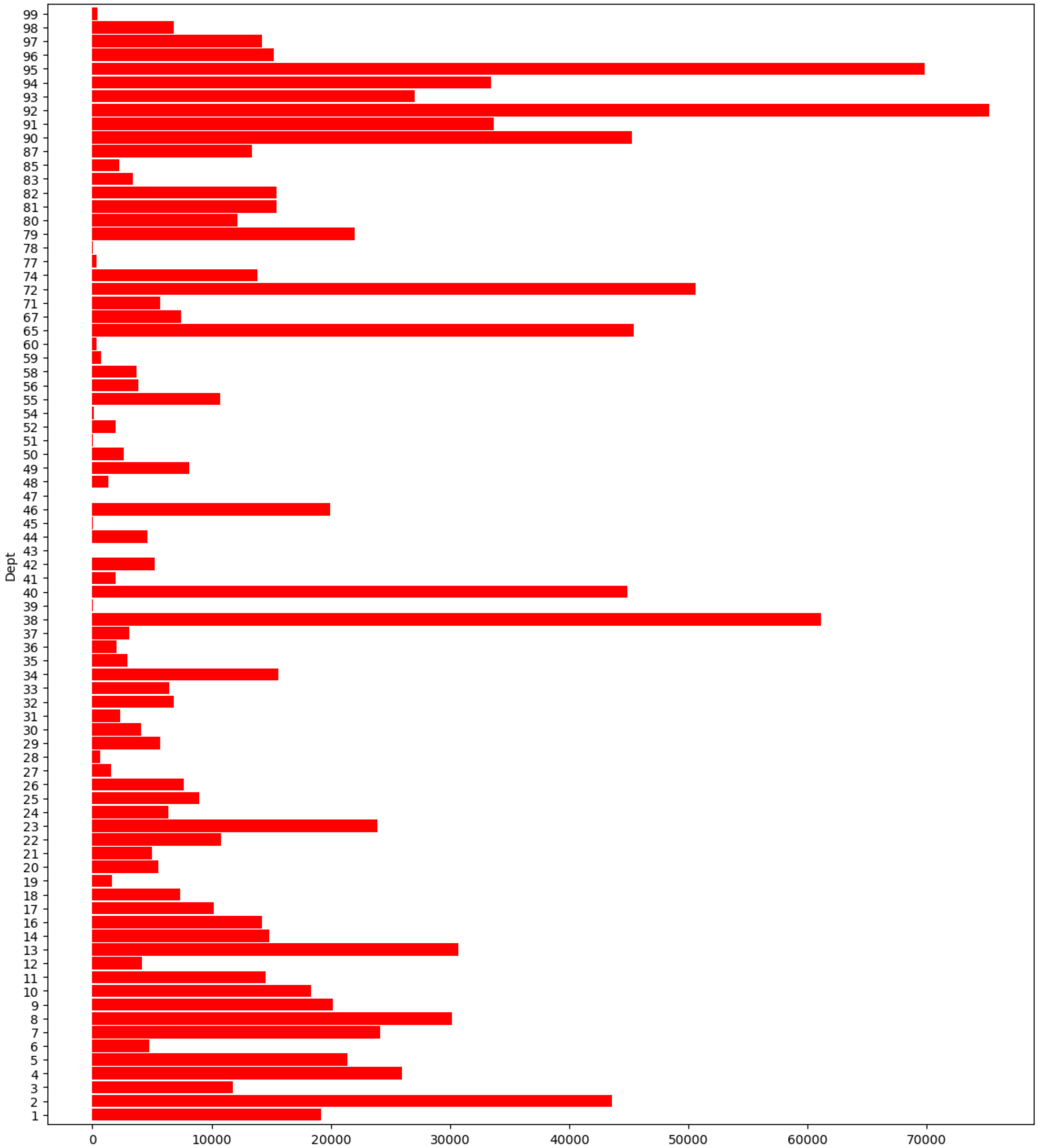
Dept	Store	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2
1	23.000000	19213.485088	0.069930	60.663782	3.358607	2429.019322	818.872810
2	23.000000	43607.020113	0.069930	60.663782	3.358607	2429.019322	818.872810
3	23.000000	11793.698516	0.069930	60.663782	3.358607	2429.019322	818.872810
4	23.000000	25974.630238	0.069930	60.663782	3.358607	2429.019322	818.872810
5	22.757366	21365.583515	0.069797	60.559367	3.365397	2462.697233	830.226332
...	...	...	...	...	...	...	...
95	23.000000	69824.423080	0.069930	60.663782	3.358607	2429.019322	818.872810
96	23.258138	15210.942761	0.069839	61.539285	3.359920	2362.845647	820.762363
97	23.357439	14255.576919	0.069767	60.490781	3.362418	2463.638764	833.096524
98	24.173920	6824.694889	0.071967	60.115942	3.372656	2569.994716	882.483088
99	21.438515	415.487065	0.110209	62.813596	3.592702	7741.403376	2164.573063

81 rows × 14 columns

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [61]:
```
fig = plt.figure(figsize = (14,16))
df_dept['Weekly_Sales'].plot(kind = 'barh', color = 'r', width = 0.9)
```

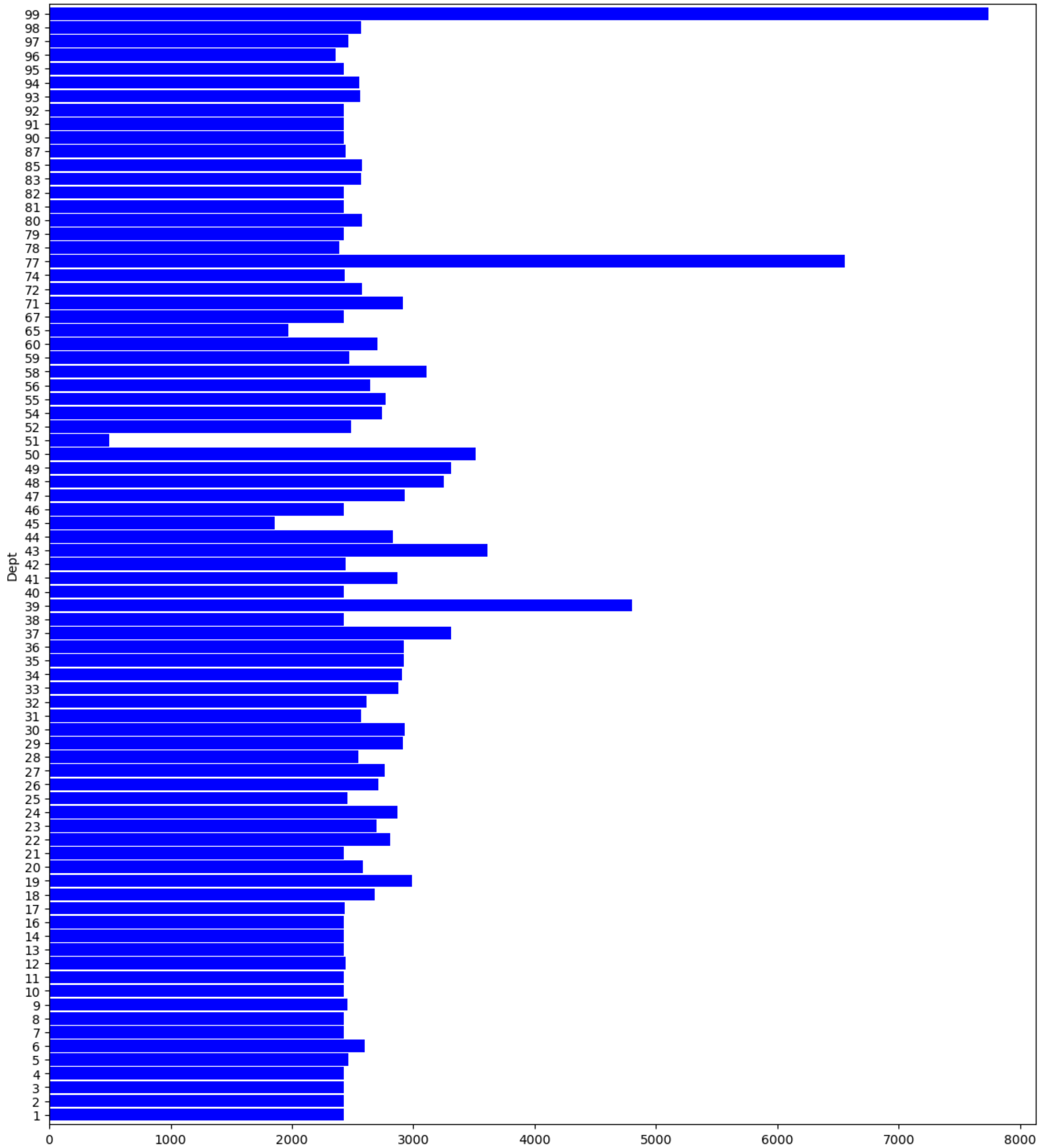
Out[61]: <Axes: ylabel='Dept'>



In [62]:
```
# checking relationship between markdown and sales
fig = plt.figure(figsize = (14,16))
df_dept['MarkDown1'].plot(kind = 'barh', color = 'blue', width = 0.9)
```

Out[62]: <Axes: ylabel='Dept'>

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js




```
In [63]: fig = plt.figure(figsize = (14,16))

         df_dept['MarkDown2'].plot(kind = 'barh', color = 'yellow', width = 0.9)
```

