Sales forecasting for a supply chain

Students:

1.Badescu Alexandru Mihai

2.Buzica Theodor Andrei

3.Gastescu Andrei Razvan

4.Popescu Andreea Daniela

Subject of the study

This study aims to compare the performance of three different Machine Learning algorithms (Random Forest, Linear Regression, and Prophet) in a sales forecasting process. The subject of the analysis is represented by a retail chain, which offers insights into its sales between 2019 and 2022 for all the belonging stores located in Romania. The prediction process is conducted by focusing on the impact of different indicators such as the Consumer Price Index (CPI), inflation rate, unemployment rate, and average salary.

Importance of the study

Sales forecasting offers a wide range of benefits to the organizations.One good example would be that it allows them to plan inventory levels and avoid stockouts or excess inventory. Moreover, it enables supply chain managers to understand and plan for future demand, allowing for better decision-making regarding product development and marketing efforts. This way, the organizations are able to create accurate budgets, enabling better financial planning and resource allocation. Better planning and estimation leads also to a more efficient collaboration with suppliers, distributors, and other partners in the supply chain.
Overall, sales forecasting provides a more reliable and efficient supply chain by allowing organizations to proactively address potential challenges and opportunities, leading to increased customer satisfaction and profitability.

State of the art

Previously, studies have been conducted to predict sales for retail corporations using available historical data. Researchers from Fiji

National University and The University of the South Pacific analyzed Walmart's sales data using big data analytics tools such as HDFS, Hadoop MapReduce, and Apache Spark, with programming languages like Scala, Java, and Python. Their goal was to examine the impact of factors in the dataset on Walmart's sales.The Walmart's Sales dataset offers historical sales data from 45 Walmart stores to predict sales for all weeks in a year. Data includes store location, department, land area, type, holiday weeks, and price markdown data. Macro-indicators such as CPI, unemployment rate, and fuel price are also provided.

In his paper, Kassambara (kassambara, 2018) discusses how to integrate interaction effects with multiple linear regression in R. He attempts to develop an additive model based on two relevant factors by starting with a basic multiple regression model that attempts to forecast sales based on advertising expenditures on Facebook and YouTube (budget for youtube and budget for facebook). In order to develop a regression model, his approach makes the assumption that the impact of YouTube advertising on sales is independent of the impact of Facebook advertising. He notes that there is an interacting relationship between the two predictor variables (youtube and facebook advertising), and this additive model outperforms the conventional regression with an R2 score of 0.98. This project also examines the interactions between the several independent variables in the dataset, such as CPI, unemployment, in, etc., in an effort to determine whether there is a correlation between these variables and monthly unemployment rates.

Michael Crown, a data scientist, analyzed a similar dataset using time series forecasting and non-seasonal ARIMA models for his predictions. ARIMA (AutoRegressive Integrated Moving Average) is a statistical model used for time series analysis and forecasting. It is a combination of three components: autoregression (AR), integration (I), and moving average (MA) and attempts to capture linear relationships between past values of a time series and its current value. He utilized ARIMA modeling to generate one-year weekly forecasts using 2.75 years of sales data, which included features such as store information, department, date, weekly sales, and holiday data. The performance was evaluated by measuring the normalized root-mean-square error (NRMSE).

The use of random forest algorithms to develop predictive models is another extension of predictive approaches pertinent to this topic. According to a study conducted by San Diego State University researchers (Lingjun et al., 2018), this tree-based machine learning approach is preferable to other regression techniques for building predictive models for the higher education industry. In their study, the authors compare the effectiveness of several models, including lasso regression and logistic regression, and highlight the significance of using random forest algorithms with prediction problems in Weka and R. They do this by using a standard classification and regression tree (CART) algorithm along with feature importance.

Many of the authors combined tools and strategies to produce effective models, and they frequently evaluated their results across all models to choose the one that performs the best for the given dataset.

This study uses 2 of the already mentioned methods, namely the linear regression and random forest and an additional one, which is called “The prophet”. The third method was introduced by Facebook (S. J. Taylor & Letham, 2018), originally for forecasting daily data with weekly and yearly seasonality, plus holiday effects.

It is essential to conduct a comparison examination of different models to make sure that the predictions are precise and that their application is not constrained. It is also vital for this study to test out different models because they behave differently depending on the type

and volume of data.

Which is our approach?

Preparation of the dataset

The dataset covers the sales data of a supply chain with 30 nationwide stores, summarized into 18 product categories (such as still water, mineral water, meat, and fish), has been extracted for a random sample of 5000 customers for a period of 4 years (2019– 2022), whereas the granularity of the data is set to month. Additionally, relevant information about customers such as various indicators were obtained:

- average net salary (nationally and at store location levels)
- national and county level unemployment rate
- national inflation rate
- national level IPC (for food, non-food, and services)
- total national level IPC

These indicators were sourced from organizations like the National Institute of Statistics (INS). All tables were combined into a single data frame and negative values from returns and outliers were removed.

Objective

The project's set objectives are to study the performance of 3 Machine Learning algorithms (Linear Regression, Random Forest and The Prophet) in predicting sales, comparing the errors calculated by each and discovering which is the most efficient and suitable algorithm for the generated dataset.
Following, each of the 3 previously mentioned algorithms will be briefly described.
Random Forest is an ensemble learning method for classification and regression in machine learning. It works by combining multiple decision trees to generate a more accurate and stable prediction. The algorithm creates multiple decision trees using random samples of the data and features. Each tree makes a prediction, and the Random Forest combines the results of all trees to produce the final prediction. The algorithm uses a technique called bagging (bootstrap aggregating) to create multiple training sets from the original data, each with replacement. In classification, the final prediction is determined by a majority vote of all the trees. In regression, the final prediction is the average of all the trees' predictions. Random Forest provides a way to overcome the limitations of single decision trees, such as overfitting, by combining the results of multiple trees to produce a more robust prediction. This makes Random Forest a popular and effective algorithm for many applications, especially in areas such as sales forecasting, customer behavior analysis, and credit risk assessment.
Linear Regression is a statistical method used for modeling the linear relationship between a dependent variable and one or more independent variables. It is a supervised learning algorithm commonly used in

regression problems, such as predicting a continuous outcome. In linear regression, the model tries to fit a line through the data points, such that the difference between the observed values and the predicted values is minimized. The line represents the relationship between the independent and dependent variables, and the parameters of the line (the slope and intercept) are estimated from the data. The equation of the line can be used to make predictions for new data points based on their values for the independent variables. Linear regression assumes that the relationship between the independent and dependent variables is linear and additive. It is a simple and fast algorithm that can be used for small datasets, but it may not always provide the best predictions for more complex relationships between variables. Overall, linear regression works by fitting a line to the data that minimizes the difference between the observed values and the values predicted by the line, and uses this line to make predictions for new data points based on their values for the independent variables.
Prophet is a time series forecasting algorithm developed by Facebook for use in business forecasting. It is designed for problems where traditional time series models may not be suitable, such as time series with complex seasonality patterns, trends, and anomalies. Prophet works by decomposing the time series into three main components: trend, seasonality, and holiday effects. The trend component models non-periodic changes in the time series, such as growth or decline. The seasonality component models periodic patterns in the time series, such as daily or weekly patterns. The holiday component models the effects of events that happen at specific times, such as holidays or promotions. Prophet then uses a combination of a piecewise linear model and a Bayesian model to fit these components to the data. The Bayesian model is based on Bayes' theorem, which states that the probability of a hypothesis given the observed data is proportional to the prior probability of the hypothesis multiplied by the likelihood of observing the data given the hypothesis. Bayesian models allow for incorporating prior knowledge, updating beliefs based on new evidence, and predicting future outcomes.The algorithm can automatically identify and handle missing data and outliers, making it suitable for real-world datasets with missing or noisy observations. All in all, Prophet works by decomposing a time series into three main components: trend, seasonality, and holiday effects, and uses a combination of a piecewise linear model and a Bayesian model to fit these components to the data, making it a suitable algorithm for complex time series forecasting problems.

Conclusions

The fact that there is a wide distribution of monthly sales per store, with huge differences over the stores in the dataset is due to the randomized samples taken from the database. Therefore, it seemed to be quite inappropriate to compare stores or to predict sales with this level of granularity. Moreover,the random sample which was used was also inappropriate due to the small variety of customer categories. One can not ensure that this sample is relevant for the entire population.
All these limitations led us to performing the sales forecasting on a product category level, whereas the algorithm with the highest efficiency was Random forest, the highest accuracy level being registered for the "Beer" product (almost 94%).

The chances of obtaining much better results are obviously highly dependent on the training dataset. Thus, a limited training dataset leads to a limited result. On the other hand, with a larger dataset, access to a wider range of analysis options is gained. A good example would be an analysis conducted from a customer and product perspective, which would predict the monthly sales of a particular product by client. Another future goal of this research could be to analyze customer purchasing patterns based on regional sales data. The split of customers by region can benefit the company by enabling them to deliver tailored messages, foster better customer connections, concentrate on profitable regions, and recognize opportunities to enhance products and services for specific regions or customers.
Additionally, the study may examine trends in sales across stores and forecast future trends using sales data. Time series forecasting methods such as ARMA and ARIMA modeling can be explored for predicting future sales for each store and its categories.

In [131…

```
from unidecode import unidecode
import pandas as pd
import numpy as np
from plotly.subplots import make_subplots
import seaborn as sns
import pandas as pd
import plotly.graph_objects as go
import matplotlib.pyplot as plt
from datetime import datetime
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from prophet import Prophet
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from IPython.display import display
from matplotlib.dates import date2num, DateFormatter
from datetime import datetime
```

Importing and processing the input data

In [2]:

```
# Read in the excel file
file = pd.read_excel('sal_mediu.xlsx')
file = file.rename(columns={'Unnamed: 0':'month_id'})

print(file)
```

```
                month_id  202212  202211  202210  202209  202208  202207  \
0               National    4141    4141    4008    4003    3933    3975
1                   Arad    3469    3469    3469    3439    3300    3354
2                  Argeş    3545    3545    3545    3523    3475    3715
3                  Bacău    3556    3556    3556    3652    3634    3585
4                  Bihor    3251    3251    3251    3249    3192    3180
5                 Braşov    3819    3819    3819    3754    3719    3688
6                  Buzău    3265    3265    3265    3233    3194    3233
7                   Cluj    4849    4849    4849    4791    4730    4756
8              Constanţa    3419    3419    3419    3377    3418    3448
9              Dâmboviţa    3106    3106    3106    3081    3035    3093
10                  Dolj    3466    3466    3466    3449    3490    3432
11                Galaţi    3502    3502    3502    3503    3459    3435
12             Hunedoara    3064    3064    3064    3060    2950    2918
13                  Iaşi    3907    3907    3907    3910    3852    3894
14             Maramureş    3198    3198    3198    3163    3095    3106
15                 Mureş    3876    3876    3876    3872    3799    3819
```

```
16                Neamț    3054    3054    3054    3078    3058    3019
17              Prahova    3558    3558    3558    3656    3452    3446
18           Satu Mare    3259    3259    3259    3247    3181    3170
19               Sibiu    4248    4248    4248    4152    4097    4119
20             Suceava    3094    3094    3094    3071    2985    3013
21               Timiş    4307    4307    4307    4335    4314    4278
22  Municipiul București    5220    5220    5220    5243    5119    5228

    202206  202205  202204  ...  201910  201909  201908  201907  201906  \
0     3977    3928    3967  ...    3116    3082    3044    3119    3142
1     3370    3295    3266  ...    2825    2789    2718    2791    2832
2     3544    3420    3541  ...    2805    2877    2793    3052    2954
3     3582    3533    3403  ...    2830    2820    2839    2908    2835
4     3184    3183    3132  ...    2502    2482    2453    2482    2458
5     3821    3629    3604  ...    2948    2923    2861    2890    2980
6     3229    3084    3086  ...    2580    2521    2492    2516    2587
7     4712    4864    4619  ...    3524    3470    3440    3498    3471
8     3428    3332    3426  ...    2774    2712    2667    2697    2821
9     3138    3052    3065  ...    2572    2560    2530    2589    2784
10    3411    3446    3351  ...    2811    2758    2768    2794    2774
11    3405    3553    3448  ...    2930    2894    2788    2901    2969
12    2992    3084    2971  ...    2549    2481    2439    2493    2546
13    3996    3943    3794  ...    3130    3098    3026    3141    3327
14    3126    3139    3112  ...    2512    2470    2441    2460    2563
15    3844    3652    3852  ...    2904    2862    2884    2873    3028
16    3037    3008    3023  ...    2593    2562    2670    2562    2592
17    3485    3483    3501  ...    2866    2850    2811    2870    2977
18    3248    3166    3148  ...    2643    2558    2514    2548    2611
19    4027    4296    4119  ...    3083    3032    2976    3038    3028
20    3090    3061    2975  ...    2503    2477    2445    2513    2510
21    4424    4257    4463  ...    3129    3097    3097    3116    3244
22    5182    5093    5287  ...    3982    3964    3901    4019    3986

    201905  201904  201903  201902  201901
0     3101    3115    3075    2933    2936
1     2749    2784    2737    2642    2636
2     2933    2851    2728    2656    2638
3     2792    2746    2745    2640    2627
4     2502    2455    2416    2396    2368
5     2891    2920    2902    2762    2802
6     2519    2532    2543    2433    2414
7     3508    3414    3478    3240    3236
8     2646    2701    2624    2536    2546
9     2619    2654    2560    2526    2558
10    2703    2725    2675    2651    2633
11    2807    2885    2775    2714    2703
12    2507    2518    2477    2443    2434
13    3092    3107    3091    2970    2978
14    2488    2433    2443    2378    2401
15    2806    2926    2792    2687    2669
16    2539    2525    2527    2436    2450
17    2847    2882    2906    2751    2729
18    2586    2487    2553    2440    2445
19    3199    3052    2981    2806    2918
20    2523    2508    2477    2423    2406
21    3088    3230    3035    2904    2915
22    3994    4063    4024    3765    3763

[23 rows x 49 columns]
```