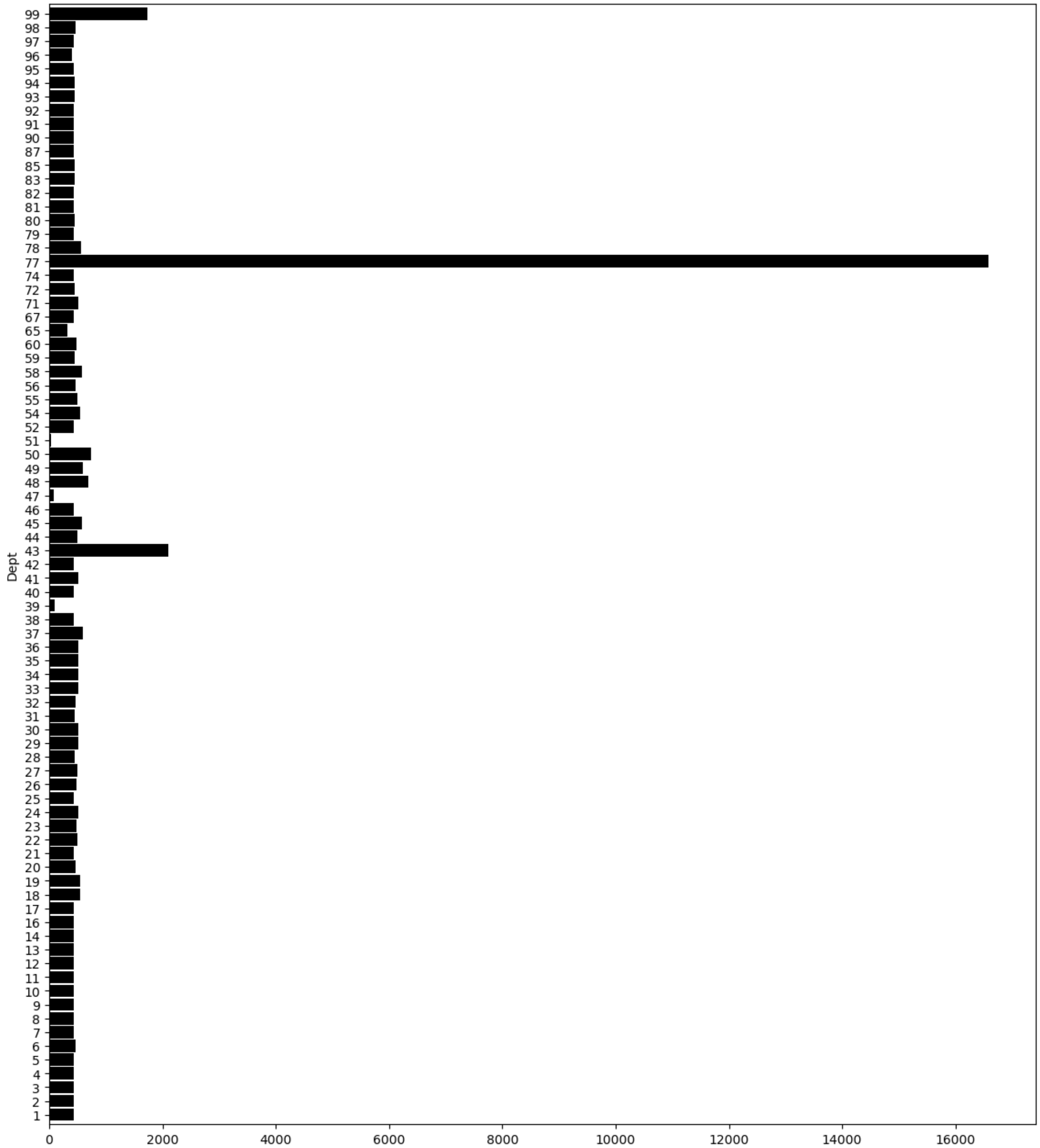
Out[63]: <Axes: ylabel='Dept'>

In [64]:

```
fig = plt.figure(figsize = (14,16))

df_dept['MarkDown3'].plot(kind = 'barh', color = 'black', width = 0.9)
```

Out[64]: <Axes: ylabel='Dept'>



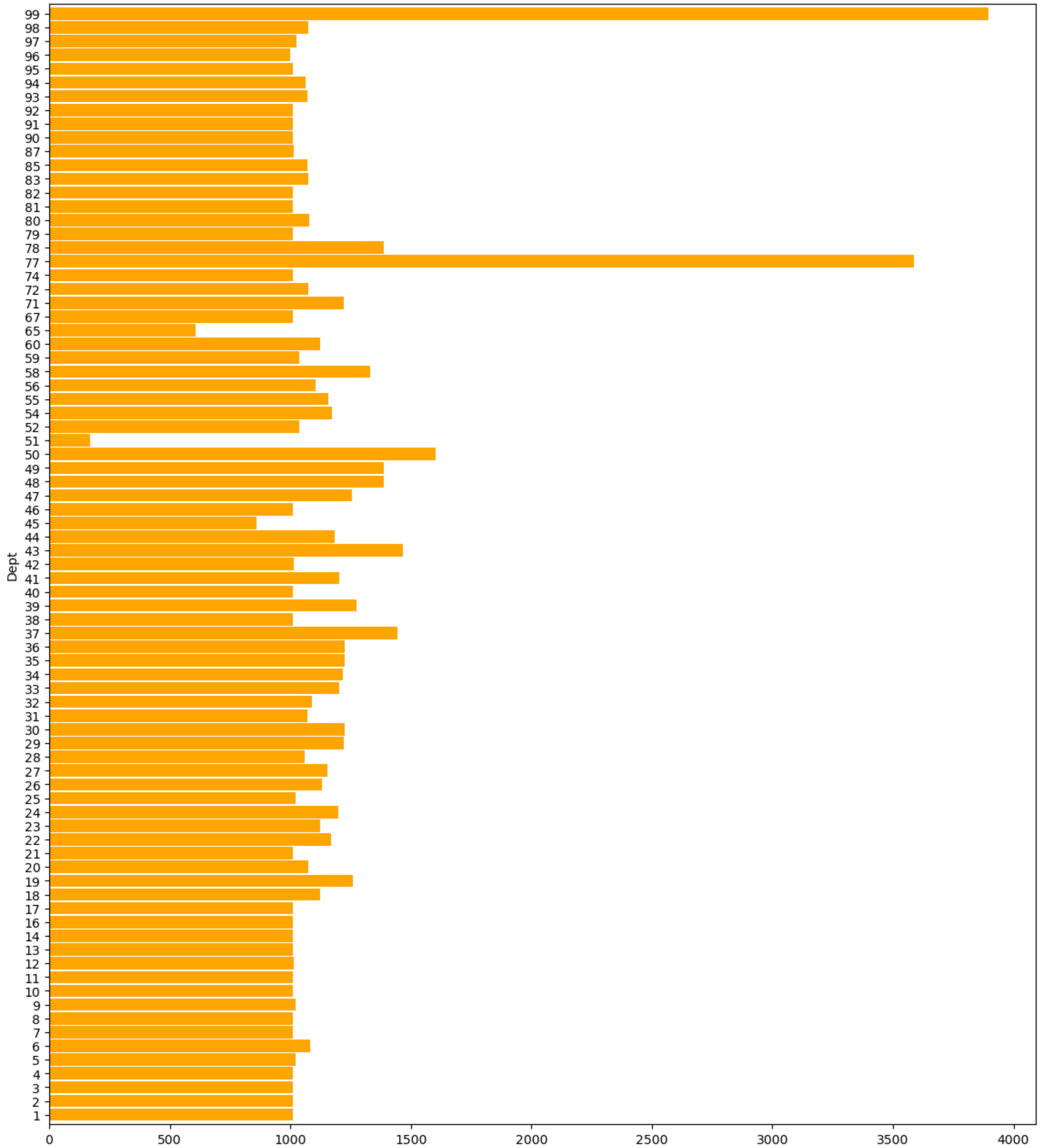
In [65]:

```
fig = plt.figure(figsize = (14,16))

df_dept['MarkDown4'].plot(kind = 'barh', color = 'orange', width = 0.9)
```

Out[65]: <Axes: ylabel='Dept'>

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

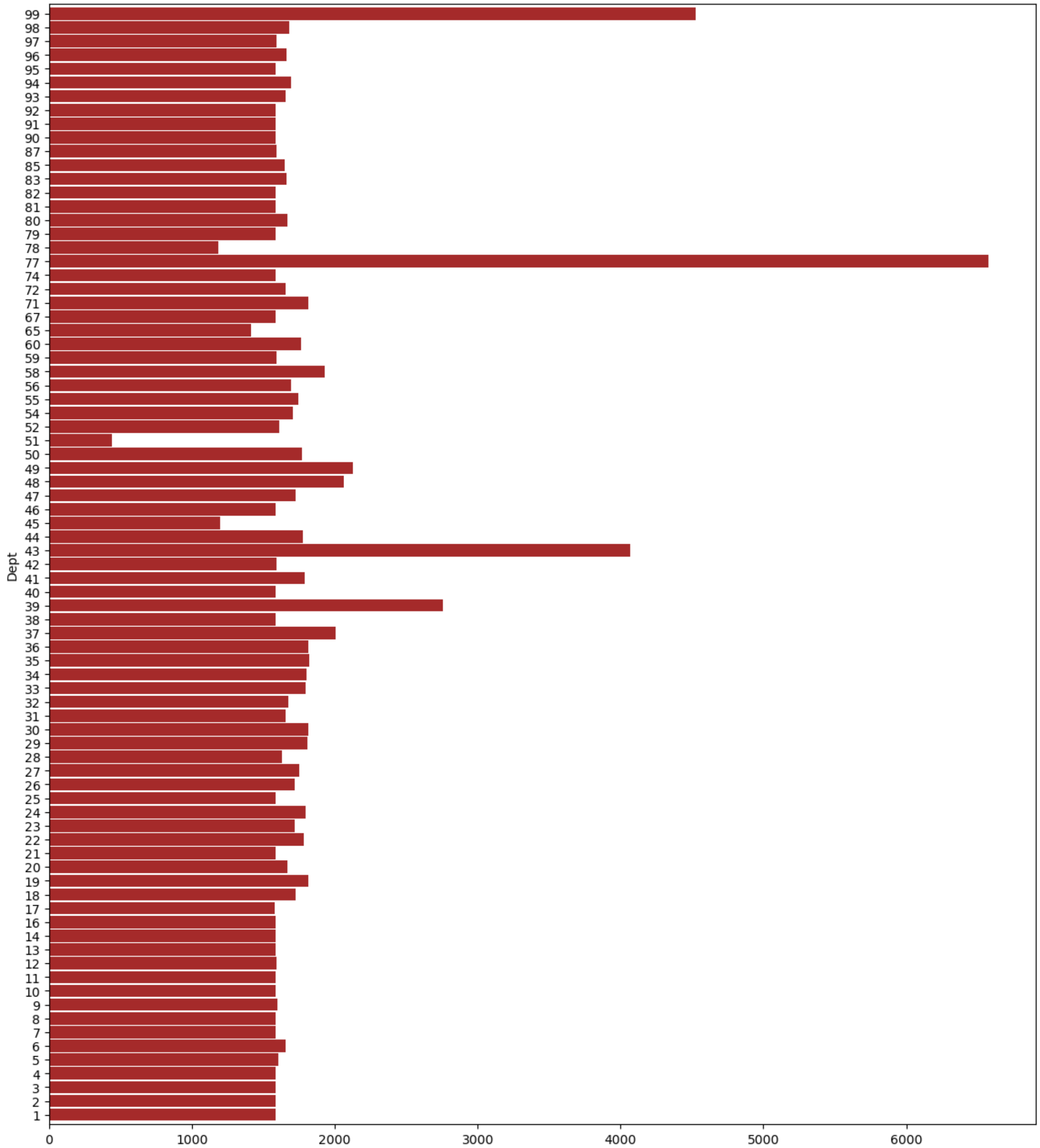


In [66]:
```
fig = plt.figure(figsize = (14,16))

df_dept['MarkDown5'].plot(kind = 'barh', color = 'brown', width = 0.9)
```

Out[66]: <Axes: ylabel='Dept'>

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js




```
In [67]: df
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Out[67]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	Marl
0	1	1	2010-05-02	24924.50	0	42.31	2.572	0.00	
1	1	2	2010-05-02	50605.27	0	42.31	2.572	0.00	
2	1	3	2010-05-02	13740.12	0	42.31	2.572	0.00	
3	1	4	2010-05-02	39954.04	0	42.31	2.572	0.00	
4	1	5	2010-05-02	32229.38	0	42.31	2.572	0.00	
...	...	...	...	...	...	...	...	...	
421565	45	93	2012-10-26	2487.80	0	58.85	3.882	4018.91	
421566	45	94	2012-10-26	5203.31	0	58.85	3.882	4018.91	
421567	45	95	2012-10-26	56017.47	0	58.85	3.882	4018.91	
421568	45	97	2012-10-26	6817.48	0	58.85	3.882	4018.91	
421569	45	98	2012-10-26	1076.80	0	58.85	3.882	4018.91	

421570 rows × 17 columns

CONCLUSIONS FROM THE DATA VISUALIZATION

- We can conclude that departments that have poor weekly sales have been assigned high number of markdowns.
- Example: check out store 77 and 99

In [68]:

```
# Sort by weekly sales
df_dept_sale = df_dept.sort_values(by = ['Weekly_Sales'], ascending = True)
df_dept_sale['Weekly_Sales'][:30]
```

Out[68]:
```
Dept
47      -7.682554
43       1.193333
78       7.296638
39      11.123750
51      21.931729
45      23.211586
54     108.305985
77     328.961800
60     347.370229
99     415.487065
28     618.085116
59     694.463564
48    1344.893576
27    1583.437727
19    1654.815030
52    1928.356252
41    1965.559998
36    2022.571061
85    2264.359407
31    2339.440287
50    2658.897010
35    2921.044946
37    3111.076193
83    3383.349838
58    3702.907419
56    3833.706211
30    4118.197208
12    4175.397021
44    4651.729658
6     4747.856188
Name: Weekly_Sales, dtype: float64
```

DATA PREPROCESSING

In [69]:
```
df
```

Out[69]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	Marl
0	1	1	2010-05-02	24924.50	0	42.31	2.572	0.00	
1	1	2	2010-05-02	50605.27	0	42.31	2.572	0.00	
2	1	3	2010-05-02	13740.12	0	42.31	2.572	0.00	
3	1	4	2010-05-02	39954.04	0	42.31	2.572	0.00	
4	1	5	2010-05-02	32229.38	0	42.31	2.572	0.00	
...	...	...	...	...	...	...	...	...	
421565	45	93	2012-10-26	2487.80	0	58.85	3.882	4018.91	
421566	45	94	2012-10-26	5203.31	0	58.85	3.882	4018.91	
421567	45	95	2012-10-26	56017.47	0	58.85	3.882	4018.91	
421568	45	97	2012-10-26	6817.48	0	58.85	3.882	4018.91	
421569	45	98	2012-10-26	1076.80	0	58.85	3.882	4018.91	

421570 rows × 17 columns

In [70]:

```
df['Dept'].dtype
```

Out[70]:

```
dtype('int64')
```

In [71]:

```
# Drop the date
df_target = df['Weekly_Sales']
df_final = df.drop(columns = ['Weekly_Sales', 'Date'])
```

In [72]:

```
df_final
```

Out[72]:

	Store	Dept	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3
0	1	1	0	42.31	2.572	0.00	0.00	0.0
1	1	2	0	42.31	2.572	0.00	0.00	0.0
2	1	3	0	42.31	2.572	0.00	0.00	0.0
3	1	4	0	42.31	2.572	0.00	0.00	0.0
4	1	5	0	42.31	2.572	0.00	0.00	0.0
...	...	...	...	...	...	...	...	...
421565	45	93	0	58.85	3.882	4018.91	58.08	100.0
421566	45	94	0	58.85	3.882	4018.91	58.08	100.0
421567	45	95	0	58.85	3.882	4018.91	58.08	100.0
421568	45	97	0	58.85	3.882	4018.91	58.08	100.0
421569	45	98	0	58.85	3.882	4018.91	58.08	100.0

421570 rows × 15 columns

In [73]:

```
df_final = pd.get_dummies(df_final, columns = ['Type', 'Store', 'Dept'], drc
```

In [74]:

```
df_final
```

Out[74]:

	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4
0	0	42.31	2.572	0.00	0.00	0.0	0.00
1	0	42.31	2.572	0.00	0.00	0.0	0.00
2	0	42.31	2.572	0.00	0.00	0.0	0.00
3	0	42.31	2.572	0.00	0.00	0.0	0.00
4	0	42.31	2.572	0.00	0.00	0.0	0.00
...	...	...	...	...	...	...	...
421565	0	58.85	3.882	4018.91	58.08	100.0	211.94
421566	0	58.85	3.882	4018.91	58.08	100.0	211.94
421567	0	58.85	3.882	4018.91	58.08	100.0	211.94
421568	0	58.85	3.882	4018.91	58.08	100.0	211.94
421569	0	58.85	3.882	4018.91	58.08	100.0	211.94

421570 rows × 138 columns

In [75]:

```
df_final.shape
```

Out[75]: (421570, 138)

In [76]:

```
df_target.shape
```

Out[76]: (421570,)

In [77]:
```
df_final
# df_final['Dept_91'].dtype
```

Out[77]:

	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4
0	0	42.31	2.572	0.00	0.00	0.0	0.00
1	0	42.31	2.572	0.00	0.00	0.0	0.00
2	0	42.31	2.572	0.00	0.00	0.0	0.00
3	0	42.31	2.572	0.00	0.00	0.0	0.00
4	0	42.31	2.572	0.00	0.00	0.0	0.00
...	...	...	...	...	...	...	...
421565	0	58.85	3.882	4018.91	58.08	100.0	211.94
421566	0	58.85	3.882	4018.91	58.08	100.0	211.94
421567	0	58.85	3.882	4018.91	58.08	100.0	211.94
421568	0	58.85	3.882	4018.91	58.08	100.0	211.94
421569	0	58.85	3.882	4018.91	58.08	100.0	211.94

421570 rows × 138 columns

In [78]:
```
X = np.array(df_final).astype('float32')
y = np.array(df_target).astype('float32')
```

In [79]:
```
# reshaping the array from (421570,) to (421570, 1)
y = y.reshape(-1,1)
y.shape
```

Out[79]: (421570, 1)

In [80]:
```
# scaling the data before feeding the model
# from sklearn.preprocessing import StandardScaler, MinMaxScaler

# scaler_x = StandardScaler()
# X = scaler_x.fit_transform(X)

# scaler_y = StandardScaler()
# y = scaler_y.fit_transform(y)
```

In [81]:
```
# spliting the data in to test and train sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.15)
X_test, X_val, y_test, y_val = train_test_split(X_test, y_test, test_size =
```

In [82]:
```
X_train
```

Out[82]:
```
array([[ 0.   , 14.48 ,  2.788, ...,  0.    ,  0.    ,  0.    ],
       [ 0.   , 68.28 ,  3.623, ...,  0.    ,  0.    ,  0.    ],
       [ 0.   , 36.46 ,  3.261, ...,  0.    ,  0.    ,  0.    ],
       ...,
       [ 0.   , 66.59 ,  4.169, ...,  0.    ,  0.    ,  0.    ],
       [ 0.   , 21.33 ,  2.788, ...,  0.    ,  0.    ,  0.    ],
       [ 1.   , 47.87 ,  2.946, ...,  0.    ,  0.    ,  0.    ]],
      dtype=float32)
```

TRAINING XGBOOST REGRESSOR IN LOCAL MODE

In [83]:
```
!pip install xgboost
```

```
Collecting xgboost
  Obtaining dependency information for xgboost from https://files.pythonhos
ted.org/packages/c1/cf/a662bc8f40588d54663edfe12980946670490bff0b6e793c7896
a4fe36df/xgboost-2.0.0-py3-none-manylinux2014_x86_64.whl.metadata
  Downloading xgboost-2.0.0-py3-none-manylinux2014_x86_64.whl.metadata (2.0
kB)
Requirement already satisfied: numpy in /home/ec2-user/anaconda3/envs/pytho
n3/lib/python3.10/site-packages (from xgboost) (1.22.3)
Requirement already satisfied: scipy in /home/ec2-user/anaconda3/envs/pytho
n3/lib/python3.10/site-packages (from xgboost) (1.11.1)
Downloading xgboost-2.0.0-py3-none-manylinux2014_x86_64.whl (297.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 297.1/297.1 MB 3.6 MB/s eta 0:0
0:00:00:0100:01
Installing collected packages: xgboost
Successfully installed xgboost-2.0.0
```

In [86]:
```
# Train an XGBoost regressor model

import xgboost as xgb


model = xgb.XGBRegressor(objective ='reg:squarederror', learning_rate = 0.1,

model.fit(X_train, y_train)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Out[86]:

▼ XGBRegressor

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rou
nds=None,
             enable_categorical=False, eval_metric=None, feature_ty
pes=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.1, max_b
in=None,
```

In [87]:

```python
# predict the score of the trained model using the testing dataset

result = model.score(X_test, y_test)

print("Accuracy : {}".format(result))
```

```
Accuracy : 0.817197891650735
```

In [88]:

```python
# make predictions on the test data

y_predict = model.predict(X_test)
```

In [114…

```python
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_erro
from math import sqrt
k = X_test.shape[1]
n = len(X_test)
RMSE = float(format(np.sqrt(mean_squared_error(y_test, y_predict)),'.3f'))
MSE = mean_squared_error(y_test, y_predict)
MAE = mean_absolute_error(y_test, y_predict)
r2 = r2_score(y_test, y_predict)
adj_r2 = 1-(1-r2)*(n-1)/(n-k-1)

print('RMSE =',RMSE, '\nMSE =',MSE, '\nMAE =',MAE, '\nR2 =', r2, '\nAdjustec
```

```
RMSE = 9598.279
MSE = 92126960.0
MAE = 6432.6113
R2 = 0.817197891650735
Adjusted R2 = 0.8163965100645283
```

TRAIN XGBOOST USING AWS SAGEMAKER

In [92]:

```python
# Convert the array into dataframe in a way that target variable is set as t
# This is because sagemaker built-in algorithm expects the data in this form

train_data = pd.DataFrame({'Target': y_train[:,0]})
```

```
for i in range(X_train.shape[1]):
    train_data[i] = X_train[:,i]
```

```
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
                   the result of calling `frame.insert` many t
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
To get a de-fragmented frame, use `newframe =
```