In [3]:

```
melt_file = pd.melt(file, id_vars=file.columns[0], value_vars=file.columns[1:])

melt_file = melt_file.rename(columns={'month_id':'county'})
melt_file = melt_file.rename(columns={'variable':'month_id'})
melt_file = melt_file.rename(columns={'value':'net_average_salary'})
```

```
melt_file = melt_file[melt_file['county']=='National']

print(melt_file)
```

```
        county month_id  net_average_salary
0     National   202212                4141
23    National   202211                4141
46    National   202210                4008
69    National   202209                4003
92    National   202208                3933
115   National   202207                3975
138   National   202206                3977
161   National   202205                3928
184   National   202204                3967
207   National   202203                3937
230   National   202202                3721
253   National   202201                3698
276   National   202112                3879
299   National   202111                3645
322   National   202110                3544
345   National   202109                3517
368   National   202108                3487
391   National   202107                3545
414   National   202106                3541
437   National   202105                3492
460   National   202104                3561
483   National   202103                3547
506   National   202102                3365
529   National   202101                3395
552   National   202012                3620
575   National   202011                3411
598   National   202010                3343
621   National   202009                3321
644   National   202008                3275
667   National   202007                3372
690   National   202006                3298
713   National   202005                3179
736   National   202004                3182
759   National   202003                3294
782   National   202002                3202
805   National   202001                3189
828   National   201912                3340
851   National   201911                3179
874   National   201910                3116
897   National   201909                3082
920   National   201908                3044
943   National   201907                3119
966   National   201906                3142
989   National   201905                3101
1012  National   201904                3115
1035  National   201903                3075
1058  National   201902                2933
1081  National   201901                2936
```

In [4]:
```
somaj = pd.read_excel('somaj_final.xlsx')

print(somaj)
```

```
               month_id  202212  202211  202210  202209  202208  202207  \
0              National    3.04    2.96    2.88    2.56    2.56    2.55
1                 Argeş    3.59    3.59    3.56    3.53    3.10    3.06
2                 Bacău    2.89    2.89    2.84    2.91    2.73    2.78
3                 Bihor    1.26    1.26    1.22    1.17    1.02    1.03
4               Braşov     2.31    2.31    2.13    2.07    1.98    2.01
5                 Buzău    5.28    5.28    5.18    5.13    4.35    5.01
6                  Cluj    1.24    1.24    1.21    1.17    1.09    1.07
```

```
7              Constanţa    2.24    2.24    2.00    1.78    1.60    1.60
8              Dâmboviţa    2.83    2.83    2.84    2.76    2.44    2.64
9                   Dolj    7.40    7.40    7.29    6.95    5.64    5.54
10               Galaţi    6.13    6.13    5.62    6.27    5.78    5.75
11            Hunedoara    3.99    3.99    4.03    3.74    3.39    3.33
12                 Iaşi    3.17    3.17    3.11    3.05    2.75    2.67
13            Maramureş    2.02    2.02    1.93    1.89    1.64    1.59
14                Mureş    3.23    3.23    3.09    2.86    2.49    2.41
15                Neamţ    4.07    4.07    4.00    3.74    3.02    3.03
16              Prahova    2.25    2.25    2.17    2.17    2.00    1.99
17            Satu Mare    3.75    3.75    3.64    3.51    3.03    3.00
18                Sibiu    2.31    2.31    2.14    1.94    1.75    1.71
19              Suceava    5.51    5.51    5.42    5.26    4.35    4.38
20                Timiş    0.84    0.84    0.87    0.86    0.78    0.78
21  Municipiul Bucureşti    1.05    1.05    1.06    1.06    1.08    1.09

    202206  202205  202204  ...  201910  201909  201908  201907  201906  \
0     2.55    2.57    2.64  ...    3.00    3.03    3.01    2.95    2.95
1     3.03    3.02    3.05  ...    2.91    2.99    3.02    2.93    2.85
2     2.87    2.89    2.99  ...    5.10    5.16    5.08    5.16    5.10
3     1.06    1.09    1.06  ...    1.25    1.30    1.35    1.29    1.25
4     1.94    1.95    2.05  ...    2.14    2.18    2.21    2.20    2.21
5     4.90    4.79    4.71  ...    6.72    6.81    6.73    6.72    6.60
6     1.07    1.10    1.10  ...    1.26    1.30    1.31    1.27    1.23
7     1.56    1.54    1.75  ...    2.34    1.98    1.96    1.89    1.93
8     2.57    2.53    2.43  ...    3.48    3.32    3.43    3.56    3.45
9     5.56    5.55    5.63  ...    6.72    6.77    6.92    6.86    6.88
10    5.55    5.50    5.44  ...    5.59    5.81    5.82    5.79    5.63
11    3.70    3.74    3.69  ...    2.93    3.01    3.24    3.28    3.18
12    2.63    2.63    2.71  ...    2.89    3.00    3.01    2.85    2.90
13    1.67    1.67    1.68  ...    2.75    2.87    3.29    3.24    3.21
14    2.38    2.33    2.36  ...    2.66    2.55    2.57    2.52    2.57
15    3.12    3.08    3.03  ...    4.06    4.18    4.28    4.31    4.03
16    1.94    1.94    1.95  ...    2.36    2.37    2.40    2.19    2.20
17    2.95    2.98    3.02  ...    2.35    2.31    2.25    2.27    2.28
18    1.66    1.71    1.76  ...    1.73    1.65    1.65    1.65    1.66
19    4.39    4.40    4.34  ...    4.83    4.92    5.03    5.05    4.64
20    0.70    0.67    0.66  ...    0.83    0.91    0.91    0.84    0.83
21    1.10    1.10    1.10  ...    1.29    1.30    1.29    1.30    1.30

    201905  201904  201903  201902  201901
0     3.00    3.19    3.31    3.32    1.42
1     2.85    2.83    3.07    3.23    3.27
2     5.10    5.40    5.73    5.94    5.91
3     1.25    1.16    1.15    1.21    1.34
4     2.21    2.09    2.20    2.28    2.27
5     6.60    6.71    6.87    7.10    7.17
6     1.23    3.00    1.29    1.30    1.33
7     1.93    2.11    2.64    2.77    2.77
8     3.45    3.49    3.68    3.74    3.83
9     6.88    7.11    7.43    7.59    7.56
10    5.63    5.55    5.91    6.10    6.09
11    3.18    3.28    3.53    3.59    3.64
12    2.90    2.94    2.95    3.01    3.04
13    3.21    3.10    3.20    3.19    3.21
14    2.57    2.63    2.79    2.92    2.98
15    4.03    4.05    4.28    4.67    4.68
16    2.20    2.25    2.44    2.49    2.54
17    2.28    2.44    2.58    2.65    2.64
18    1.66    1.73    1.81    1.87    1.82
19    4.64    4.63    4.93    5.20    5.22
20    0.83    0.69    0.74    0.80    0.82
21    1.30    1.30    1.31    1.31    1.31

[22 rows x 49 columns]
```

In [5]:
```
melt_somaj = pd.melt(somaj, id_vars=somaj.columns[0], value_vars=somaj.columns[1:])

melt_somaj = melt_somaj.rename(columns={'month_id':'county'})
melt_somaj = melt_somaj.rename(columns={'variable':'month_id'})
melt_somaj = melt_somaj.rename(columns={'value':'unnemployment'})

melt_somaj = melt_somaj[melt_somaj['county']=='National']

print(melt_somaj)
```

```
      county month_id  unnemployment
0   National   202212           3.04
22  National   202211           2.96
44  National   202210           2.88
66  National   202209           2.56
88  National   202208           2.56
110 National   202207           2.55
132 National   202206           2.55
154 National   202205           2.57
176 National   202204           2.64
198 National   202203           2.67
220 National   202202           2.68
242 National   202201           2.69
264 National   202112           2.72
286 National   202111           2.76
308 National   202110           2.85
330 National   202109           2.93
352 National   202108           2.96
374 National   202107           3.00
396 National   202106           3.06
418 National   202105           3.16
440 National   202104           3.33
462 National   202103           3.35
484 National   202102           3.34
506 National   202101           3.38
528 National   202012           3.32
550 National   202011           3.27
572 National   202010           3.26
594 National   202009           3.30
616 National   202008           3.02
638 National   202007           3.00
660 National   202006           2.87
682 National   202005           2.90
704 National   202004           2.88
726 National   202003           2.95
748 National   202002           2.98
770 National   202001           2.97
792 National   201912           2.98
814 National   201911           2.98
836 National   201910           3.00
858 National   201909           3.03
880 National   201908           3.01
902 National   201907           2.95
924 National   201906           2.95
946 National   201905           3.00
968 National   201904           3.19
990 National   201903           3.31
1012 National  201902           3.32
1034 National  201901           1.42
```

In [6]:
```
final = pd.read_excel('vanzari_judet_inflatie.xlsx')
print(final)
```

```
       home_store_id     county    cust_key  month_id denumire_produs   colli  \
0                 11  Bucuresti  1100007106    201907       apa_plata  12.000
1                 11  Bucuresti  1100007106    202003       apa_plata   4.000
```

```
2                11  Bucuresti  1100008007    201907       apa_plata  12.000
3                11  Bucuresti  1100008007    201908       apa_plata  12.000
4                11  Bucuresti  1100008007    201911       apa_plata   6.000
...             ...        ...         ...       ...             ...     ...
46441            58       Cluj  5800012771    202206           cafea  84.000
46442            58       Cluj  5800012771    202207           lapte   3.000
46443            58       Cluj  5800012771    202207       apa_plata  12.000
46444            58       Cluj  5800012771    202207         carnati   1.135
46445            58       Cluj  5800012771    202207      ulei_soare   6.000

       sales  TOTAL IPC (%)  IPC Marfuri alimentare (%)  \
0      23.04          99.80                       99.34
1      13.64         100.50                      101.46
2      30.00          99.80                       99.34
3      30.00         100.06                       99.71
4      15.00         100.23                      100.43
...      ...            ...                         ...
46441  15.78         100.76                      100.62
46442   5.64         100.89                      100.92
46443  19.41         100.89                      100.92
46444  15.61         100.89                      100.92
46445  50.34         100.89                      100.92

       IPC Marfuri nealimentare (%)  IPC Servicii (%)  \
0                             99.98            100.10
1                             99.91            100.35
2                             99.98            100.10
3                            100.22            100.25
4                            100.12            100.15
...                             ...               ...
46441                        100.92            100.54
46442                        100.87            100.88
46443                        100.87            100.88
46444                        100.87            100.88
46445                        100.87            100.88

       Inflatie(de la o luna la alta)  Inflatie anuala
0                               -0.20              3.8
1                                0.50              2.6
2                               -0.20              3.8
3                                0.06              3.8
4                                0.23              3.8
...                               ...              ...
46441                            0.76             13.8
46442                            0.89             13.8
46443                            0.89             13.8
46444                            0.89             13.8
46445                            0.89             13.8

[46446 rows x 13 columns]
```

Concatenation of the tables

In [7]:
```
merge_final_somaj = pd.merge(final, melt_somaj, on=['month_id'])
merge_final_somaj = merge_final_somaj.drop('county_y', axis=1)
print(merge_final_somaj)
merge_final_somaj.to_csv('merge_final_somaj.csv')
```

```
       home_store_id   county_x    cust_key month_id denumire_produs    colli  \
0                 11  Bucuresti  1100007106   201907       apa_plata   12.000
1                 11  Bucuresti  1100008007   201907       apa_plata   12.000
2                 11  Bucuresti  1100009984   201907           cafea    6.000
3                 11  Bucuresti  1100009984   201907       apa_plata  336.000
4                 11  Bucuresti  1100012905   201907         carnati    0.394
```

```
...              ...         ...          ...      ...             ...      ...
46441             56  Dâmbovița  5600015963   202104          legume    1.000
46442             56  Dâmbovița  5600015963   202104      ulei_soare   20.000
46443             56  Dâmbovița  5600016169   202104             oua  240.000
46444             58       Cluj  5800012550   202104           lapte    2.000
46445             58       Cluj  5800012550   202104             oua   30.000

        sales  TOTAL IPC (%)  IPC Marfuri alimentare (%)  \
0       23.04          99.80                       99.34
1       30.00          99.80                       99.34
2        9.06          99.80                       99.34
3      568.40          99.80                       99.34
4       13.37          99.80                       99.34
...       ...            ...                         ...
46441   18.15         100.45                      100.45
46442  143.20         100.45                      100.45
46443   89.10         100.45                      100.45
46444    5.89         100.45                      100.45
46445   10.50         100.45                      100.45

       IPC Marfuri nealimentare (%)  IPC Servicii (%)  \
0                             99.98             100.1
1                             99.98             100.1
2                             99.98             100.1
3                             99.98             100.1
4                             99.98             100.1
...                             ...               ...
46441                        100.47             100.4
46442                        100.47             100.4
46443                        100.47             100.4
46444                        100.47             100.4
46445                        100.47             100.4

       Inflatie(de la o luna la alta)  Inflatie anuala  unnemployment
0                               -0.20              3.8           2.95
1                               -0.20              3.8           2.95
2                               -0.20              3.8           2.95
3                               -0.20              3.8           2.95
4                               -0.20              3.8           2.95
...                               ...              ...            ...
46441                            0.45              5.1           3.33
46442                            0.45              5.1           3.33
46443                            0.45              5.1           3.33
46444                            0.45              5.1           3.33
46445                            0.45              5.1           3.33

[46446 rows x 14 columns]
```

In [8]:
```
merge_final_somaj_salmed = pd.merge(merge_final_somaj, melt_file, on=['month_id'])
merge_final_somaj_salmed = merge_final_somaj_salmed.drop('county', axis=1)
print(merge_final_somaj_salmed)
merge_final_somaj_salmed.to_csv('merge_final_somaj_salmed.csv')
```

```
       home_store_id   county_x    cust_key month_id denumire_produs    colli  \
0                 11  Bucuresti  1100007106   201907       apa_plata   12.000
1                 11  Bucuresti  1100008007   201907       apa_plata   12.000
2                 11  Bucuresti  1100009984   201907           cafea    6.000
3                 11  Bucuresti  1100009984   201907       apa_plata  336.000
4                 11  Bucuresti  1100012905   201907         carnati    0.394
...              ...         ...         ...      ...             ...      ...
46441             56  Dâmbovița  5600015963   202104          legume    1.000
46442             56  Dâmbovița  5600015963   202104      ulei_soare   20.000
46443             56  Dâmbovița  5600016169   202104             oua  240.000
46444             58       Cluj  5800012550   202104           lapte    2.000
46445             58       Cluj  5800012550   202104             oua   30.000
```

```
        sales  TOTAL IPC (%)  IPC Marfuri alimentare (%)  \
0       23.04          99.80                       99.34
1       30.00          99.80                       99.34
2        9.06          99.80                       99.34
3      568.40          99.80                       99.34
4       13.37          99.80                       99.34
...       ...            ...                         ...
46441   18.15         100.45                      100.45
46442  143.20         100.45                      100.45
46443   89.10         100.45                      100.45
46444    5.89         100.45                      100.45
46445   10.50         100.45                      100.45

       IPC Marfuri nealimentare (%)  IPC Servicii (%)  \
0                             99.98             100.1
1                             99.98             100.1
2                             99.98             100.1
3                             99.98             100.1
4                             99.98             100.1
...                             ...               ...
46441                        100.47             100.4
46442                        100.47             100.4
46443                        100.47             100.4
46444                        100.47             100.4
46445                        100.47             100.4

       Inflatie(de la o luna la alta)  Inflatie anuala  unnemployment  \
0                               -0.20              3.8           2.95
1                               -0.20              3.8           2.95
2                               -0.20              3.8           2.95
3                               -0.20              3.8           2.95
4                               -0.20              3.8           2.95
...                               ...              ...            ...
46441                            0.45              5.1           3.33
46442                            0.45              5.1           3.33
46443                            0.45              5.1           3.33
46444                            0.45              5.1           3.33
46445                            0.45              5.1           3.33

       net_average_salary
0                    3119
1                    3119
2                    3119
3                    3119
4                    3119
...                   ...
46441                3561
46442                3561
46443                3561
46444                3561
46445                3561

[46446 rows x 15 columns]
```