```
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
```

```
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
the result of calling `frame.insert` many t
```

```
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
/tmp/ipykernel_31899/2145295216.py:6: PerformanceWarning: DataFrame is high
ly fragmented.  This is usually the result of calling `frame.insert` many t
imes, which has poor performance.  Consider joining all columns at once usi
ng pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data[i] = X_train[:,i]
```

In [91]:

```
train_data.head()
```

Out[91]:

	Target	0	1	2	3	4	5	6	
0	6700.479980	0.0	14.480000	2.788	0.000000	0.000000	0.00	0.000000	0.0000
1	4752.750000	0.0	68.279999	3.623	0.000000	0.000000	0.00	0.000000	0.0000
2	3350.169922	0.0	36.459999	3.261	6725.290039	12764.990234	15.98	299.730011	3851.6899
3	12487.440430	0.0	62.060001	2.992	0.000000	0.000000	0.00	0.000000	0.0000
4	17425.750000	0.0	60.380001	4.066	0.000000	0.000000	0.00	0.000000	0.0000

5 rows × 139 columns

In [93]:

```
val_data = pd.DataFrame({'Target':y_val[:,0]})
for i in range(X_val.shape[1]):
    val_data[i] = X_val[:,i]
```

```
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
```
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js
```
the result of calling `frame.insert` many ti
```

```
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
o get a de-fragmented frame, use `newframe =
```

```
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
```

```
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
                                                the result of calling `frame.insert` many ti
```

```
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
/tmp/ipykernel_31899/565076207.py:3: PerformanceWarning: DataFrame is highl
y fragmented.  This is usually the result of calling `frame.insert` many ti
mes, which has poor performance.  Consider joining all columns at once usin
g pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  val_data[i] = X_val[:,i]
```

In [94]:
```
val_data.head()
```

Out[94]:

	Target	0	1	2	3	4	5	6	7	8	...	128	129	130	1:
0	5107.509766	0.0	57.160000	2.886	0.0	0.0	0.0	0.0	0.0	214.701782	...	0.0	0.0	0.0	C
1	74557.250000	0.0	70.900002	2.725	0.0	0.0	0.0	0.0	0.0	209.170776	...	0.0	0.0	0.0	C
2	2631.409912	0.0	47.410000	3.567	0.0	0.0	0.0	0.0	0.0	129.793671	...	0.0	0.0	0.0	C
3	6683.109863	0.0	36.779999	2.817	0.0	0.0	0.0	0.0	0.0	126.793404	...	0.0	0.0	0.0	C
4	6921.649902	0.0	48.889999	2.625	0.0	0.0	0.0	0.0	0.0	211.907166	...	0.0	0.0	0.0	C

5 rows × 139 columns

In [95]:
```
val_data.shape
```

Out[95]:
```
(31618, 139)
```

In [96]:
```
# save train_data and validation_data as csv files.

train_data.to_csv('train.csv', header = False, index = False)
val_data.to_csv('validation.csv', header = False, index = False)
```

In [97]:
```
# Boto3 is the Amazon Web Services (AWS) Software Development Kit (SDK) for
# Boto3 allows Python developer to write software that makes use of services

import sagemaker
import boto3
from sagemaker import Session

# create a Sagemaker session
sagemaker_session = sagemaker.Session()
bucket = Session().default_bucket()
prefix = 'XGBoost-Regressor'
key = 'XGBoost-Regressor'
#Roles give learning and hosting access to the data
#This is specified while opening the sagemakers instance in "Create an IAM r
role = sagemaker.get_execution_role()
```

In [98]:
```
print(role)
```

```
arn:aws:iam::058058168096:role/service-role/AmazonSageMaker-ExecutionRole-2
0230906T182286
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [99]:
```
# read the data from csv file and then upload the data to s3 bucket
import os
with open('train.csv','rb') as f:
    # The following code uploads the data into S3 bucket to be accessed late
    boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix

# print out the training data location in s3
s3_train_data = 's3://{}/{}/train/{}'.format(bucket, prefix, key)
print('uploaded training data location: {}'.format(s3_train_data))
```

```
uploaded training data location: s3://sagemaker-us-east-1-058058168096/XGBo
ost-Regressor/train/XGBoost-Regressor
```

In [100…
```
# read the data from csv file and then upload the data to s3 bucket

with open('validation.csv','rb') as f:
    # The following code uploads the data into S3 bucket to be accessed late

    boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix
# print out the validation data location in s3
s3_validation_data = 's3://{}/{}/validation/{}'.format(bucket, prefix, key)
print('uploaded validation data location: {}'.format(s3_validation_data))
```

```
uploaded validation data location: s3://sagemaker-us-east-1-058058168096/XG
Boost-Regressor/validation/XGBoost-Regressor
```

In [101…
```
# creates output placeholder in S3 bucket to store the output

output_location = 's3://{}/{}/output'.format(bucket, prefix)
print('training artifacts will be uploaded to: {}'.format(output_location))
```

```
training artifacts will be uploaded to: s3://sagemaker-us-east-1-0580581680
96/XGBoost-Regressor/output
```

In [102…
```
# This code is used to get the training container of sagemaker built-in algo

# obtain a reference to the XGBoost container image
# all regression models are named estimators

from sagemaker.amazon.amazon_estimator import get_image_uri

container = get_image_uri(boto3.Session().region_name, 'xgboost','0.90-2')
```

```
The method get_image_uri has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
```

In [103…
```
# Specify the type of instance that we would like to use for training
# output path and sagemaker session into the Estimator.
# specify how many instances we would like to use for training

# XGBoost works by combining an ensemble of weak models to generate accurate
# The weak models are randomized to avoid overfitting

# num_round: The number of rounds to run the training.


# … on weights. Increasing this value makes mode
```

```
# colsample_by_tree: fraction of features that will be used to train each tr

# eta: Step size shrinkage used in updates to prevent overfitting.
# After each boosting step, eta parameter shrinks the feature weights to mak


Xgboost_regressor1 = sagemaker.estimator.Estimator(container,
                                       role,
                                       train_instance_count = 1,
                                       train_instance_type = 'ml.m5.2xlarge'
                                       output_path = output_location,
                                       sagemaker_session = sagemaker_sessior

# tune the hyper-parameters to improve the performance of the model

Xgboost_regressor1.set_hyperparameters(max_depth = 10,
                           objective = 'reg:linear',
                           colsample_bytree = 0.3,
                           alpha = 10,
                           eta = 0.1,
                           num_round = 100
                           )
```

```
train_instance_count has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
train_instance_type has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
```

In [104…

```
# Creating "train", "validation" channels to feed in the model
# Source: https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-algo-doc

train_input = sagemaker.session.s3_input(s3_data = s3_train_data, content_ty
valid_input = sagemaker.session.s3_input(s3_data = s3_validation_data, conte


data_channels = {'train': train_input,'validation': valid_input}


Xgboost_regressor1.fit(data_channels)
```

```
The class sagemaker.session.s3_input has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
The class sagemaker.session.s3_input has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
INFO:sagemaker:Creating training-job with name: sagemaker-xgboost-2023-09-1
2-10-00-17-538
```

```
2023-09-12 10:00:17 Starting - Starting the training job...
2023-09-12 10:00:34 Starting - Preparing the instances for trainin
g............
2023-09-12 10:02:23 Downloading - Downloading input data...
2023-09-12 10:03:13 Training - Training image download completed. Training
in progress....INFO:sagemaker-containers:Imported framework sagemaker_xgboo
st_container.training
INFO:sagemaker-containers:Failed to parse hyperparameter objective value re
g:linear to Json.
Returning the value itself
INFO:sagemaker-containers:No GPUs detected (normal if no gpus installed)
INFO:sagemaker_xgboost_container.training:Running XGBoost Sagemaker in algo
rithm mode
INFO:root:Determined delimiter of CSV input is ','
INFO:root:Determined delimiter of CSV input is ','
INFO:root:Determined delimiter of CSV input is ','
[10:03:35] 358334x138 matrix with 49450092 entries loaded from /opt/ml/inpu
t/data/train?format=csv&label_column=0&delimiter=,
INFO:root:Determined delimiter of CSV input is ','
[10:03:35] 31618x138 matrix with 4363284 entries loaded from /opt/ml/input/
data/validation?format=csv&label_column=0&delimiter=,
INFO:root:Single node training.
[2023-09-12 10:03:35.816 ip-10-0-230-88.ec2.internal:7 INFO json_config.py:
90] Creating hook from json_config at /opt/ml/input/config/debughookconfig.
json.
[2023-09-12 10:03:35.816 ip-10-0-230-88.ec2.internal:7 INFO hook.py:151] te
nsorboard_dir has not been set for the hook. SMDebug will not be exporting
tensorboard summaries.
[2023-09-12 10:03:35.816 ip-10-0-230-88.ec2.internal:7 INFO hook.py:196] Sa
ving to /opt/ml/output/tensors
INFO:root:Debug hook created from config
INFO:root:Train matrix has 358334 rows
INFO:root:Validation matrix has 31618 rows
[10:03:35] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:lin
ear is now deprecated in favor of reg:squarederror.
[0]#011train-rmse:26586.6#011validation-rmse:26500.8
[2023-09-12 10:03:37.547 ip-10-0-230-88.ec2.internal:7 INFO hook.py:325] Mo
nitoring the collections: metrics
[1]#011train-rmse:25435.3#011validation-rmse:25345.3
[2]#011train-rmse:24456.2#011validation-rmse:24352.1
[3]#011train-rmse:23715.4#011validation-rmse:23612.5
[4]#011train-rmse:23000#011validation-rmse:22921.6
[5]#011train-rmse:22305.6#011validation-rmse:22207
[6]#011train-rmse:21433.6#011validation-rmse:21321.2
[7]#011train-rmse:20519.7#011validation-rmse:20420.3
[8]#011train-rmse:20178.4#011validation-rmse:20073.9
[9]#011train-rmse:19578.1#011validation-rmse:19484
[10]#011train-rmse:19084.7#011validation-rmse:18985.6
[11]#011train-rmse:18452.2#011validation-rmse:18337.8
[12]#011train-rmse:17838.4#011validation-rmse:17742.4
[13]#011train-rmse:17285.9#011validation-rmse:17192.8
[14]#011train-rmse:16832.3#011validation-rmse:16744.9
[15]#011train-rmse:16394.1#011validation-rmse:16297.5
[16]#011train-rmse:16107.5#011validation-rmse:16016.1
[17]#011train-rmse:15983.1#011validation-rmse:15893.2
alidation-rmse:15699.9
```

```
[19]#011train-rmse:15689.2#011validation-rmse:15605.1
[20]#011train-rmse:15371.4#011validation-rmse:15293.9
[21]#011train-rmse:15212.1#011validation-rmse:15136.2
[22]#011train-rmse:14988.2#011validation-rmse:14915.8
[23]#011train-rmse:14678.1#011validation-rmse:14610.3
[24]#011train-rmse:14504.4#011validation-rmse:14433.4
[25]#011train-rmse:14355.7#011validation-rmse:14300.5
[26]#011train-rmse:14133.2#011validation-rmse:14094.6
[27]#011train-rmse:14015.1#011validation-rmse:13979.3
[28]#011train-rmse:13560.6#011validation-rmse:13544
[29]#011train-rmse:13302.2#011validation-rmse:13279.4
[30]#011train-rmse:13115.1#011validation-rmse:13090.3
[31]#011train-rmse:12959.7#011validation-rmse:12945.8
[32]#011train-rmse:12861.6#011validation-rmse:12847.5
[33]#011train-rmse:12716.7#011validation-rmse:12718.2
[34]#011train-rmse:12389.7#011validation-rmse:12394.7
[35]#011train-rmse:12193.2#011validation-rmse:12200
[36]#011train-rmse:12081.2#011validation-rmse:12098.9
[37]#011train-rmse:11845.4#011validation-rmse:11850.7
[38]#011train-rmse:11713.9#011validation-rmse:11723
[39]#011train-rmse:11614.2#011validation-rmse:11622
[40]#011train-rmse:11426.1#011validation-rmse:11435.8
[41]#011train-rmse:11268.6#011validation-rmse:11272.2
[42]#011train-rmse:11166.8#011validation-rmse:11179.4
[43]#011train-rmse:10840.9#011validation-rmse:10869.8
[44]#011train-rmse:10746.9#011validation-rmse:10782.4
[45]#011train-rmse:10627.4#011validation-rmse:10665.6
[46]#011train-rmse:10527.4#011validation-rmse:10574
[47]#011train-rmse:10446.6#011validation-rmse:10515.6
[48]#011train-rmse:10249.5#011validation-rmse:10334.2
[49]#011train-rmse:10085.7#011validation-rmse:10171.6
[50]#011train-rmse:9979.64#011validation-rmse:10066.1
[51]#011train-rmse:9910.94#011validation-rmse:9997.85
[52]#011train-rmse:9850.34#011validation-rmse:9938.27
[53]#011train-rmse:9714.49#011validation-rmse:9817.57
[54]#011train-rmse:9646.54#011validation-rmse:9756.74
[55]#011train-rmse:9564.92#011validation-rmse:9683.46
[56]#011train-rmse:9514.01#011validation-rmse:9634.65
[57]#011train-rmse:9432.3#011validation-rmse:9561.6
[58]#011train-rmse:9360.52#011validation-rmse:9481.7
[59]#011train-rmse:9310.02#011validation-rmse:9432.56
[60]#011train-rmse:9180.98#011validation-rmse:9303.13
[61]#011train-rmse:9065.62#011validation-rmse:9201.94
[62]#011train-rmse:8986.18#011validation-rmse:9129.1
[63]#011train-rmse:8914.86#011validation-rmse:9067.02
[64]#011train-rmse:8838.01#011validation-rmse:8998.49
[65]#011train-rmse:8781.28#011validation-rmse:8942.54
[66]#011train-rmse:8722.13#011validation-rmse:8895.07
[67]#011train-rmse:8689.65#011validation-rmse:8862.72
[68]#011train-rmse:8619.56#011validation-rmse:8793.46
[69]#011train-rmse:8560.63#011validation-rmse:8742.47
[70]#011train-rmse:8492.06#011validation-rmse:8684.47
[71]#011train-rmse:8373.66#011validation-rmse:8566.67
[72]#011train-rmse:8242.66#011validation-rmse:8430.21
[73]#011train-rmse:8144.76#011validation-rmse:8328.04
alidation-rmse:8251.17
```

```
[75]#011train-rmse:7984.53#011validation-rmse:8170.59
[76]#011train-rmse:7951.54#011validation-rmse:8137.55
[77]#011train-rmse:7924.07#011validation-rmse:8110.42
[78]#011train-rmse:7880.46#011validation-rmse:8070.16
[79]#011train-rmse:7848.41#011validation-rmse:8048.21
[80]#011train-rmse:7806.43#011validation-rmse:8009.08
[81]#011train-rmse:7771.96#011validation-rmse:7977.42
[82]#011train-rmse:7696.19#011validation-rmse:7904.07
[83]#011train-rmse:7657.25#011validation-rmse:7866.07
[84]#011train-rmse:7632.37#011validation-rmse:7844.34
[85]#011train-rmse:7569.4#011validation-rmse:7785.22
[86]#011train-rmse:7534.03#011validation-rmse:7751.14
[87]#011train-rmse:7508.04#011validation-rmse:7728.27
[88]#011train-rmse:7470.61#011validation-rmse:7695.96
[89]#011train-rmse:7433.76#011validation-rmse:7658.96
[90]#011train-rmse:7389.37#011validation-rmse:7618.55
[91]#011train-rmse:7375.29#011validation-rmse:7606.16
[92]#011train-rmse:7293.65#011validation-rmse:7533.58

2023-09-12 10:04:20 Uploading - Uploading generated training model
2023-09-12 10:04:20 Completed - Training job completed
[93]#011train-rmse:7259.8#011validation-rmse:7507.92
[94]#011train-rmse:7197.65#011validation-rmse:7446.53
[95]#011train-rmse:7170.32#011validation-rmse:7423.62
[96]#011train-rmse:7121.84#011validation-rmse:7382.06
[97]#011train-rmse:7091.56#011validation-rmse:7358.16
[98]#011train-rmse:7064.85#011validation-rmse:7333.56
[99]#011train-rmse:7018.94#011validation-rmse:7297.27
Training seconds: 118
Billable seconds: 118
```

DEPLOYMENT AND PREDICTIONS

In [105…

```
# Deploy the model to perform inference

Xgboost_regressor = Xgboost_regressor1.deploy(initial_instance_count = 1, ir
```

```
INFO:sagemaker:Creating model with name: sagemaker-xgboost-2023-09-12-10-08
-20-377
INFO:sagemaker:Creating endpoint-config with name sagemaker-xgboost-2023-09
-12-10-08-20-377
INFO:sagemaker:Creating endpoint with name sagemaker-xgboost-2023-09-12-10-
08-20-377
----!
```

In [112…

```
'''
Content type over-rides the data that will be passed to the deployed model,
in text/csv format, we specify this as content -type.

Serializer accepts a single argument, the input data, and returns a sequence
type

Reference: https://sagemaker.readthedocs.io/en/stable/predictors.html
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
'''
# from sagemaker.predictor import csv_serializer, json_deserializer
# sagemaker.serializers.CSVSerializer()
json_deserializer = sagemaker.serializers.JSONSerializer()
csv_serializer = sagemaker.serializers.CSVSerializer()
Xgboost_regressor.serializer = csv_serializer

# Xgboost_regressor.serializer = sagemaker.serializers.CSVSerializer()
```

In [118…
```
X_test.shape
```

Out[118]: (31618, 138)

Making Predictions

In [119…
```
predictions1 = Xgboost_regressor.predict(X_test[0:10000])
```

In [120…
```
predictions2 = Xgboost_regressor.predict(X_test[10000:20000])
```

In [121…
```
predictions3 = Xgboost_regressor.predict(X_test[20000:30000])
```

In [122…
```
predictions4 = Xgboost_regressor.predict(X_test[30000:31618])
```

In [123…
```
predictions4
```

Out[123]: b'8543.11328125,54440.7421875,13812.4130859375,8785.1767578125,2328.251464
84375,6407.22509765625,11913.3310546875,7317.0625,47968.51953125,2328.2514
6484375,7687.2724609375,5340.92578125,13483.8310546875,11745.2626953125,47
009.4375,5346.1796875,5714.16650390625,14951.8486328125,6197.35107421875,4
125.037109375,14129.912109375,11157.7685546875,3235.2568359375,1255.873046
875,19357.87890625,2308.013916015625,9425.5751953125,6450.61767578125,5292
9.265625,8382.1640625,8858.6396484375,-932.96240234375,10775.3203125,6648.
84814453125,2415.410888671875,31984.16015625,17211.77734375,2454.21289062
5,50732.23828125,10703.4970703125,12341.6640625,3031.784912109375,17520.66
2109375,10835.7216796875,6647.330078125,1670.9278564453125,6766.6743164062
5,42430.44140625,6753.4404296875,95109.265625,93058.0703125,3356.531494140
625,-175.30093383789062,41878.03515625,15140.76953125,15102.7373046875,107
54.056640625,31083.78515625,11054.7080078125,59487.54296875,7332.90234375,
73159.625,12256.166015625,7352.41162109375,4411.3369140625,6798.7026367187
5,2363.69970703125,6516.232421875,6588.16796875,14788.3046875,5931.3481445
3125,29284.8359375,19526.912109375,5347.8095703125,10684.2783203125,67091.
4140625,8762.6015625,1486.032470703125,3328.400146484375,17653.107421875,1
4083.68359375,18934.12890625,1540.938720703125,33308.55078125,7225.8364257
8125,16747.025390625,7008.83544921875,17369.94140625,5516.8759765625,1145
4.2939453125,865.013427734375,29092.791015625,5622.203125,5110.8291015625,
20540.95703125,3124.50244140625,512.2969360351562,12773.580078125,1323.904
9072265625,106.65623474121094,66617.6640625,26334.052734375,2856.626464843
75,2871.98193359375,1945.7899169921875,58954.796875,28100.49609375,41422.2
8515625,14388.24609375,23209.169921875,76747.140625,15355.560546875,25591.
763671875,8775.19140625,3987.917724609375,13378.763671875,2442.3671875,374
9.666259765625,10682.9775390625,38263.42578125,29405.865234375,22925.16992
1875,1502.859130859375,10972.2802734375,10645.1591796875,3321.28198242187
5,39070.2890625,15790.4580078125,11279.2294921875,1571.204833984375,3082.7
66357421875,10584.3193359375,16538.150390625,24550.724609375,8891.75097656
25,1030.732421875,17491.91796875,10433.49609375,4925.68408203125,6204.1430
6640625,19713.3359375,10627.3701171875,10931.9892578125,3601.23486328125,6
674.8291015625,64265.79296875,2465.50634765625,5518.892578125,45533.628906
25,13155.927734375,3224.54638671875,22199.642578125,5090.55419921875,7300.
3603515625,9361.193359375,3300.51806640625,8723.6357421875,2302.0676269531
25,4149.31201171875,15077.427734375,11362.8125,14384.3349609375,4224.43261
71875,29670.326171875,19163.0,14384.3349609375,5682.33544921875,4007.57226
5625,80905.4140625,11183.4794921875,7746.88134765625,10561.0693359375,3227
9.046875,1182.980712890625,11676.3740234375,16373.173828125,9361.19335937
5,4095.330078125,23324.763671875,8553.7333984375,44644.59375,5139.39550781
25,38490.28125,9000.703125,13181.5419921875,8591.107421875,10224.116210937
5,20069.400390625,2770.615966796875,13423.078125,14658.2236328125,9693.706
0546875,3731.5556640625,48106.16796875,17207.068359375,3065.31005859375,15
878.7587890625,18489.412109375,2366.7001953125,52072.50390625,12194.634765
625,5302.70263671875,14326.142578125,-502.24127197265625,6421.63037109375,
6232.79052734375,10017.794921875,2520.2978515625,10672.9697265625,44659.69
140625,12137.6552734375,3614.283447265625,7251.84716796875,3678.86328125,6
2955.92578125,16349.1220703125,38027.3125,16564.244140625,4489.3466796875,
-2030.1536865234375,5431.87939453125,8198.0361328125,13130.326171875,212.7
566680908203,-1976.7977294921875,20774.1875,16610.83203125,12943.76953125,
10627.3701171875,41696.2421875,22722.908203125,55549.72265625,16385.917968
75,3149.370361328125,3050.989990234375,4528.54248046875,7087.1171875,3062
3.544921875,11362.125,11819.451171875,18664.380859375,18745.380859375,307
0.87353515625,16131.9228515625,5795.7431640625,9743.8955078125,13546.87304
6875,15452.6005859375,-510.7937927246094,6617.5361328125,4845.84619140625,
5340.92578125,13591.83984375,4718.67919921875,9699.416015625,1810.58862304
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js 734375,3651.7939453125,5187.89599609375,1106