In [9]:
```
df = merge_final_somaj_salmed
df = df.rename(columns={'county_x': 'county'})

df['store_county'] = df['home_store_id'].astype(str) + '-' + df['county']
print(df)
```

```
       home_store_id     county    cust_key month_id denumire_produs    colli  \
0                 11  Bucuresti  1100007106   201907       apa_plata   12.000
1                 11  Bucuresti  1100008007   201907       apa_plata   12.000
2                 11  Bucuresti  1100009984   201907           cafea    6.000
3                 11  Bucuresti  1100009984   201907       apa_plata  336.000
4                 11  Bucuresti  1100012905   201907         carnati    0.394
```

```
...              ...         ...          ...      ...             ...      ...
46441             56  Dâmbovița  5600015963   202104          legume    1.000
46442             56  Dâmbovița  5600015963   202104      ulei_soare   20.000
46443             56  Dâmbovița  5600016169   202104             oua  240.000
46444             58       Cluj  5800012550   202104           lapte    2.000
46445             58       Cluj  5800012550   202104             oua   30.000

        sales  TOTAL IPC (%)  IPC Marfuri alimentare (%)  \
0       23.04          99.80                       99.34
1       30.00          99.80                       99.34
2        9.06          99.80                       99.34
3      568.40          99.80                       99.34
4       13.37          99.80                       99.34
...       ...            ...                         ...
46441   18.15         100.45                      100.45
46442  143.20         100.45                      100.45
46443   89.10         100.45                      100.45
46444    5.89         100.45                      100.45
46445   10.50         100.45                      100.45

       IPC Marfuri nealimentare (%)  IPC Servicii (%)  \
0                             99.98             100.1
1                             99.98             100.1
2                             99.98             100.1
3                             99.98             100.1
4                             99.98             100.1
...                             ...               ...
46441                        100.47             100.4
46442                        100.47             100.4
46443                        100.47             100.4
46444                        100.47             100.4
46445                        100.47             100.4

       Inflatie(de la o luna la alta)  Inflatie anuala  unnemployment  \
0                               -0.20              3.8           2.95
1                               -0.20              3.8           2.95
2                               -0.20              3.8           2.95
3                               -0.20              3.8           2.95
4                               -0.20              3.8           2.95
...                               ...              ...            ...
46441                            0.45              5.1           3.33
46442                            0.45              5.1           3.33
46443                            0.45              5.1           3.33
46444                            0.45              5.1           3.33
46445                            0.45              5.1           3.33

       net_average_salary  store_county
0                    3119  11-Bucuresti
1                    3119  11-Bucuresti
2                    3119  11-Bucuresti
3                    3119  11-Bucuresti
4                    3119  11-Bucuresti
...                   ...           ...
46441                3561  56-Dâmbovița
46442                3561  56-Dâmbovița
46443                3561  56-Dâmbovița
46444                3561       58-Cluj
46445                3561       58-Cluj

[46446 rows x 16 columns]
```

In [10]:

```
#group by magazin si month id

grouped_sales = df.groupby(['store_county', 'month_id'], group_keys=False)['sales'].sum(

print(grouped_sales)
```

```
store_county  month_id
11-Bucuresti  201901      1918.90
              201902      2173.78
              201903      2481.80
              201904      3553.16
              201905      3398.04
                           ...
58-Cluj       202205       203.01
              202206       109.91
              202207       226.42
              202208        98.93
              202212        72.32
Name: sales, Length: 1365, dtype: float64
```

In [11]:

```
# convert the month_id column to string
df['month_id'] = df['month_id'].astype(str)

# format the month_id column to the desired format
df['month_id'] = pd.to_datetime(df['month_id'], format='%Y%m').dt.strftime('%m-%Y')

# Print the data set
print(df)
```

```
       home_store_id     county    cust_key month_id denumire_produs    colli  \
0                 11  Bucuresti  1100007106  07-2019       apa_plata   12.000
1                 11  Bucuresti  1100008007  07-2019       apa_plata   12.000
2                 11  Bucuresti  1100009984  07-2019           cafea    6.000
3                 11  Bucuresti  1100009984  07-2019       apa_plata  336.000
4                 11  Bucuresti  1100012905  07-2019         carnati    0.394
...              ...        ...         ...      ...             ...      ...
46441             56  Dâmbovița  5600015963  04-2021          legume    1.000
46442             56  Dâmbovița  5600015963  04-2021      ulei_soare   20.000
46443             56  Dâmbovița  5600016169  04-2021            oua  240.000
46444             58       Cluj  5800012550  04-2021           lapte    2.000
46445             58       Cluj  5800012550  04-2021             oua   30.000

        sales  TOTAL IPC (%)  IPC Marfuri alimentare (%)  \
0       23.04          99.80                       99.34
1       30.00          99.80                       99.34
2        9.06          99.80                       99.34
3      568.40          99.80                       99.34
4       13.37          99.80                       99.34
...       ...            ...                         ...
46441   18.15         100.45                      100.45
46442  143.20         100.45                      100.45
46443   89.10         100.45                      100.45
46444    5.89         100.45                      100.45
46445   10.50         100.45                      100.45

       IPC Marfuri nealimentare (%)  IPC Servicii (%)  \
0                             99.98             100.1
1                             99.98             100.1
2                             99.98             100.1
3                             99.98             100.1
4                             99.98             100.1
...                             ...               ...
46441                        100.47             100.4
46442                        100.47             100.4
46443                        100.47             100.4
46444                        100.47             100.4
46445                        100.47             100.4

       Inflatie(de la o luna la alta)  Inflatie anuala  unnemployment  \
0                               -0.20              3.8           2.95
1                               -0.20              3.8           2.95
```

```
2                              -0.20             3.8           2.95
3                              -0.20             3.8           2.95
4                              -0.20             3.8           2.95
...                              ...             ...            ...
46441                           0.45             5.1           3.33
46442                           0.45             5.1           3.33
46443                           0.45             5.1           3.33
46444                           0.45             5.1           3.33
46445                           0.45             5.1           3.33

       net_average_salary  store_county
0                    3119  11-Bucuresti
1                    3119  11-Bucuresti
2                    3119  11-Bucuresti
3                    3119  11-Bucuresti
4                    3119  11-Bucuresti
...                   ...           ...
46441                3561  56-Dâmbovița
46442                3561  56-Dâmbovița
46443                3561  56-Dâmbovița
46444                3561       58-Cluj
46445                3561       58-Cluj

[46446 rows x 16 columns]
```

In [12]:
```
# Sort the dataframe by the 'date' column in ascending order
df = df.sort_values(by='month_id')
print(df)
```

```
       home_store_id     county    cust_key month_id denumire_produs    colli  \
29712             25  Maramures  2500029737  01-2019       apa_plata    6.000
29719             25  Maramures  2500511731  01-2019           lapte  103.000
29720             25  Maramures  2500511731  01-2019          legume   15.000
29721             25  Maramures  2500511731  01-2019       apa_plata   13.000
29722             25  Maramures  2500511731  01-2019         carnati    1.042
...              ...        ...         ...      ...             ...      ...
28681             22    Prahova  2200821895  12-2022       apa_plata    5.000
28680             22    Prahova  2200821895  12-2022           lapte   12.000
28679             22    Prahova  2200821895  12-2022           pizza    1.000
28708             23      Mures  2300035033  12-2022       apa_plata   47.000
28465             18  Bucuresti  1800517168  12-2022            orez   18.000

        sales  TOTAL IPC (%)  IPC Marfuri alimentare (%)  \
29712   12.60         100.83                      101.30
29719  220.73         100.83                      101.30
29720   25.70         100.83                      101.30
29721   26.34         100.83                      101.30
29722   11.62         100.83                      101.30
...       ...            ...                         ...
28681   17.60         100.37                      101.26
28680   58.68         100.37                      101.26
28679    9.95         100.37                      101.26
28708  141.02         100.37                      101.26
28465   93.96         100.37                      101.26

       IPC Marfuri nealimentare (%)  IPC Servicii (%)  \
29712                        100.63            100.57
29719                        100.63            100.57
29720                        100.63            100.57
29721                        100.63            100.57
29722                        100.63            100.57
...                             ...               ...
28681                         99.68            100.67
28680                         99.68            100.67
28679                         99.68            100.67
28708                         99.68            100.67
```

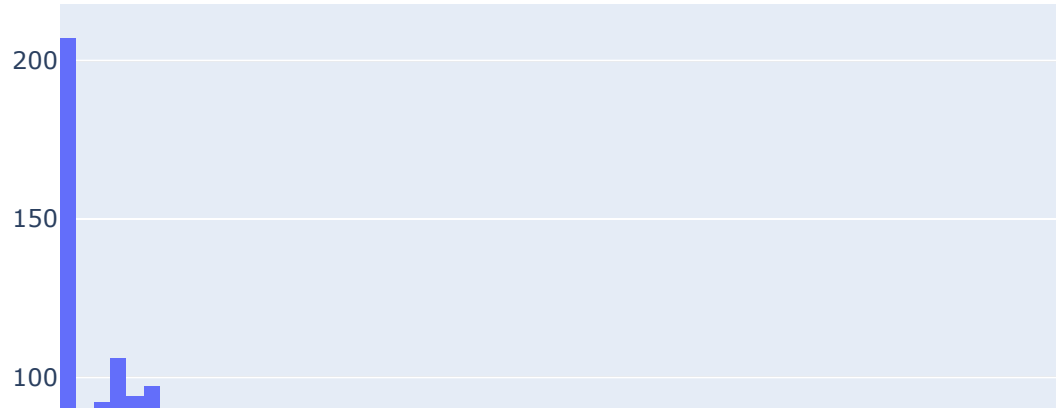
```
28465                          99.68            100.67

       Inflatie(de la o luna la alta)  Inflatie anuala  unnemployment  \
29712                            0.83              3.8           1.42
29719                            0.83              3.8           1.42
29720                            0.83              3.8           1.42
29721                            0.83              3.8           1.42
29722                            0.83              3.8           1.42
...                               ...              ...            ...
28681                            0.37             13.8           3.04
28680                            0.37             13.8           3.04
28679                            0.37             13.8           3.04
28708                            0.37             13.8           3.04
28465                            0.37             13.8           3.04

       net_average_salary  store_county
29712                2936  25-Maramures
29719                2936  25-Maramures
29720                2936  25-Maramures
29721                2936  25-Maramures
29722                2936  25-Maramures
...                   ...           ...
28681                4141    22-Prahova
28680                4141    22-Prahova
28679                4141    22-Prahova
28708                4141      23-Mures
28465                4141  18-Bucuresti

[46446 rows x 16 columns]
```

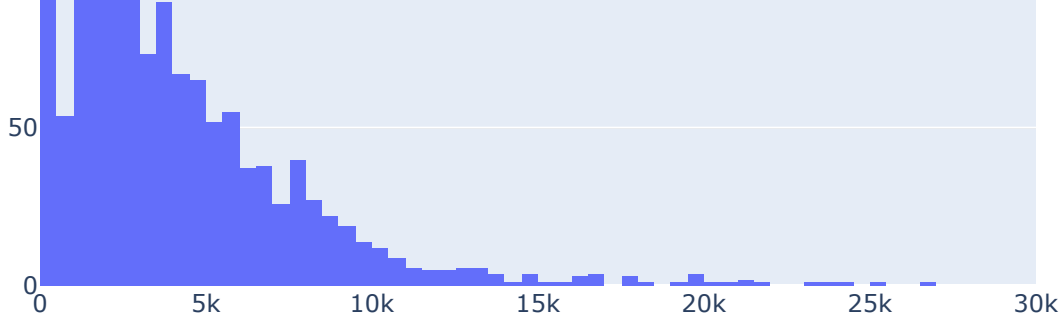
Data exploration

The Histogram below describes the frequency of monthly sales per store.

OX Axis represents the montly sales per store OY Axis represents the frequency of these sales for the given period. The high frequency of the low value total sales denotes that the random sample extracted from the database is too small, therefore irrelevant for this analysis.