9.7470703125,11711.85546875,9471.9775390625,6764.6279296875,-363.44335937
5,13080.1552734375,10430.4228515625,27461.650390625,23452.533203125,23854.
666015625,9121.0048828125,39094.81640625,16432.576171875,30806.80859375,11
193.998046875,15140.76953125,12409.0576171875,2820.9189453125,4567.4145507
8125,5874.96826171875,9225.7529296875,27223.798828125,5492.56591796875,703
9.24658203125,15749.4755859375,18499.705078125,2328.307861328125,4036.1840
8203125,7234.56787109375,5675.36181640625,63677.6328125,11239.634765625,20
257.86328125,5948.33203125,25415.4296875,3494.1376953125,9563.5927734375,4
2736.5,29449.37890625,1342.3929443359375,6573.98193359375,6827.9072265625,
7737.564453125,4085.01904296875,7623.02197265625,11326.916015625,10946.690
4296875,6832.845703125,16325.37109375,3417.31201171875,5469.71630859375,22
33.09765625,35016.81640625,44724.03125,23379.47265625,189.7080841064453,12
544.0029296875,2855.045654296875,3188.4365234375,18180.966796875,41427.074
21875,59873.69921875,4090.66455078125,7975.181640625,2798.362060546875,722
6.50146484375,50654.48828125,20236.0390625,3371.468994140625,15987.460937
5,8402.080078125,48250.79296875,7443.2431640625,3069.302734375,36119.0,384
4.37548828125,8202.6591796875,17369.94140625,6632.87890625,3637.9841308593
75,24286.32421875,3531.04931640625,16693.6484375,32732.33984375,29859.9570
3125,15357.3134765625,51609.3515625,2350.035888671875,19389.96875,11323.60
7421875,12349.296875,5256.49462890625,10897.52734375,14341.5390625,42284.2
734375,10960.6953125,11152.427734375,20920.939453125,24903.533203125,3020
1.708984375,36870.3671875,2838.567626953125,196.6475067138672,2901.5268554
6875,18241.833984375,24738.0625,9763.40625,24079.626953125,19199.28125,317
25.2734375,17211.77734375,42726.29296875,2788.422119140625,4184.607421875,
62041.87109375,4115.9384765625,9087.5703125,15913.255859375,-292.615753173
8281,21192.037109375,9789.9609375,5183.73486328125,3406.914794921875,1068
5.865234375,6346.67919921875,11542.0322265625,35098.90625,14018.920898437
5,8231.083984375,15414.1240234375,10561.0693359375,18715.42578125,21768.33
203125,14025.1318359375,5889.1552734375,3868.00390625,70721.640625,3865.17
5048828125,7771.697265625,27329.9921875,9415.0849609375,3918.763427734375,
6615.73681640625,9092.0888671875,55599.515625,11830.8740234375,20218.77148
4375,25415.4296875,4919.55517578125,16106.44921875,2232.453369140625,1833
4.2734375,-960.0330810546875,5605.45703125,7285.248046875,16783.80859375,4
650.5888671875,15765.5634765625,23795.953125,16246.46875,68468.734375,-22
5.74046325683594,18107.701171875,9361.193359375,4761.8486328125,8050.66992
1875,9984.837890625,3843.45654296875,7044.35693359375,16412.615234375,797
8.048828125,1612.2457275390625,2960.9423828125,20750.435546875,2731.002685
546875,79959.90625,6672.56396484375,66131.4296875,13899.7470703125,3730.80
859375,47217.94140625,22848.177734375,6261.89501953125,7413.2705078125,490
30.9140625,16269.8681640625,16627.005859375,60715.21875,4673.05029296875,1
8925.85546875,13680.6083984375,6352.76953125,1155.29833984375,6353.9736328
125,9936.3056640625,57520.71484375,11274.9775390625,16060.3818359375,7087.
44287109375,459.615234375,15155.2724609375,69250.9140625,8148.4130859375,1
4539.650390625,14948.45703125,502.0535583496094,13450.9140625,5038.9995117
1875,16382.630859375,16199.3203125,10116.8505859375,16911.41796875,25734.9
23828125,13996.5703125,7629.62646484375,11894.21484375,12219.3408203125,13
640.6044921875,25429.9765625,5989.564453125,25327.859375,28425.427734375,1
9226.994140625,44116.28125,55422.41796875,20280.18359375,12981.6953125,833
5.9765625,92629.4765625,6465.6337890625,9378.7177734375,3497.358642578125,
56274.4140625,-1338.8760986328125,4096.61865234375,-105.8684310913086,1953
3.44140625,21373.60546875,58330.69921875,2463.863037109375,2970.1215820312
5,6542.44189453125,5989.564453125,81161.265625,13044.6201171875,7744.33447
265625,33279.92578125,11304.1630859375,15851.2890625,5679.47119140625,-87
4.0453491210938,10593.359375,5542.642578125,1223.8507080078125,16509.83593
75,14265.6181640625,5227.4736328125,10221.05078125,3389.154541015625,1183
62,7284.3779296875,6142.39404296875,3078.277

83203125,2864.78759765625,7866.9697265625,23054.619140625,4298.3662109375,
3104.504638671875,3091.586181640625,5892.11376953125,17899.427734375,1438
4.3349609375,17518.544921875,3063.839111328125,16514.728515625,7848.667480
46875,39265.171875,9126.0986328125,7021.50537109375,4140.298828125,221.626
8768310547,11361.1201171875,35939.93359375,7072.1171875,11838.546875,6044
2.53515625,3266.570556640625,24708.068359375,5989.52685546875,7225.8364257
8125,19180.96875,10122.1220703125,87350.5546875,8341.5439453125,6831.07275
390625,16565.9921875,9969.8759765625,8786.146484375,44911.7734375,8104.809
08203125,11573.416015625,-128.06846618652344,4058.236083984375,8077.100585
9375,15231.0283203125,1401.9324951171875,6438.15625,5582.17431640625,374.7
129821777344,6090.98388671875,21025.865234375,-725.19775390625,52810.53906
25,4808.95947265625,27634.7109375,13285.212890625,5577.96826171875,9084.48
14453125,25090.078125,10702.650390625,19523.072265625,127563.3125,2626.842
7734375,12584.30078125,16106.44921875,34456.64453125,4375.73828125,20604.2
265625,16881.27734375,8348.7373046875,10290.9189453125,9367.3095703125,676
4.6279296875,31192.552734375,4231.9921875,17609.90625,11600.7529296875,139
80.150390625,10960.453125,14145.0087890625,23146.57421875,19273.845703125,
16906.416015625,12965.755859375,3097.418212890625,10348.0810546875,10501.2
197265625,58887.66796875,5299.2958984375,28621.90625,6209.95068359375,1656
5.9921875,4618.2294921875,27221.466796875,2200.823974609375,8044.065429687
5,12179.9140625,5984.22802734375,15152.990234375,54181.94140625,23049.4921
875,10108.619140625,55462.65234375,24577.447265625,13194.625,12156.1347656
25,7021.50537109375,9858.6083984375,13520.3623046875,1276.89599609375,2681
8.064453125,11239.634765625,10995.515625,9282.4091796875,6827.9072265625,1
0287.5751953125,11152.5927734375,1400.9892578125,5231.61279296875,22854.28
7109375,64282.60546875,-267.39007568359375,6935.27490234375,4827.336914062
5,73108.5234375,8933.642578125,-940.1513671875,39736.1875,7140.4541015625,
6124.0966796875,9207.001953125,16747.025390625,2471.65576171875,17062.1992
1875,7653.23974609375,36314.6640625,6796.39892578125,33127.27734375,12268.
015625,39736.1875,2853.6181640625,1100.417236328125,32073.787109375,8869.3
759765625,15321.2744140625,9361.193359375,8003.76123046875,111276.375,349
1.78515625,11061.1435546875,20755.349609375,5924.501953125,13834.264648437
5,20877.38671875,4568.48388671875,22275.720703125,20734.912109375,9395.911
1328125,111367.0,10944.0146484375,2716.766357421875,9264.4677734375,8134.2
5439453125,8976.0703125,9585.064453125,5620.49560546875,2595.4873046875,18
645.0859375,12593.2392578125,15560.4267578125,72918.46875,15564.810546875,
8068.32568359375,6424.66552734375,100154.140625,12930.7919921875,13628.298
828125,5401.42724609375,7303.17578125,47032.12109375,14265.6181640625,676
1.7783203125,13640.6044921875,9471.9775390625,-92.89285278320312,9959.0263
671875,1697.031127929 6875,16134.8046875,-616.3517456054688,2451.803710937
5,14160.9189453125,11898.279296875,47465.390625,63224.03125,11534.6992187
5,8001.5810546875,-1813.2421875,10619.8662109375,8684.67578125,8451.746093
75,7966.5751953125,2676.197509765625,86476.203125,16106.44921875,7917.8886
71875,6465.6337890625,15120.8974609375,2636.98291015625,8726.7763671875,89
93.61328125,44406.6015625,14617.400390625,2502.830078125,30086.521484375,3
1504.607421875,511.9810791015625,3031.784912109375,16000.6806640625,8598.8
857421875,6868.29345703125,18757.275390625,7284.3779296875,2083.7546386718
75,4847.091796875,17015.033203125,6849.173828125,4076.783203125,4030.33789
0625,6725.56689453125,3912.19091796875,13994.6220703125,23206.8984375,1116
8.404296875,10280.30859375,20782.98046875,12096.43359375,14250.265625,8850
0.4921875,66698.7421875,3704.445556640625,22004.806640625,3785.0903320312
5,-1269.989013671875,14558.4189453125,4585.7119140625,3448.976806640625,76
343.0390625,6950.48828125,9775.455078125,9538.166015625,42180.55859375,151
40.76953125,66443.328125,3745.058349609375,9150.513671875,29386.205078125,
2555.52490234375,7714.38232421875,4481.13134765625,23664.05078125,722.8819
728515625,14358.7451171875,15014.8564453125,7