In [13]:
```
data = [go.Histogram(x=grouped_sales)]
layout = go.Layout(xaxis=dict(range=[0, 30000]))
fig = go.Figure(data=data, layout=layout)
fig.show()
```

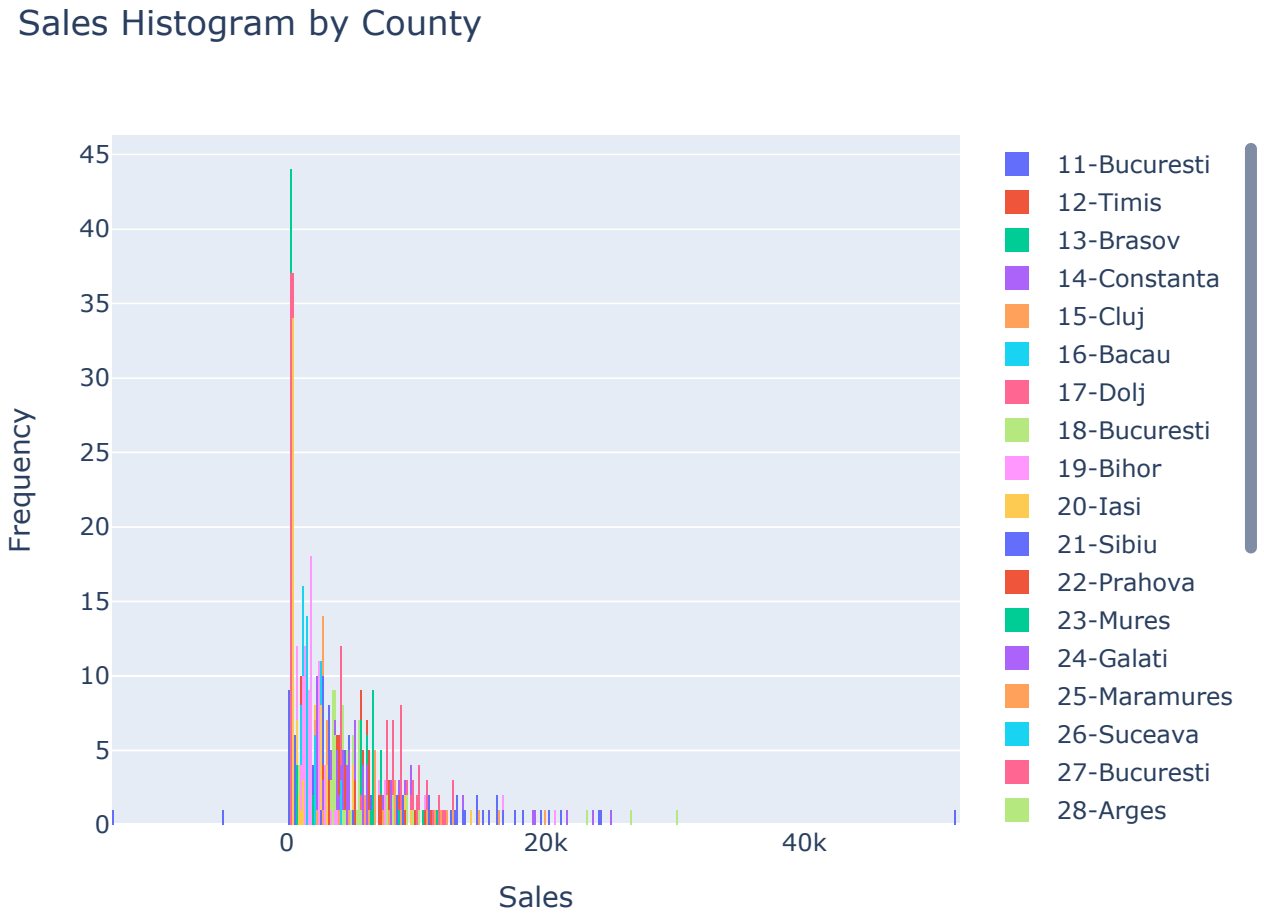




The following histogram leads the same analysis but on a more granular level, namely by store.

This plot displays clearly the outliers and also the high discrepancies between the stores in terms of sales frequency per month. This view helped us decide to renounce on conducting the analysis from a store perspective.

In [14]:
```
counties = grouped_sales.index.get_level_values(0).unique()
data = [go.Histogram(x=grouped_sales.loc[county], name=county) for county in counties]

layout = go.Layout(title='Sales Histogram by County', xaxis_title='Sales', yaxis_title='
fig = go.Figure(data=data, layout=layout)
fig.show()
```



The negative outlier is caused by the returned products.

In [15]:
```
# Remove negative values from the "sales" column
df = df[df["sales"] >= 0]

# Remove outliers using the interquartile range method
q1 = df["sales"].quantile(0.25)
q3 = df["sales"].quantile(0.75)
iqr = q3 - q1
lower_bound = q1 - (1.5 * iqr)
upper_bound = q3 + (1.5 * iqr)
df = df[(df["sales"] > lower_bound) & (df["sales"] < upper_bound)]

# Save the cleaned data set to a new file
df.to_csv("cleaned_data.csv", index=False)
```

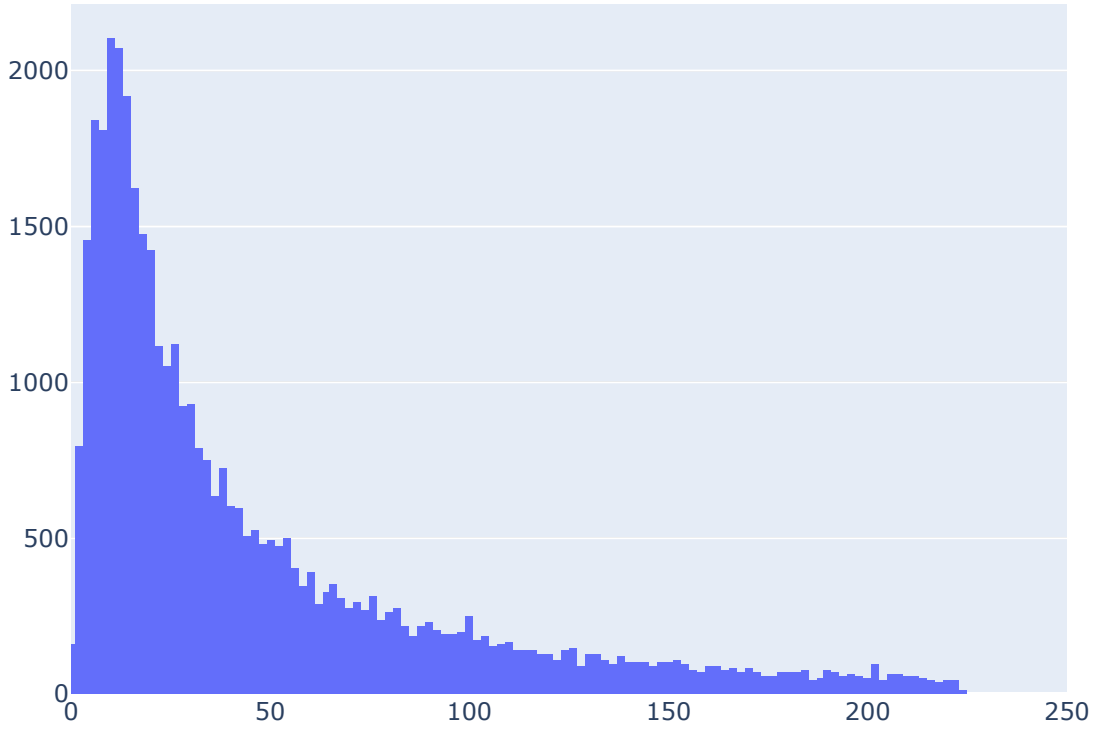
In [16]:
```
print(df)
```

```
       home_store_id     county    cust_key month_id denumire_produs    colli  \
29712             25  Maramures  2500029737  01-2019       apa_plata    6.000
29719             25  Maramures  2500511731  01-2019           lapte  103.000
29720             25  Maramures  2500511731  01-2019          legume   15.000
29721             25  Maramures  2500511731  01-2019       apa_plata   13.000
29722             25  Maramures  2500511731  01-2019         carnati    1.042
...              ...        ...         ...      ...             ...      ...
28681             22    Prahova  2200821895  12-2022       apa_plata    5.000
28680             22    Prahova  2200821895  12-2022           lapte   12.000
28679             22    Prahova  2200821895  12-2022           pizza    1.000
28708             23      Mures  2300035033  12-2022       apa_plata   47.000
28465             18  Bucuresti  1800517168  12-2022            orez   18.000

        sales  TOTAL IPC (%)  IPC Marfuri alimentare (%)  \
29712   12.60         100.83                      101.30
29719  220.73         100.83                      101.30
29720   25.70         100.83                      101.30
29721   26.34         100.83                      101.30
29722   11.62         100.83                      101.30
...       ...            ...                         ...
28681   17.60         100.37                      101.26
28680   58.68         100.37                      101.26
28679    9.95         100.37                      101.26
28708  141.02         100.37                      101.26
28465   93.96         100.37                      101.26

       IPC Marfuri nealimentare (%)  IPC Servicii (%)  \
29712                        100.63            100.57
29719                        100.63            100.57
29720                        100.63            100.57
29721                        100.63            100.57
29722                        100.63            100.57
...                             ...               ...
28681                         99.68            100.67
28680                         99.68            100.67
28679                         99.68            100.67
28708                         99.68            100.67
28465                         99.68            100.67

       Inflatie(de la o luna la alta)  Inflatie anuala  unnemployment  \
29712                            0.83              3.8           1.42
29719                            0.83              3.8           1.42
29720                            0.83              3.8           1.42
29721                            0.83              3.8           1.42
29722                            0.83              3.8           1.42
...                               ...              ...            ...
28681                            0.37             13.8           3.04
28680                            0.37             13.8           3.04
28679                            0.37             13.8           3.04
```

```
28708                          0.37            13.8            3.04
28465                          0.37            13.8            3.04

       net_average_salary  store_county
29712                2936  25-Maramures
29719                2936  25-Maramures
29720                2936  25-Maramures
29721                2936  25-Maramures
29722                2936  25-Maramures
...                   ...           ...
28681                4141    22-Prahova
28680                4141    22-Prahova
28679                4141    22-Prahova
28708                4141      23-Mures
28465                4141  18-Bucuresti

[40775 rows x 16 columns]
```

The following histogram reflects the client behaviour (The frequency of monthly purchases per client) At a first glance, the histogram shows an odd behaviour, where most of the purchases range between 0 and 50 lei. This unrealistic behaviour is probably caused by the limited random sample, which was used.

In [17]:
```
data = [go.Histogram(x=df['sales'])]
layout = go.Layout(xaxis=dict(range=[0, 250]))
fig = go.Figure(data=data, layout=layout)
fig.show()
```



In [18]:
```
df['month_id'] = pd.to_datetime(df['month_id'], format='%m-%Y')
```

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 40775 entries, 29712 to 28465
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   home_store_id                   40775 non-null  int64
 1   county                          40775 non-null  object
 2   cust_key                        40775 non-null  int64
 3   month_id                        40775 non-null  datetime64[ns]
 4   denumire_produs                 40775 non-null  object
 5   colli                           40775 non-null  float64
 6   sales                           40775 non-null  float64
 7   TOTAL IPC (%)                   40775 non-null  float64
 8   IPC Marfuri alimentare (%)      40775 non-null  float64
 9   IPC Marfuri nealimentare (%)    40775 non-null  float64
 10  IPC Servicii (%)                40775 non-null  float64
 11  Inflatie(de la o luna la alta)  40775 non-null  float64
 12  Inflatie anuala                 40775 non-null  float64
 13  unnemployment                   40775 non-null  float64
 14  net_average_salary              40775 non-null  int64
 15  store_county                    40775 non-null  object
dtypes: datetime64[ns](1), float64(9), int64(3), object(3)
memory usage: 5.3+ MB
```

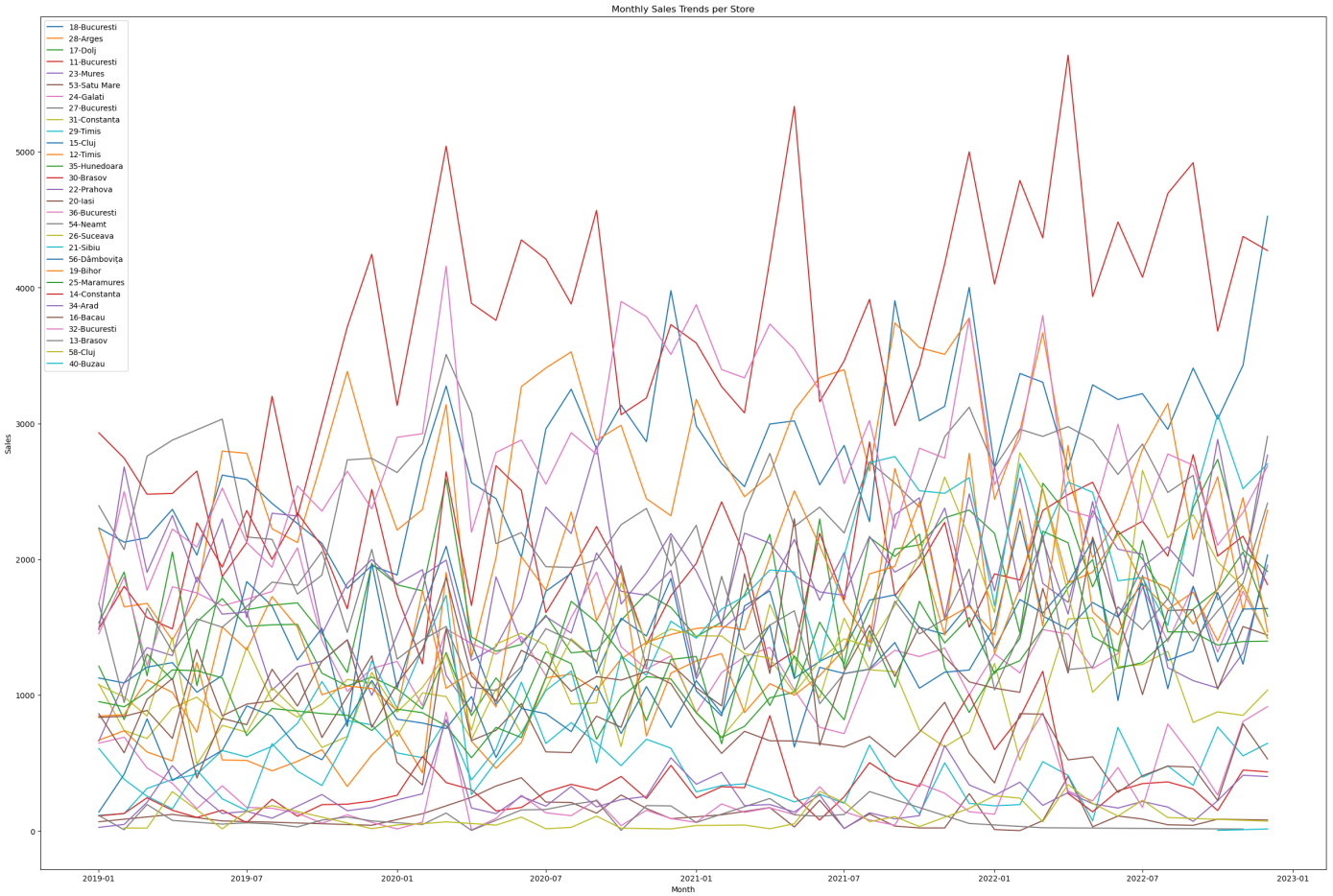
In [20]:
```
# Group by store_id and month_id and sum the sales
store_monthly_sales = df.groupby(["store_county", "month_id"])["sales"].sum().reset_inde

store_monthly_sales = store_monthly_sales.sort_values(by='sales', ascending=False)

# Print the store_monthly_sales dataframe
print(store_monthly_sales)
```

```
      store_county   month_id    sales
183   14-Constanta 2022-04-01  5711.81
172   14-Constanta 2021-05-01  5335.49
158   14-Constanta 2020-03-01  5042.44
179   14-Constanta 2021-12-01  5000.37
188   14-Constanta 2022-09-01  4920.58
...            ...        ...      ...
1167  36-Bucuresti 2020-04-01     7.63
1258      54-Neamt 2020-10-01     4.49
1200      40-Buzau 2022-10-01     4.30
1253      54-Neamt 2020-04-01     4.25
1230  53-Satu Mare 2022-02-01     2.73

[1354 rows x 3 columns]
```

The following line chart is a further proof that the store based approach for analysis is not appropiate. (The variation is too high from store to store.)

In [21]:
```
store_monthly_sales = store_monthly_sales.sort_values(by='month_id', ascending=True)
plt.figure(figsize=(30,20))

# Create a line chart of the sales trends over each month for each store
for store in store_monthly_sales["store_county"].unique():
    store_data = store_monthly_sales[store_monthly_sales["store_county"] == store]
    plt.plot(store_data["month_id"], store_data["sales"], label=store)

# Add axis labels and a chart title
plt.xlabel("Month")
plt.ylabel("Sales")
plt.title("Monthly Sales Trends per Store")
plt.legend()
```

```
# Show the chart
plt.show()
```



Further on we decided to adopt a different approach, looking at the data from a different perspective: By product.

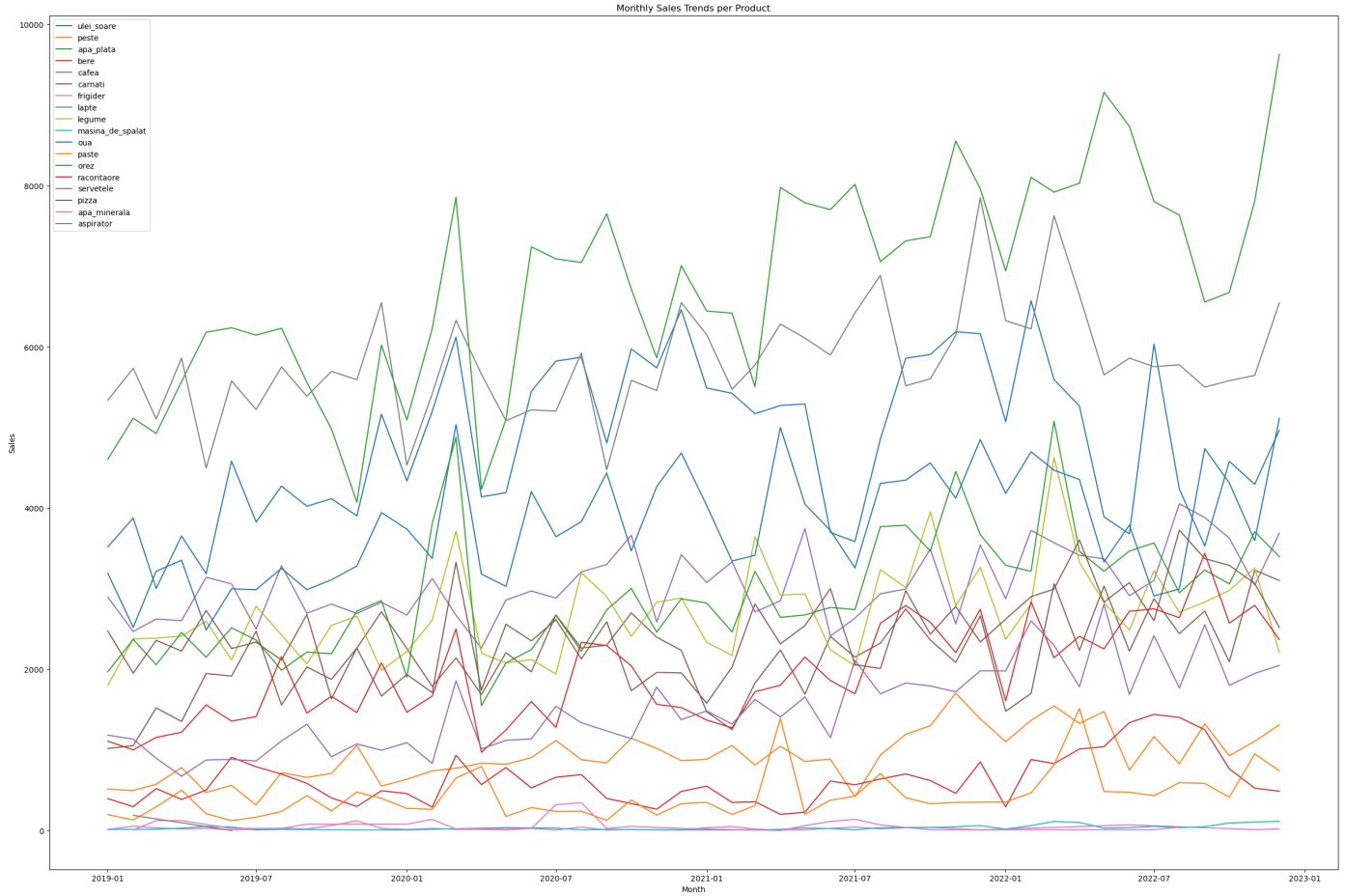
In [22]:
```
# Group by store_id and month_id and sum the sales
product_monthly_sales = df.groupby(["denumire_produs", "month_id"])["sales"].sum().reset

product_monthly_sales = product_monthly_sales.sort_values(by='month_id', ascending=False

# Print the store_monthly_sales dataframe
print(product_monthly_sales)
```

```
      denumire_produs   month_id    sales
770         ulei_soare 2022-12-01  4967.01
578              peste 2022-12-01  1311.05
71          apa_plata 2022-12-01  9630.35
121              bere 2022-12-01   487.94
169             cafea 2022-12-01  3688.86
..                ...        ...      ...
435               oua 2019-01-01  3191.59
723         ulei_soare 2019-01-01  3519.02
254              lapte 2019-01-01  5339.90
302             legume 2019-01-01  1802.61
350  masina_de_spalat 2019-01-01    14.84

[771 rows x 3 columns]
```

The following line chart shows the sales evolution per product this time. Even though the variations are still very high from product to product, when focusing at one specific product, there is not that much of

varience (orders of magnitude) as compared to the variation in the client and store cases. It seems more realistic to direct our approach toward a product-oriented granularity, as opposed to client- or store-related granularity because it is possible to control the product population as opposed to the client population.

In [23]:
```
plt.figure(figsize=(30,20))

# Create a line chart of the sales trends over each month for each store
for product in product_monthly_sales["denumire_produs"].unique():
    product_data = product_monthly_sales[product_monthly_sales["denumire_produs"] == pro
    plt.plot(product_data["month_id"], product_data["sales"], label=product)

# Add axis labels and a chart title
plt.xlabel("Month")
plt.ylabel("Sales")
plt.title("Monthly Sales Trends per Product")
plt.legend()



# Show the chart
plt.show()
```



In [24]:
```
print(df)

df.to_csv("df2.csv", index=False)

cleaned_data_2 = df.groupby(['month_id', 'denumire_produs', 'TOTAL IPC (%)', 'IPC Marfur

print(cleaned_data_2)
cleaned_data_2.to_csv("cleaned_data_2.csv", index=False)
```

```
       home_store_id     county    cust_key   month_id denumire_produs  \
29712             25  Maramures  2500029737 2019-01-01       apa_plata
```

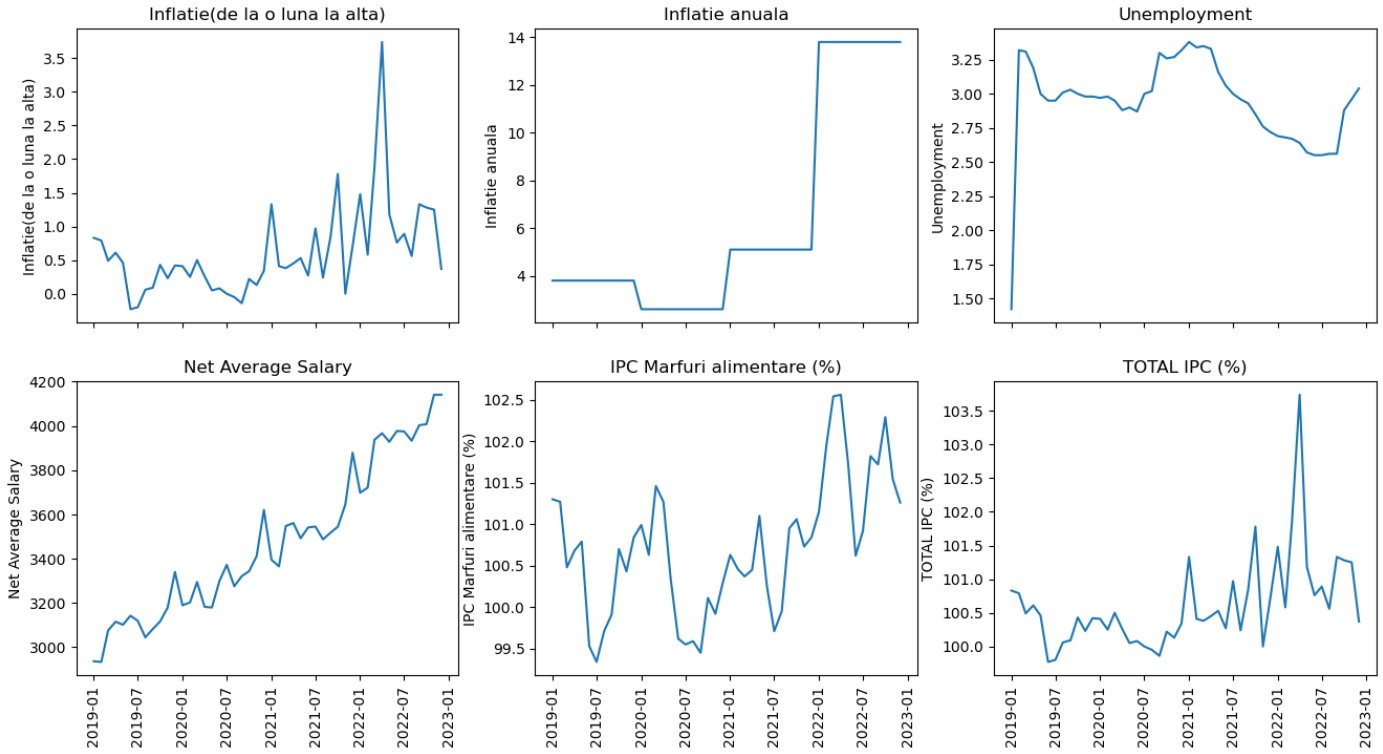
```
29719            25  Maramures  2500511731 2019-01-01          lapte
29720            25  Maramures  2500511731 2019-01-01         legume
29721            25  Maramures  2500511731 2019-01-01      apa_plata
29722            25  Maramures  2500511731 2019-01-01        carnati
...             ...         ...         ...        ...            ...
28681            22    Prahova  2200821895 2022-12-01      apa_plata
28680            22    Prahova  2200821895 2022-12-01          lapte
28679            22    Prahova  2200821895 2022-12-01          pizza
28708            23      Mures  2300035033 2022-12-01      apa_plata
28465            18  Bucuresti  1800517168 2022-12-01           orez

         colli   sales  TOTAL IPC (%)  IPC Marfuri alimentare (%)  \
29712    6.000   12.60         100.83                      101.30
29719  103.000  220.73         100.83                      101.30
29720   15.000   25.70         100.83                      101.30
29721   13.000   26.34         100.83                      101.30
29722    1.042   11.62         100.83                      101.30
...        ...     ...            ...                         ...
28681    5.000   17.60         100.37                      101.26
28680   12.000   58.68         100.37                      101.26
28679    1.000    9.95         100.37                      101.26
28708   47.000  141.02         100.37                      101.26
28465   18.000   93.96         100.37                      101.26

       IPC Marfuri nealimentare (%)  IPC Servicii (%)  \
29712                        100.63            100.57
29719                        100.63            100.57
29720                        100.63            100.57
29721                        100.63            100.57
29722                        100.63            100.57
...                             ...               ...
28681                         99.68            100.67
28680                         99.68            100.67
28679                         99.68            100.67
28708                         99.68            100.67
28465                         99.68            100.67

       Inflatie(de la o luna la alta)  Inflatie anuala  unnemployment  \
29712                            0.83              3.8           1.42
29719                            0.83              3.8           1.42
29720                            0.83              3.8           1.42
29721                            0.83              3.8           1.42
29722                            0.83              3.8           1.42
...                               ...              ...            ...
28681                            0.37             13.8           3.04
28680                            0.37             13.8           3.04
28679                            0.37             13.8           3.04
28708                            0.37             13.8           3.04
28465                            0.37             13.8           3.04

       net_average_salary  store_county
29712                2936  25-Maramures
29719                2936  25-Maramures
29720                2936  25-Maramures
29721                2936  25-Maramures
29722                2936  25-Maramures
...                   ...           ...
28681                4141    22-Prahova
28680                4141    22-Prahova
28679                4141    22-Prahova
28708                4141      23-Mures
28465                4141  18-Bucuresti

[40775 rows x 16 columns]
      month_id denumire_produs  TOTAL IPC (%)  IPC Marfuri alimentare (%)  \
0   2019-01-01       apa_plata         100.83                      101.30
```

```
1   2019-01-01            bere         100.83                      101.30
2   2019-01-01           cafea         100.83                      101.30
3   2019-01-01         carnati         100.83                      101.30
4   2019-01-01        frigider         100.83                      101.30
..         ...             ...            ...                         ...
766 2022-12-01           peste         100.37                      101.26
767 2022-12-01           pizza         100.37                      101.26
768 2022-12-01     racoritaore         100.37                      101.26
769 2022-12-01       servetele         100.37                      101.26
770 2022-12-01      ulei_soare         100.37                      101.26

     IPC Marfuri nealimentare (%)  IPC Servicii (%)  \
0                          100.63            100.57
1                          100.63            100.57
2                          100.63            100.57
3                          100.63            100.57
4                          100.63            100.57
..                            ...               ...
766                         99.68            100.67
767                         99.68            100.67
768                         99.68            100.67
769                         99.68            100.67
770                         99.68            100.67