629.896484375,95697.9375,17271.0,17065.66796875,26665.380859375,45863.6484
375,24282.828125,9784.7314453125,4987.37890625,5120.02587890625,4096.24658
203125,7882.31640625,8721.7685546875,16941.759765625,-177.7259979248047,10
36.96142578125,7324.76416015625,3980.96435546875,1270.24853515625,3162.072
265625,344.79046630859375,22653.390625,8756.169921875,15284.759765625,947
1.9775390625,15656.412109375,5169.20556640625,4328.87841796875,8920.433593
75,5365.19287109375,55347.44140625,7795.0595703125,13885.708984375,20347.4
00390625,1227.5938720703125,5778.41357421875,5964.564453125,16874.99414062
5,-115.423095703125,10306.5849609375,14927.2431640625,10967.8955078125,103
71.3955078125,3862.168701171875,15140.76953125,8824.2978515625,3362.900878
90625,2564.439208984375,6931.828125,12005.572265625,1187.315673828125,1595
0.8603515625,9271.4072265625,39931.0703125,12255.3583984375,1306.340454101
5625,11923.8330078125,40572.609375,14656.8642578125,26737.232421875,10556.
4833984375,4581.68896484375,17057.197265625,10263.6181640625,8300.24804687
5,62026.140625,12255.3583984375,4876.71435546875,5979.84326171875,2982.114
501953125,21389.908203125,11832.5146484375,16289.4970703125,11867.4882812
5,11399.4853515625,22612.99609375,7222.544921875,14398.271484375,8140.5473
6328125,57648.9765625,7673.708984375,6124.0966796875,2328.307861328125,110
77.18359375,14774.1015625,30465.099609375,17926.36328125,8495.0224609375,5
965.3212890625,3376.046630859375,17025.931640625,9135.1796875,19540.95312
5,6617.5361328125,14083.68359375,8654.111328125,2700.652587890625,44080.31
640625,62964.64453125,68735.3515625,5221.35498046875,741.5753173828125,902
0.146484375,10694.8642578125,5724.1845703125,76413.359375,7571.5532226562
5,88726.9140625,16199.3203125,95912.4609375,11399.4853515625,16468.9785156
25,3179.58349609375,20347.400390625,2475.491943359375,11468.0390625,18229.
4140625,3364.946044921875,44874.671875,3868.00390625,4148.59521484375,789
4.89404296875,7073.75927734375,23242.47265625,1791.0765380859375,11863.964
84375,10430.4228515625,5620.49560546875,14838.47265625,9667.7919921875,112
16.109375,3930.66650390625,19163.0,18702.529296875,894.783203125,9853.0947
265625,4685.8642578125,31618.015625,13236.634765625,16289.4970703125,2197
6.421875,7696.3447265625,1179.172607421875,5716.49609375,3441.19946289062
5,17211.77734375,10659.5390625,12610.1025390625,11006.9462890625,20577.382
8125,10318.4462890625,15515.2822265625,10108.619140625,46060.16015625,353
8.5166015625,7452.08251953125,16747.5703125,27864.849609375,72968.640625,2
5406.900390625,3065.31005859375,19630.96484375,27860.064453125,9146.123046
875,4455.0185546875,20774.1875,1823.7025146484375,9117.1630859375,10995.51
5625,18886.029296875,16149.08984375,3913.028564453125,6256.92138671875,79
7.753662109375,51798.09765625,15140.76953125,6033.64453125,15321.274414062
5,2798.362060546875,63658.1484375,5724.1845703125,2636.026611328125,-1046.
644287109375,2029.2723388671875,32400.357421875,11538.423828125,47280.5976
5625,23209.134765625,15526.10546875,1110.438232421875,3859.9716796875,-91
5.7093505859375,15503.98046875,6795.95556640625,2979.890380859375,4647.575
1953125,5593.7109375,15851.9345703125,10556.4833984375,46117.87109375,294
4.828369140625,5637.07421875,11054.7080078125,722.8819580078125,4027.58569
3359375,43925.078125,18109.564453125,4135.44482421875,6886.263671875,9960.
0400390625,9784.7314453125,29092.791015625,7558.69091796875,15012.03417968
75,115909.5,3248.5927734375,4537.751953125,16970.494140625,4557.4760742187
5,1018.4276733398438,12179.9140625,7165.74755859375,23949.953125,33606.957
03125,14287.4873046875,15339.201171875,22612.99609375,5493.1904296875,4002
0.19140625,-281.3099670410156,-1228.2255859375,10280.30859375,15464.170898
4375,6913.90185546875,3724.70361328125,12591.037109375,1375.6962890625,113
07.9423828125,65186.828125,23449.775390625,49932.9453125,10761.8427734375,
2961.884033203125,9663.548828125,14603.7470703125,4148.59521484375,1666.89
8681640625,4463.41064453125,4505.20849609375,2104.309814453125,3600.5,325
7.86474609375,7395.29638671875,2216.626953125,14541.96875,47661.96875,598
[illegible]4227.5498046875,9690.404296875,9225.75292968

75,7357.28662109375,7443.2431640625,8895.6708984375,1571.3826904296875,345
6.464111328125,28080.837890625,11769.275390625,7662.095703125,15043.472656
25,4663.1376953125,75732.046875,11939.3427734375,4727.71728515625,16130.75
68359375,20252.537109375,25397.3046875,3211.25537109375,37543.578125,-171.
00885009765625,753.8356323242188,664.6282958984375,21185.5859375,17268.455
078125,741.5753173828125,13546.873046875,467.8045959472656,15090.90039062
5,10667.267578125,17791.650390625,5600.5087890625,6894.2099609375,2449.033
69140625,39592.04296875,30438.2890625,3740.19189453125,22014.30078125,4793
4.59765625,9507.6455078125,58050.5625,15757.646484375,3056.439453125,852.2
520751953125,2347.540771484375,6608.35595703125,20441.181640625,27339.9746
09375,8976.0703125,16915.876953125,44479.99609375,-93.57918548583984,3878.
031005859375,20527.359375,18787.19921875,35045.87890625,9409.3935546875,29
83.558349609375,12021.8564453125,6163.94970703125,32900.61328125,4787.2416
9921875,13738.7783203125,1863.0950927734375,1982.6461181640625,12283.48925
78125,8690.2578125,52735.8671875,11485.8486328125,12143.59375,5276.5483398
4375,84241.5,14997.1484375,821.4973754882812,15202.9560546875,3717.0266113
28125,3385.331787109375,4502.7744140625,9302.34375,17926.36328125,30661.36
328125,4676.35107421875,9227.947265625,55855.3359375,10835.7216796875,5061
8.796875,6446.794921875,87320.2734375,17925.353515625,30949.689453125,1988
9.453125,2821.621826171875,9451.1650390625,15656.412109375,18740.921875,13
339.021484375,19113.6875,4955.791015625,37484.82421875,8723.6357421875,134
23.078125,5710.5810546875,15160.5009765625,2821.621826171875,8908.50683593
75,1072.989990234375,7411.00048828125,72755.9296875,11047.330078125,6061.3
2421875,32399.2578125,6072.86572265625,15507.896484375,3690.4169921875,920
7.001953125,52974.11328125,7231.10009765625,17518.544921875,4783.44726562
5,22583.79296875,76948.53125,9504.3935546875,4605.79296875,33234.1640625,9
049.9326171875,42418.3984375,21372.935546875,9333.5595703125,14078.7138671
875,16106.44921875,8991.2216796875,3705.020751953125,-745.2735595703125,16
327.5263671875,1961.753173828125,1580.62109375,9030.1884765625,13952.44628
90625,3243.7451171875,32655.423828125,5392.3251953125,3984.28369140625,610
6.87109375,5984.42236328125,121517.2109375,12190.283203125,50030.5390625,4
2018.671875,7882.74365234375,9885.091796875,3652.6455078125,8458.997070312
5,61472.96875,27025.470703125,7040.7587890625,4394.3857421875,5007.4736328
125,3699.28125,11640.4375,2593.3271484375,13755.154296875,18374.140625,293
5.5703125,2654.598876953125,16289.4970703125,63048.078125,3830.07495117187
5,28641.126953125,611.7021484375,10741.7109375,454.60009765625,59607.04687
5,34099.75,6948.09716796875,9556.0703125,50645.6484375,5737.3427734375,108
60.03125,-161.6197967529297,12931.23046875,29220.099609375,14834.37109375,
8158.34619140625,11130.7236328125,43225.57421875,22529.076171875,21729.705
078125,20619.41015625,-160.76254272460938,4982.1943359375,13546.873046875,
8029.5556640625,12520.3701171875,19455.244140625,26840.916015625,5077.0571
2890625,19163.0,12897.267578125,5039.55517578125,50664.90625,7859.86767578
125,13624.2080078125,11894.21484375,69916.71875,16536.490234375,4954.36425
78125,9497.6064453125,11830.091796875,46566.79296875,6103.73193359375,554
2.46484375,6753.501953125,33351.73046875,8511.69921875,15556.388671875,24
9.67657470703125,9321.013671875,4008.08935546875,11133.57421875,3243.74511
71875,6522.4765625,3149.140380859375,4569.37060546875,7577.35791015625,272
64.482421875,17171.025390625,10172.123046875,5885.310546875,14078.71386718
75,5303.857421875,13583.7109375,10645.5546875,9471.9775390625,10120.918945
3125,30456.048828125,68618.3828125,13285.212890625,54348.43359375,15458.86
71875,7333.986328125,90016.1015625,95074.765625,23895.298828125,7168.74511
71875,8604.4736328125,29466.998046875,1054.1541748046875,23249.17578125,10
545.5009765625,12452.2900390625,-789.722900390625,11867.48828125,3896.6904
296875,5190.15087890625,12053.0234375,6617.5361328125,15656.412109375,1157
9.99609375,16874.994140625,9074.1435546875,1502.859130859375,19100.4375,25
[illegible]5,2484.0576171875,14326.142578125,3566.92895