     Inflatie(de la o luna la alta)  Inflatie anuala  unnemployment  \
0                              0.83              3.8           1.42
1                              0.83              3.8           1.42
2                              0.83              3.8           1.42
3                              0.83              3.8           1.42
4                              0.83              3.8           1.42
..                              ...              ...            ...
766                            0.37             13.8           3.04
767                            0.37             13.8           3.04
768                            0.37             13.8           3.04
769                            0.37             13.8           3.04
770                            0.37             13.8           3.04

     net_average_salary    sales
0                  2936  4607.36
1                  2936   395.56
2                  2936  2896.35
3                  2936  1017.35
4                  2936    13.84
..                  ...      ...
766                4141  1311.05
767                4141  2520.87
768                4141  2370.11
769                4141  2047.38
770                4141  4967.01

[771 rows x 11 columns]
```

The following charts reflect the variation of the indexes, which more or less influence the sales.

In [25]:
```
# Create a figure with 2 rows and 3 columns
fig, axs = plt.subplots(2,3,figsize=(16,8))


# Plot the first line graph in the top left subplot
axs[0, 0].plot(cleaned_data_2['month_id'], cleaned_data_2['Inflatie(de la o luna la alta
axs[0, 0].set_ylabel('Inflatie(de la o luna la alta)')
axs[0, 0].set_title('Inflatie(de la o luna la alta)')

# Plot the second line graph in the top middle subplot
axs[0, 1].plot(cleaned_data_2['month_id'], cleaned_data_2['Inflatie anuala'])
```

```
axs[0, 1].set_ylabel('Inflatie anuala')
axs[0, 1].set_title('Inflatie anuala')

# Plot the third line graph in the top right subplot
axs[0, 2].plot(cleaned_data_2['month_id'], cleaned_data_2['unnemployment'])
axs[0, 2].set_ylabel('Unemployment')
axs[0, 2].set_title('Unemployment')

# Plot the fourth line graph in the bottom left subplot
axs[1, 0].plot(cleaned_data_2['month_id'], cleaned_data_2['net_average_salary'])
axs[1, 0].set_ylabel('Net Average Salary')
axs[1, 0].set_title('Net Average Salary')

# Plot the fifth line graph in the bottom middle subplot
axs[1, 1].plot(cleaned_data_2['month_id'], cleaned_data_2['IPC Marfuri alimentare (%)'])
axs[1, 1].set_ylabel('IPC Marfuri alimentare (%)')
axs[1, 1].set_title('IPC Marfuri alimentare (%)')

# Plot the sixth line graph in the bottom right subplot
axs[1, 2].plot(cleaned_data_2['month_id'], cleaned_data_2['TOTAL IPC (%)'])
axs[1, 2].set_ylabel('TOTAL IPC (%)')
axs[1, 2].set_title('TOTAL IPC (%)')
# Formating x-axis for all subplots
for ax in axs.flat:
    ax.xaxis_date()
    ax.xaxis.set_major_formatter(DateFormatter("%Y-%m"))
    ax.xaxis.set_tick_params(rotation=90)

# Remove the x-tick labels from all but the bottom subplot
for ax in axs[:-1, :].flat:
    ax.set_xticklabels([])

plt.show()
```



In order to have a comprehensive understanding of the charts plotted above, let's interpret them simultaneously, along the time frame, examining each of the four years in detail.

Before proceeding, it is important to note that the net average salary shows a general upward trend throughout the entire analyzed period, from nearly 3000 lei to about 4200 lei. This seems to be the

most consistent development among the indicators analyzed.

In 2019, the inflation rate was relatively low (its trend appeared to decline in the first half of the year and increase in the second half). This year, we also observe that indicators such as Food products CPI, Total CPI, and net average salary reached their lowest values. This makes sense given that the unemployment rate hit its peak (3.4) in the first quarter of the year and then showed a downward trend until the end of the year.

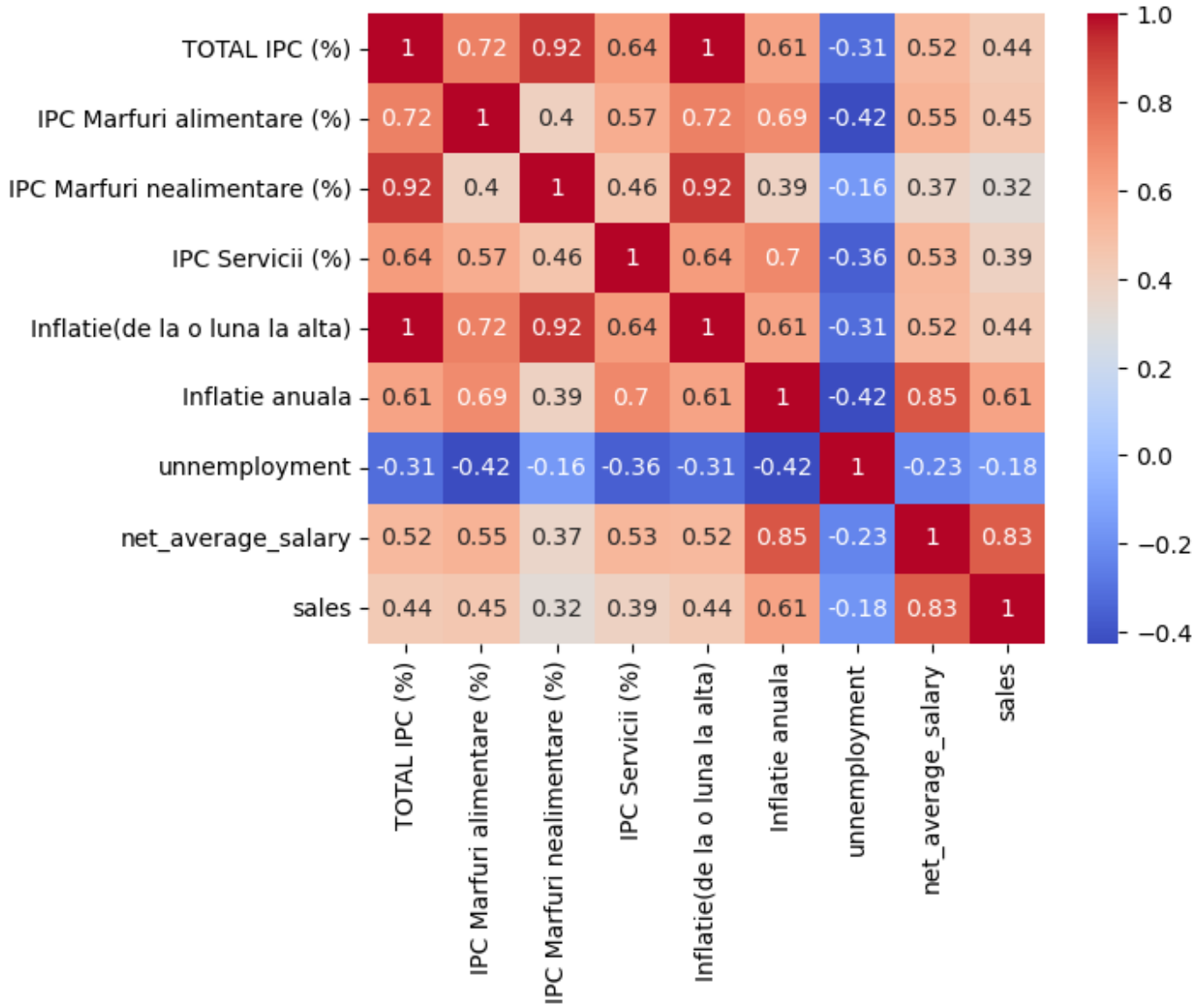
In 2020, the lowest inflation rate (2%) of the entire analyzed period was recorded. This year showed some significant changes: a sharp increase in the unemployment rate to near its highest level, which was obviously due to the peak of the pandemic, and a sharp decrease in the food products CPI.

Regarding 2022, we observe that the three strongly correlated indicators, such as food products CPI, Total CPI, and annual inflation rate, reached their highest values: 102.5%, 103.5%, and 14%. Although the lowest unemployment rate was noted in mid-2022, the second half of the year saw a sharp increase in this indicator.

Correlation Matrix

The main focus in interpreting the correlation matrix lays on the correlation betwen sales and all the other indicators. From a first glance we can notice that the strongest correlated indicators to sales are the following, presented in a descending rate order: Net average salary, annual inflation, food products CPI, total CPI, etc. So far these correlations turn out to follow the same logic as in the real life. The value of the negative correlation between the unemployment rate and sales is just a further proof of the previous statement.

In [26]:
```
cleaned_data_3 = cleaned_data_2.groupby(['month_id', 'TOTAL IPC (%)', 'IPC Marfuri alime

corr_matrix = cleaned_data_3.corr()

import seaborn as sns
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()

#cleaned_data_2_corr = cleaned_data_2[["denumire_produs","TOTAL IPC (%)","IPC Marfuri al
#corr_matrix = cleaned_data_2_corr.corr()
```



Machine Learning

In the following section we followed the same steps for each product category:

-Sorting the dateframe based on the product
-Generating vizualizations of the dataframe
-Splitting the dataset into 2 parts: The Training Dataset and the Testing dataset. The training dataset reflects 42 months out of the total 48 and the training dataset focuses on the rest.
-Applying the 3 different Machine Learning algorithms:
    Linear Regression (LR)
    Random Forest (RF)
    Prophet (PR)

Apa plata dataframe

In [27]:
```
# Create DataFrame to store all the errors
columns = ['product', 'LR-MAE', 'LR-MSE', 'LR-RMSE', 'RF-MAE', 'RF-MSE', 'RF-RMSE', 'PR-
errors_df = pd.DataFrame(columns=columns)
```

```
# Filter the initial DataFrame
```

In [28]:
```
product = 'apa_plata'
filter_df = df.loc[df['denumire_produs'] == product]
filter_df = filter_df.drop(columns=['colli', 'store_county'])
#print(apa_plata_df)

# Group by month_id and aggregate the other columns
filter_df_sales = filter_df.groupby("month_id").agg({'sales': 'sum','Inflatie anuala': '
filter_df_sales = filter_df_sales.sort_values(by='month_id', ascending=True)

# Print the apa_plata_sales dataframe
print(filter_df_sales)
```

month_id	sales	Inflatie anuala	unnemployment	\
2019-01-01	4607.36	3.8	1.42	
2019-02-01	5115.91	3.8	3.32	
2019-03-01	4926.30	3.8	3.31	
2019-04-01	5570.24	3.8	3.19	
2019-05-01	6182.06	3.8	3.00	
2019-06-01	6237.20	3.8	2.95	
2019-07-01	6146.88	3.8	2.95	
2019-08-01	6231.62	3.8	3.01	
2019-09-01	5559.33	3.8	3.03	
2019-10-01	4983.65	3.8	3.00	
2019-11-01	4072.02	3.8	2.98	
2019-12-01	6023.52	3.8	2.98	
2020-01-01	5094.68	2.6	2.97	
2020-02-01	6226.18	2.6	2.98	
2020-03-01	7858.21	2.6	2.95	
2020-04-01	4232.11	2.6	2.88	
2020-05-01	5099.82	2.6	2.90	
2020-06-01	7241.63	2.6	2.87	
2020-07-01	7092.75	2.6	3.00	
2020-08-01	7047.26	2.6	3.02	
2020-09-01	7651.21	2.6	3.30	
2020-10-01	6717.88	2.6	3.26	
2020-11-01	5864.32	2.6	3.27	
2020-12-01	7008.20	2.6	3.32	
2021-01-01	6445.91	5.1	3.38	
2021-02-01	6418.96	5.1	3.34	
2021-03-01	5504.75	5.1	3.35	
2021-04-01	7978.51	5.1	3.33	
2021-05-01	7787.91	5.1	3.16	
2021-06-01	7703.45	5.1	3.06	
2021-07-01	8017.40	5.1	3.00	
2021-08-01	7057.91	5.1	2.96	
2021-09-01	7316.42	5.1	2.93	
2021-10-01	7367.81	5.1	2.85	
2021-11-01	8555.37	5.1	2.76	
2021-12-01	7963.32	5.1	2.72	
2022-01-01	6943.12	13.8	2.69	
2022-02-01	8103.99	13.8	2.68	
2022-03-01	7920.40	13.8	2.67	
2022-04-01	8031.31	13.8	2.64	
2022-05-01	9161.03	13.8	2.57	
2022-06-01	8736.13	13.8	2.55	
2022-07-01	7802.99	13.8	2.55	
2022-08-01	7636.66	13.8	2.56	
2022-09-01	6556.32	13.8	2.56	
2022-10-01	6676.34	13.8	2.88	
2022-11-01	7806.97	13.8	2.96	
2022-12-01	9630.35	13.8	3.04	

month_id	Inflatie(de la o luna la alta)	net_average_salary	\
2019-01-01	0.83	2936.0	

month_id		
2019-02-01	0.79	2933.0
2019-03-01	0.49	3075.0
2019-04-01	0.61	3115.0
2019-05-01	0.46	3101.0
2019-06-01	-0.23	3142.0
2019-07-01	-0.20	3119.0
2019-08-01	0.06	3044.0
2019-09-01	0.09	3082.0
2019-10-01	0.43	3116.0
2019-11-01	0.23	3179.0
2019-12-01	0.42	3340.0
2020-01-01	0.41	3189.0
2020-02-01	0.25	3202.0
2020-03-01	0.50	3294.0
2020-04-01	0.26	3182.0
2020-05-01	0.05	3179.0
2020-06-01	0.08	3298.0
2020-07-01	0.00	3372.0
2020-08-01	-0.05	3275.0
2020-09-01	-0.14	3321.0
2020-10-01	0.22	3343.0
2020-11-01	0.13	3411.0
2020-12-01	0.34	3620.0
2021-01-01	1.33	3395.0
2021-02-01	0.41	3365.0
2021-03-01	0.38	3547.0
2021-04-01	0.45	3561.0
2021-05-01	0.53	3492.0
2021-06-01	0.27	3541.0
2021-07-01	0.97	3545.0
2021-08-01	0.24	3487.0
2021-09-01	0.84	3517.0
2021-10-01	1.78	3544.0
2021-11-01	0.00	3645.0
2021-12-01	0.71	3879.0
2022-01-01	1.48	3698.0
2022-02-01	0.58	3721.0
2022-03-01	1.88	3937.0
2022-04-01	3.74	3967.0
2022-05-01	1.18	3928.0
2022-06-01	0.76	3977.0
2022-07-01	0.89	3975.0
2022-08-01	0.56	3933.0
2022-09-01	1.33	4003.0
2022-10-01	1.28	4008.0
2022-11-01	1.25	4141.0
2022-12-01	0.37	4141.0

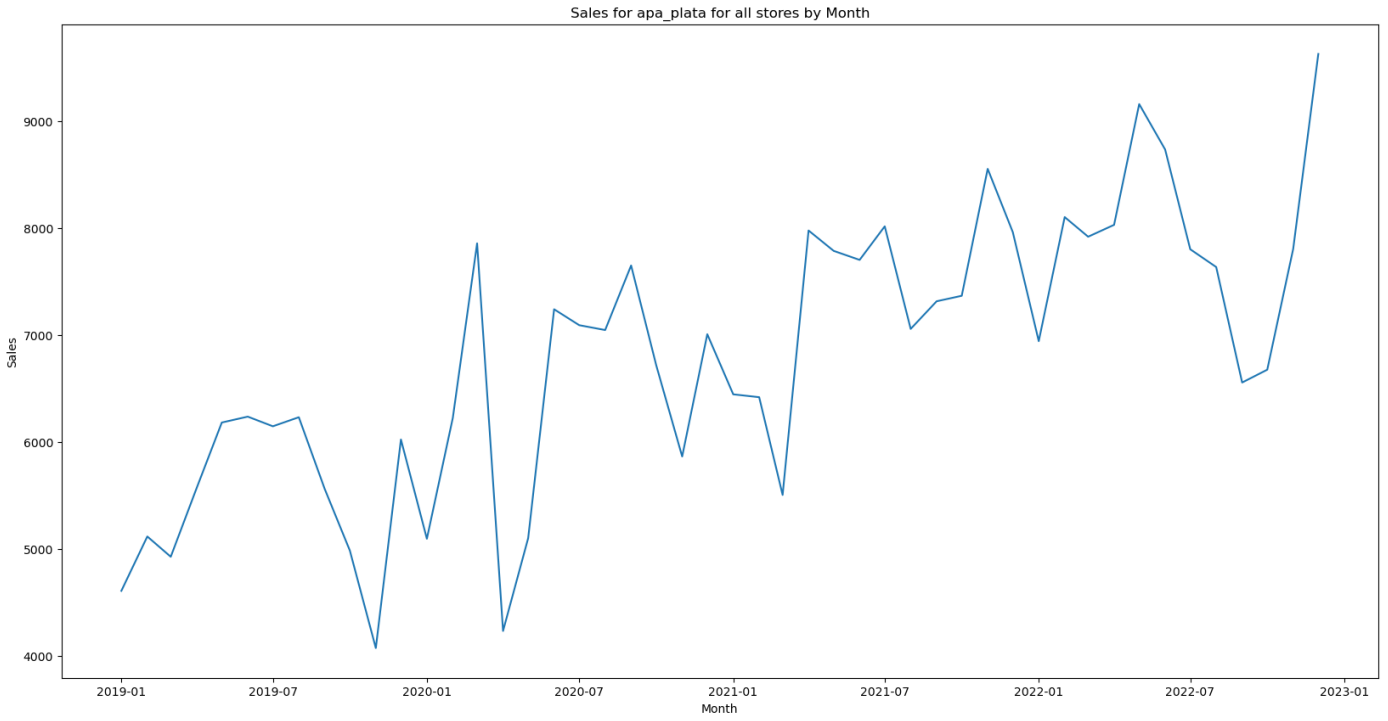
month_id	IPC Marfuri alimentare (%)	IPC Marfuri nealimentare (%)	\
2019-01-01	101.30	100.63	
2019-02-01	101.27	100.57	
2019-03-01	100.48	100.54	
2019-04-01	100.68	100.53	
2019-05-01	100.79	100.20	
2019-06-01	99.53	99.76	
2019-07-01	99.34	99.98	
2019-08-01	99.71	100.22	
2019-09-01	99.91	100.13	
2019-10-01	100.70	100.32	
2019-11-01	100.43	100.12	
2019-12-01	100.84	100.28	
2020-01-01	100.99	100.02	
2020-02-01	100.63	99.94	
2020-03-01	101.46	99.91	
2020-04-01	101.27	99.67	

2020-05-01	100.34	99.82
2020-06-01	99.62	100.28
2020-07-01	99.55	100.19
2020-08-01	99.59	100.08
2020-09-01	99.45	99.99
2020-10-01	100.11	100.31
2020-11-01	99.92	100.29
2020-12-01	100.29	100.51
2021-01-01	100.63	102.24
2021-02-01	100.46	100.47
2021-03-01	100.37	100.46
2021-04-01	100.45	100.47
2021-05-01	101.10	100.28
2021-06-01	100.25	100.29
2021-07-01	99.71	102.02
2021-08-01	99.95	100.34
2021-09-01	100.95	100.73
2021-10-01	101.06	102.78
2021-11-01	100.73	99.47
2021-12-01	100.84	100.74
2022-01-01	101.15	101.73
2022-02-01	101.96	99.70
2022-03-01	102.54	101.86
2022-04-01	102.56	105.45
2022-05-01	101.73	101.00
2022-06-01	100.62	100.92
2022-07-01	100.92	100.87
2022-08-01	101.82	99.81
2022-09-01	101.72	101.28
2022-10-01	102.29	100.80
2022-11-01	101.54	101.03
2022-12-01	101.26	99.68

month_id	IPC Servicii (%)
2019-01-01	100.57
2019-02-01	100.55
2019-03-01	100.40
2019-04-01	100.72
2019-05-01	100.55
2019-06-01	100.17
2019-07-01	100.10
2019-08-01	100.25
2019-09-01	100.27
2019-10-01	100.25
2019-11-01	100.15
2019-12-01	100.12
2020-01-01	100.43
2020-02-01	100.39
2020-03-01	100.35
2020-04-01	100.00
2020-05-01	100.11
2020-06-01	100.34
2020-07-01	100.31
2020-08-01	100.21
2020-09-01	100.20
2020-10-01	100.22
2020-11-01	100.07
2020-12-01	100.04
2021-01-01	100.25
2021-02-01	100.20
2021-03-01	100.24
2021-04-01	100.40
2021-05-01	100.27
2021-06-01	100.23
2021-07-01	100.39

```
2021-08-01            100.43
2021-09-01            100.96
2021-10-01            100.42
2021-11-01            100.20
2021-12-01            100.42
2022-01-01            101.37
2022-02-01            100.60
2022-03-01            100.67
2022-04-01            100.94
2022-05-01            100.61
2022-06-01            100.54
2022-07-01            100.88
2022-08-01            100.37
2022-09-01            100.72
2022-10-01            100.70
2022-11-01            101.31
2022-12-01            100.67
```

In [29]:
```
# Line graph for apa_plata for all stores
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index.get_level_values(0), filter_df_sales['sales'])
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales for ' + product + ' for all stores by Month')
plt.show()
```

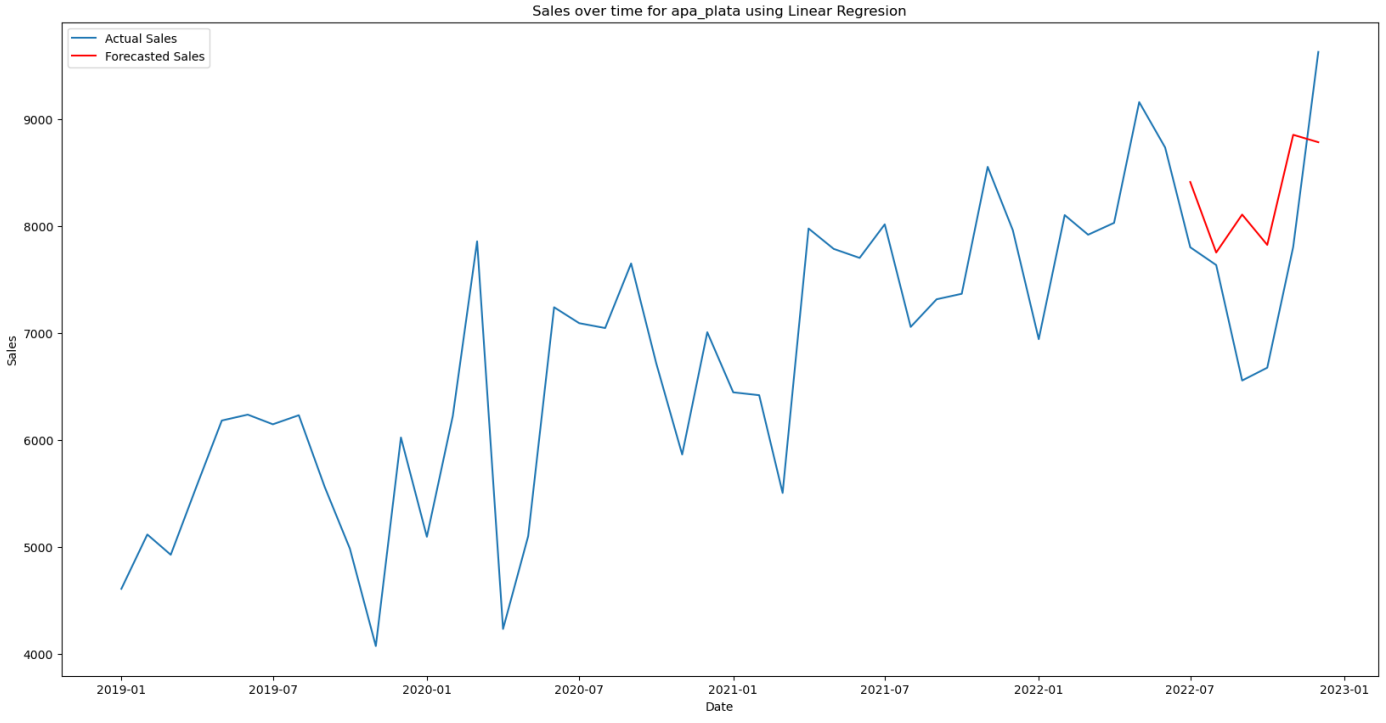


LINEAR REGRESSION - apa plata

In [30]:
```
# Get the last 6 months of data
last_6_months = filter_df_sales.tail(6)

# Split the data into train and test sets
X_train = filter_df_sales.drop('sales', axis=1)
y_train = filter_df_sales['sales']
X_test = last_6_months.drop('sales', axis=1)
y_test = last_6_months['sales']

# Create a linear regression model
model = LinearRegression()
```

```
# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the error
lr_mae = mean_absolute_error(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)

print("Mean Absolute Error: ", lr_mae)
print("Mean Squared Error: ", lr_mse)
print("Root Mean Squared Error:", lr_rmse)

all_sales = filter_df_sales['sales']

# Get the index of the last 6 months of data
last_6_months_index = last_6_months.index

# Plot the entire period of sales data
plt.figure(figsize=(20,10))
plt.plot(all_sales)

# Make predictions on the last 6 months of data
y_pred = model.predict(X_test)

# Plot the forecasted sales on the same chart
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Linear Regresion')
plt.legend(['Actual Sales', 'Forecasted Sales'])

plt.show()
```

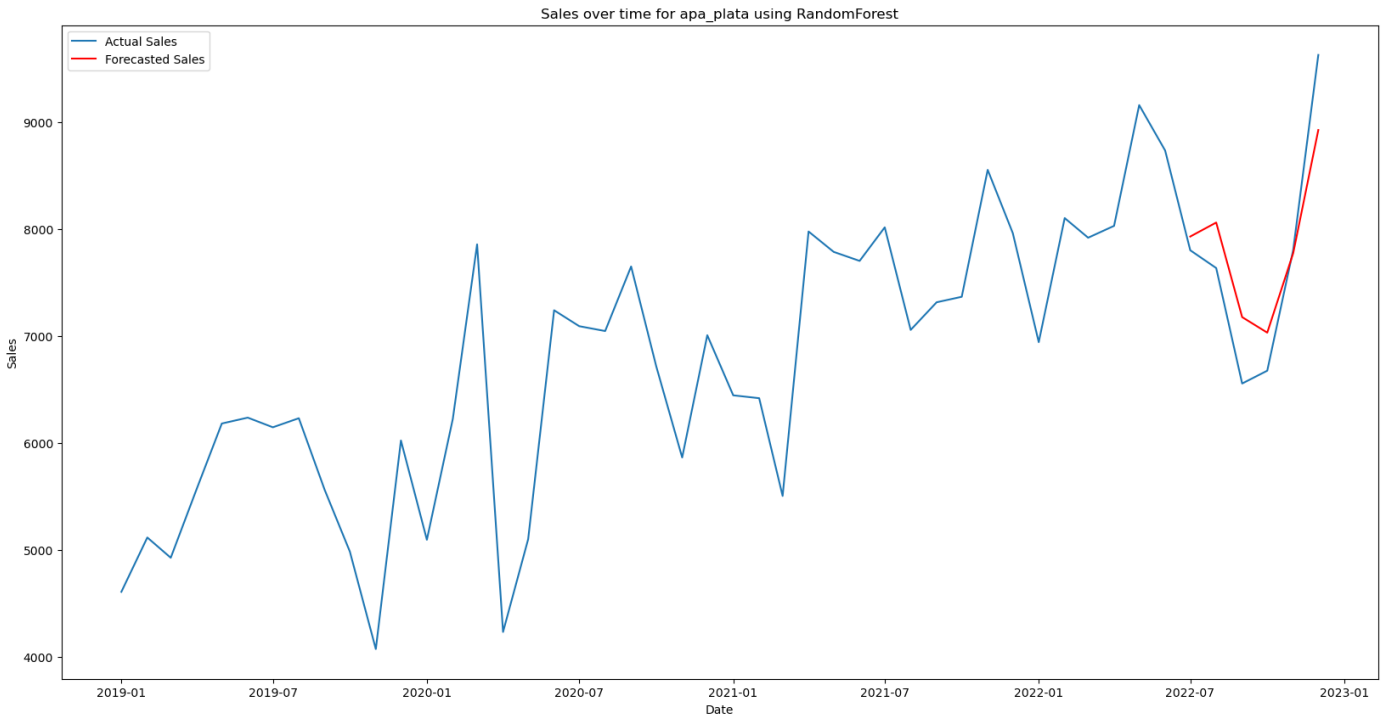
```
Mean Absolute Error:  887.0158288546555
Mean Squared Error:  988186.8483074202
Root Mean Squared Error: 994.0758765342916
```



Random Forest - apa plata

In [31]:
```
last_6_months = filter_df_sales.tail(6)