5078125,3213.129150390625,17838.064453125,17211.77734375,2756.78515625,402
8.2275390625,24395.427734375,90222.3515625,14958.1123046875,53601.5625,471
74.50390625,43949.640625,10498.61328125,5607.8251953125,70953.5703125,275
0.5263671875,14970.203125,8635.7041015625,31131.544921875,412.437377929687
5,8453.4482421875,8183.74951171875,4685.80908203125,10561.0693359375,1224.
5467529296875,8419.15234375,8088.9599609375,5422.314453125,144934.734375,1
0172.123046875,16874.994140625,16839.212890625,21471.404296875,16199.32031
25,19794.603515625,4002.76513671875,3735.530029296875,19226.31640625,1811
3.052734375,35937.7109375,65088.19140625,55260.7890625,615.408203125,5718.
37548828125,4769.6181640625,19163.0,21619.20703125,18166.8828125,15656.412
109375,4585.7119140625,5899.01806640625,8249.72265625,15312.65234375,2857
0.935546875,25521.21484375,14273.3291015625,7681.7265625,7570.77685546875,
4061.168701171875,6029.2568359375,13130.326171875,19385.150390625,7253.731
4453125,18804.66015625,40174.9296875,12291.3359375,9145.9921875,11771.6816
40625,2298.27099609375,1296.424072265625,59556.09765625,11391.0048828125,1
7516.244140625,6766.1328125,9905.6171875,8872.189453125,26675.908203125,14
819.080078125,7971.4375,10145.76953125,2334.746826171875,23865.921875,1419
4.3505859375,993.7883911132812,9732.4287109375,15352.9208984375,4859.04736
328125,12857.6494140625,9207.001953125,5090.19384765625,46641.26171875,144
2.99365234375,777.3087158203125,4161.76904296875,1003.4268188476562,8275.6
2109375,12015.0986328125,36976.578125,7859.86767578125,15217.740234375,810
5.29345703125,3444.663330078125,10285.8466796875,46241.26171875,37077.1914
0625,8105.0556640625,19129.412109375,25555.216796875,19171.02734375,5113.7
216796875,21210.419921875,40631.140625,10447.0615234375,46864.05859375,743
7.11279296875,2776.304931640625,2328.307861328125,5764.615234375,5139.2031
25,4216.603515625,15773.3134765625,6261.89501953125,1011.7501831054688,155
00.0908203125,15285.1689453125,22042.90625,8808.6044921875,7407.702148437
5,2121.3818359375,23158.521484375,10255.4423828125,2599.303466796875,1000
3.814453125,8231.083984375,10561.0693359375,11086.126953125,18373.06445312
5,4937.49072265625,30806.80859375,-161.6197967529297,8824.2978515625,9391.
1591796875,1621.8656005859375,43819.640625,21295.13671875,2004.9868164062
5,16389.1328125,18687.953125,4256.6787109375,2263.967041015625,53189.66015
625,26954.23046875,8134.25439453125,10659.1328125,5586.537109375,81467.468
75,3531.830322265625,15952.859375,12382.4326171875,44254.48828125,21054.54
4921875,4949.8173828125,4868.07666015625,29859.95703125,4070.63623046875,7
437.11279296875,16432.576171875,690.093017578125,13994.6220703125,56906.94
140625,2410.5224609375,5190.095703125,15321.2744140625,15248.8798828125,11
421.4404296875,15906.3837890625,3483.871826171875,9044.767578125,5660.9877
9296875,1836.2774658203125,12857.6494140625,6299.62255859375,15656.4121093
75,27053.3203125,2240.71337890625,8068.32568359375,5547.7646484375,2008.48
54736328125,17947.994140625,7091.74365234375,12134.537109375,17741.0507812
5,22014.30078125,6410.78857421875,25242.9375,9068.328125,12429.8291015625,
21468.775390625,14621.5927734375,4502.7744140625,9251.279296875,8169.27880
859375,17373.078125,2791.228271484375,3217.605712890625,17633.4296875,-143
8.840087890625,6456.59521484375,56060.01953125,19082.302734375,18845.76562
5,17217.89453125,4816.65185546875,614.84033203125,51009.0,8424.3193359375,
4324.47998046875,8895.6708984375,19009.10546875,-153.90223693847656,2465.5
0634765625,17515.833984375,17231.00390625,4746.20654296875,10945.48828125,
23249.18359375,24024.548828125,30404.923828125,15254.90234375,-2123.924072
265625,5593.7109375,13469.0380859375,6004.91015625,3968.6025390625,25732.8
671875,5280.1025390625,2066.6484375,4565.345703125,10960.453125,17165.2890
625,2941.152099609375,11358.923828125,14193.69921875,27912.984375,3986.271
484375,4273.15673828125,18845.765625,4502.7744140625,87214.6796875,2926.16
455078125,6945.49951171875,4535.71630859375,-1046.644287109375,439.1944580
078125,7439.95263671875,20511.63671875,8060.642578125,11049.0419921875,502
[illegible]6901.1015625,8089.1416015625,10457.145507812

```
5,12810.11328125,5030.2763671875,6097.126953125,3097.418212890625,93808.66
40625,37077.19140625,8486.193359375,12890.5625,23212.978515625,-634.947265
625,3445.075439453125,11800.7626953125,-194.56985473632812,10835.301757812
5,13450.509765625,10087.482421875,18757.275390625,18454.97265625,12381.216
796875,1466.5040283203125,3734.720703125,3153.87158203125,15656.412109375,
10498.345703125,16024.7197265625,42887.6484375,73517.9296875,16739.539062
5,2151.15283203125,4874.2822265625,54728.99609375,8261.98046875,44139.8046
875,5518.892578125,56219.52734375,20303.19921875,3328.23876953125,6632.878
90625,8726.4599609375,13339.021484375,98580.1875,143157.40625,12143.59375'
```

In [124…
```
# custom code to convert the values in bytes format to array

def bytes_2_array(x):

    # makes entire prediction as string and splits based on ','
    l = str(x).split(',')

    # Since the first element contains unwanted characters like (b,',') we r
    l[0] = l[0][2:]
    #same-thing as above remove the unwanted last character (')
    l[-1] = l[-1][:-1]

    # iterating through the list of strings and converting them into float t
    for i in range(len(l)):
        l[i] = float(l[i])

    # converting the list into array
    l = np.array(l).astype('float32')

    # reshape one-dimensional array to two-dimensional array
    return l.reshape(-1,1)
```

In [125…
```
predicted_values_1 = bytes_2_array(predictions1)
```

In [126…
```
predicted_values_1.shape
```

Out[126]:
```
(10000, 1)
```

In [127…
```
predicted_values_2 = bytes_2_array(predictions2)
predicted_values_2.shape
```

Out[127]:
```
(10000, 1)
```

In [128…
```
predicted_values_3 = bytes_2_array(predictions3)
predicted_values_3.shape
```

Out[128]:
```
(10000, 1)
```

In [129…
```
predicted_values_4 = bytes_2_array(predictions4)
predicted_values_4.shape
```

Out[129]:
```
(1618, 1)
```

In [130…

```
predicted_values = np.concatenate((predicted_values_1, predicted_values_2, p
```

In [131…

```
predicted_values.shape
```

Out[131]:

```
(31618, 1)
```

In [132…

```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_errc
from math import sqrt
k = X_test.shape[1]
n = len(X_test)
RMSE = float(format(np.sqrt(mean_squared_error(y_test, predicted_values)),'.
MSE = mean_squared_error(y_test, predicted_values)
MAE = mean_absolute_error(y_test, predicted_values)
r2 = r2_score(y_test, predicted_values)
adj_r2 = 1-(1-r2)*(n-1)/(n-k-1)

print('RMSE =',RMSE, '\nMSE =',MSE, '\nMAE =',MAE, '\nR2 =', r2, '\nAdjustec
```

```
RMSE = 7092.51
MSE = 50303700.0
MAE = 4342.0317
R2 = 0.9001853264770594
Adjusted R2 = 0.8997477514287362
```

In [133…

```
# Delete the end-point

Xgboost_regressor.delete_endpoint()
```

```
INFO:sagemaker:Deleting endpoint configuration with name: sagemaker-xgboost
-2023-09-12-10-08-20-377
INFO:sagemaker:Deleting endpoint with name: sagemaker-xgboost-2023-09-12-10
-08-20-377
```

RESULTS AND DISCUSSION

In this machine learning project, I aimed to forecast weekly retail store sales for a specific department based on historical data. The dataset included information on weekly sales, holidays, and promotional markdowns across 99 departments and 45 different stores.

After training and evaluating the model, I obtained the following performance metrics:

- Root Mean Squared Error (RMSE): 7092.51
- Mean Squared Error (MSE): 50,303,700.0
- Mean Absolute Error (MAE): 4342.0317
- R-squared (R2): 0.9002
- Adjusted R-squared: 0.8997

These metrics provide insights into the accuracy and goodness of fit of our model.

An R2 value of 0.9002 indicates that our model explains approximately 90.02% of the variance in the weekly sales data.

The adjusted R-squared value of 0.8997 accounts for the number of predictors in our model, providing a more robust evaluation of model performance.

CONCLUSION AND RECOMMENDATIONS

This machine learning model has shown strong predictive capabilities, with an R2 value of 0.9002, indicating a high degree of accuracy in forecasting weekly retail store sales. This model can be instrumental in helping stores make informed decisions and optimize their business processes, especially concerning promotional markdowns and holiday sales.

However, there are several avenues for further improvement and exploration:

1. **Feature Engineering**: Further exploration of additional features or feature transformations that may enhance the model's predictive power. Feature selection techniques could help identify the most influential variables.
2. **Hyperparameter Tuning**: Further experiments with different machine learning algorithms and hyperparameter settings could potentially improve model performance further.
3. **Time Series Analysis**: Delving deeper into time series analysis techniques would capture temporal patterns and seasonality in the data.
4. **Cross-Validation**: More robust cross-validation techniques could ensure proper validation of the model's generalization performance and mitigate overfitting.

REFERENCES

- Chen, T., & Guestrin, C. (2016, August). XGBoost: A Scalable Tree Boosting System.
- Csörgő, A., & Bentéjac, C. (2019, November). A Comparative Analysis of XGBoost.
- Su, W., Jiang, F., Shi, C., Wu, D., Liu, L., Li, S., Yuan, Y., & Shi, J. (2023, December). An XGBoost-Based Knowledge Tracing Model.
- Zhang, P., Jia, Y., & Shang, Y. (2022, June). Research and Application of XGBoost in Imbalanced Data.
- Data Source. (n.d.). Retail Dataset. Kaggle. Retrieved from ngh/retaildataset