X_train = filter_df_sales.drop(columns=["sales"])
y_train = filter_df_sales["sales"]
X_test = last_6_months.drop(columns=["sales"])
y_test = last_6_months["sales"]

# Create a random forest model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Use the model to make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate the model's accuracy
accuracy = rf.score(X_test, y_test)
print("Accuracy: ", accuracy)

# Calculate error
rf_mae = mean_absolute_error(y_test, y_pred)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rf_mse)

print("Mean Absolute Error: ", rf_mae)
print("Mean Squared Error: ", rf_mse)
print("Root Mean Squared Error:", rf_rmse)

# Line graph for the forcast
plt.figure(figsize=(20,10))

plt.plot(all_sales)
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using RandomForest')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
Accuracy:  0.8027523949094745
Mean Absolute Error:  377.42626666667
Mean Squared Error:  200759.7701166457
Root Mean Squared Error: 448.0622391104228
```

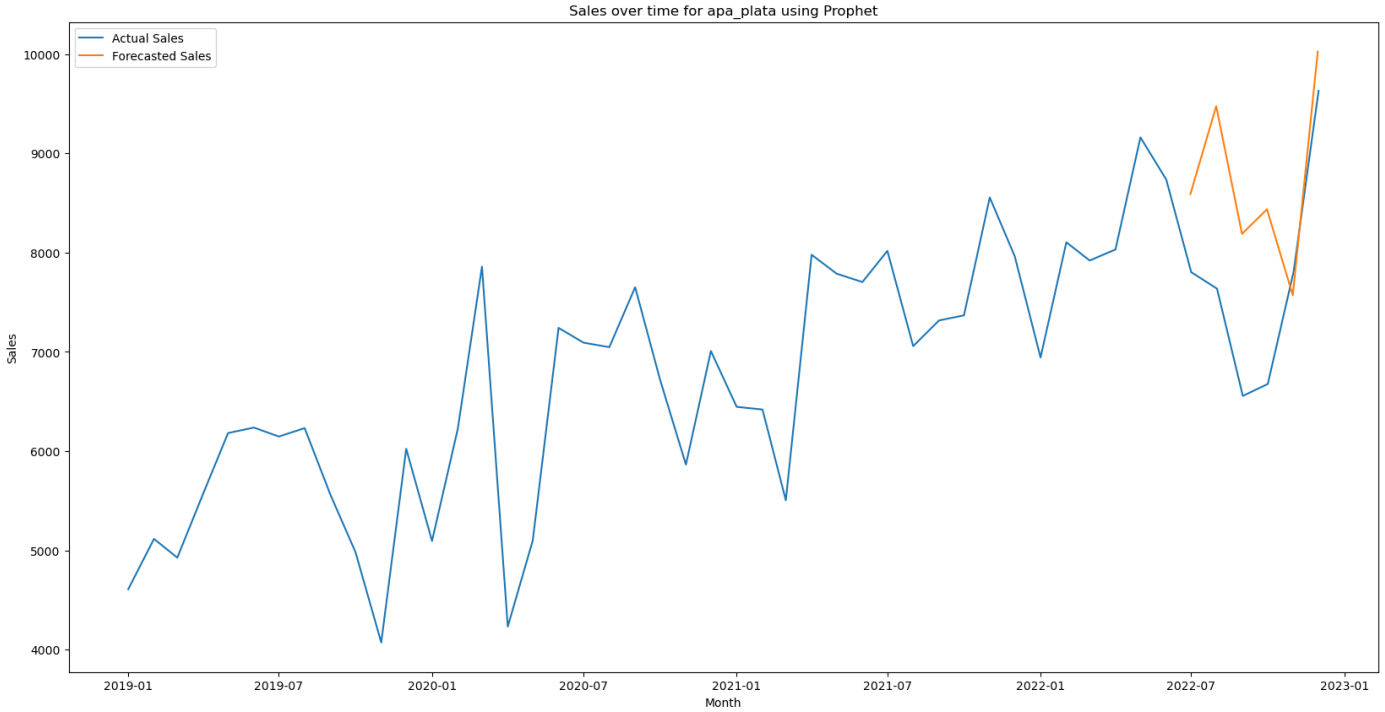


Prophet - apa plata

In [32]:
```
# create a copy of the dataframe with a column named 'ds' with the index values
filter_df_sales_prophet = filter_df_sales.reset_index().rename(columns={'month_id': 'ds'
filter_df_sales_prophet = filter_df_sales_prophet.rename(columns={'sales':'y'})

train_data = filter_df_sales_prophet[:-6]
test_data = filter_df_sales_prophet[-6:]

model = Prophet()

# Fit the model on the data
model.fit(train_data)

# Create a dataframe to hold the forecast
future_data = model.make_future_dataframe(periods=6, freq='m')

# Make predictions
forecast = model.predict(future_data)
forecast = forecast.tail(6)


# Calculate error
pr_mae = mean_absolute_error(test_data['y'], forecast[-6:]['yhat'])
pr_mse = mean_squared_error(test_data['y'], forecast[-6:]['yhat'])
pr_rmse = np.sqrt(pr_mse)
print("Mean Absolute Error:", pr_mae)
print("Mean Squared Error:", pr_mse)
print("Root Mean Squared Error:", pr_rmse)

# Plot the forecast
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index, filter_df_sales['sales'], label='Actual Sales')

# Plot the predicted values for the last six months
plt.plot(forecast.ds, forecast.yhat, label='Predicted Sales')

plt.xlabel('Month')
plt.ylabel('Sales')
```

```
plt.title('Sales over time for ' +product+ ' using Prophet')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
21:34:45 - cmdstanpy - INFO - Chain [1] start processing
21:34:45 - cmdstanpy - INFO - Chain [1] done processing
Mean Absolute Error: 1107.9106493623256
Mean Squared Error: 1661800.8818475215
Root Mean Squared Error: 1289.1085609239904
```



In [33]:

```
# Add the errors from all methods to the DataFrame
new_row_df =  pd.DataFrame({'product': [product], 'LR-MAE': [lr_mae], 'LR-MSE': [lr_mse]
errors_df = pd.concat([errors_df, new_row_df], ignore_index=True)
print(errors_df)
```

```
     product       LR-MAE          LR-MSE      LR-RMSE       RF-MAE  \
0  apa_plata   887.015829   988186.848307   994.075877   377.426267

          RF-MSE      RF-RMSE        PR-MAE         PR-MSE       PR-RMSE
0  200759.770117   448.062239   1107.910649   1.661801e+06   1289.108561
```

Apa minerala dataframe

In [34]:

```
# Filter the initial DataFrame
product = 'apa_minerala'
filter_df = df.loc[df['denumire_produs'] == product]
filter_df = filter_df.drop(columns=['colli', 'store_county'])
#print(apa_plata_df)

# Group by month_id and aggregate the other columns
filter_df_sales = filter_df.groupby("month_id").agg({'sales': 'sum','Inflatie anuala': '
filter_df_sales = filter_df_sales.sort_values(by='month_id', ascending=True)

# Print the apa_plata_sales dataframe
print(filter_df_sales)
```

```
              sales  Inflatie anuala  unnemployment  \
month_id
2019-02-01     5.82              3.8           3.32
2019-03-01  123.75               3.8           3.31
2019-04-01  123.75               3.8           3.19
```

2019-06-01	26.19	3.8	2.95
2019-08-01	26.01	3.8	3.01
2019-09-01	78.03	3.8	3.03
2020-01-01	78.03	2.6	2.97
2020-02-01	138.78	2.6	2.98
2020-03-01	17.46	2.6	2.95
2020-05-01	26.19	2.6	2.90
2020-06-01	26.01	2.6	2.87
2020-07-01	320.85	2.6	3.00
2020-08-01	345.03	2.6	3.02
2020-09-01	26.01	2.6	3.30
2020-10-01	52.02	2.6	3.26
2020-12-01	26.01	2.6	3.32
2021-04-01	0.00	5.1	3.33
2021-05-01	60.75	5.1	3.16
2021-06-01	112.95	5.1	3.06
2021-07-01	138.96	5.1	3.00
2021-08-01	69.66	5.1	2.96
2021-10-01	8.73	5.1	2.85
2022-07-01	10.38	13.8	2.55
2022-08-01	42.78	13.8	2.56

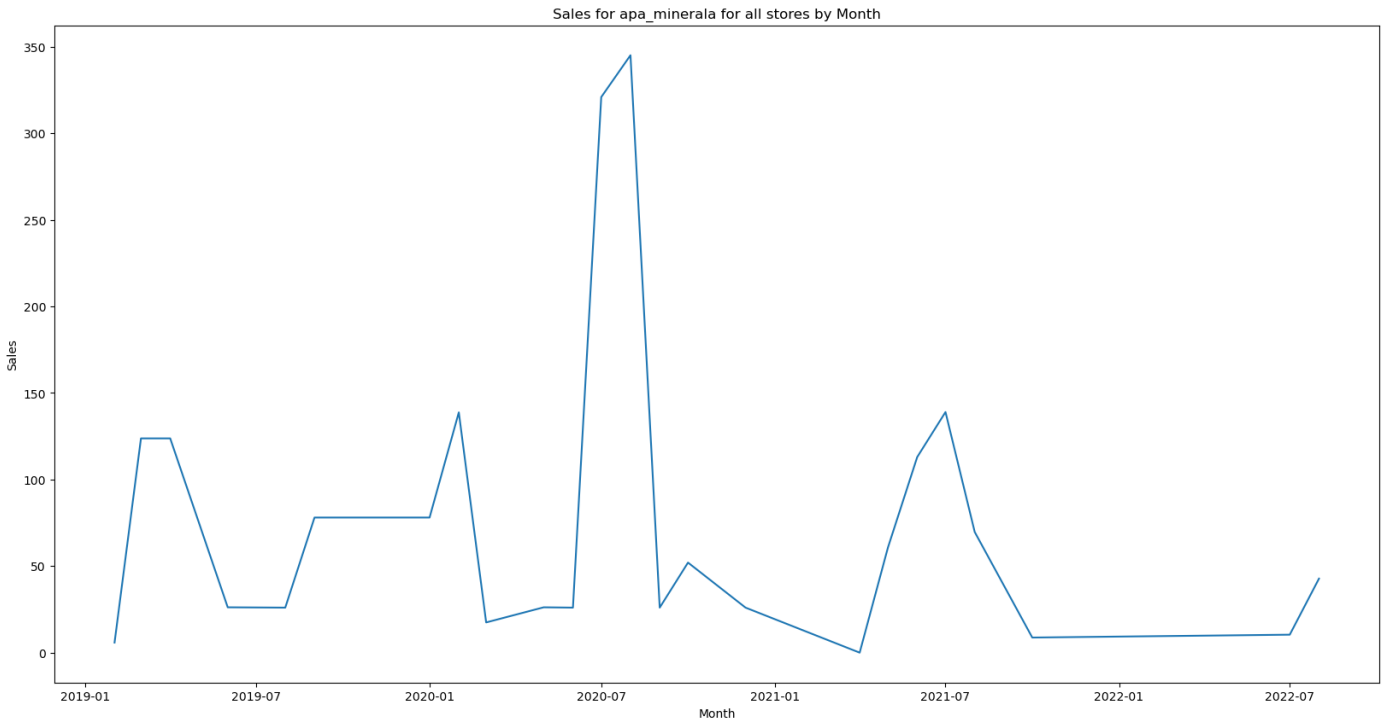
month_id	Inflatie(de la o luna la alta)	net_average_salary	\
2019-02-01	0.79	2933.0	
2019-03-01	0.49	3075.0	
2019-04-01	0.61	3115.0	
2019-06-01	-0.23	3142.0	
2019-08-01	0.06	3044.0	
2019-09-01	0.09	3082.0	
2020-01-01	0.41	3189.0	
2020-02-01	0.25	3202.0	
2020-03-01	0.50	3294.0	
2020-05-01	0.05	3179.0	
2020-06-01	0.08	3298.0	
2020-07-01	0.00	3372.0	
2020-08-01	-0.05	3275.0	
2020-09-01	-0.14	3321.0	
2020-10-01	0.22	3343.0	
2020-12-01	0.34	3620.0	
2021-04-01	0.45	3561.0	
2021-05-01	0.53	3492.0	
2021-06-01	0.27	3541.0	
2021-07-01	0.97	3545.0	
2021-08-01	0.24	3487.0	
2021-10-01	1.78	3544.0	
2022-07-01	0.89	3975.0	
2022-08-01	0.56	3933.0	

month_id	IPC Marfuri alimentare (%)	IPC Marfuri nealimentare (%)	\
2019-02-01	101.27	100.57	
2019-03-01	100.48	100.54	
2019-04-01	100.68	100.53	
2019-06-01	99.53	99.76	
2019-08-01	99.71	100.22	
2019-09-01	99.91	100.13	
2020-01-01	100.99	100.02	
2020-02-01	100.63	99.94	
2020-03-01	101.46	99.91	
2020-05-01	100.34	99.82	
2020-06-01	99.62	100.28	
2020-07-01	99.55	100.19	
2020-08-01	99.59	100.08	
2020-09-01	99.45	99.99	
2020-10-01	100.11	100.31	

```
2020-12-01                    100.29                    100.51
2021-04-01                    100.45                    100.47
2021-05-01                    101.10                    100.28
2021-06-01                    100.25                    100.29
2021-07-01                     99.71                    102.02
2021-08-01                     99.95                    100.34
2021-10-01                    101.06                    102.78
2022-07-01                    100.92                    100.87
2022-08-01                    101.82                     99.81
```

```
            IPC Servicii (%)
month_id
2019-02-01            100.55
2019-03-01            100.40
2019-04-01            100.72
2019-06-01            100.17
2019-08-01            100.25
2019-09-01            100.27
2020-01-01            100.43
2020-02-01            100.39
2020-03-01            100.35
2020-05-01            100.11
2020-06-01            100.34
2020-07-01            100.31
2020-08-01            100.21
2020-09-01            100.20
2020-10-01            100.22
2020-12-01            100.04
2021-04-01            100.40
2021-05-01            100.27
2021-06-01            100.23
2021-07-01            100.39
2021-08-01            100.43
2021-10-01            100.42
2022-07-01            100.88
2022-08-01            100.37
```

In [35]:
```
# Line graph for apa_plata for all stores
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index.get_level_values(0), filter_df_sales['sales'])
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales for ' + product + ' for all stores by Month')
plt.show()
```

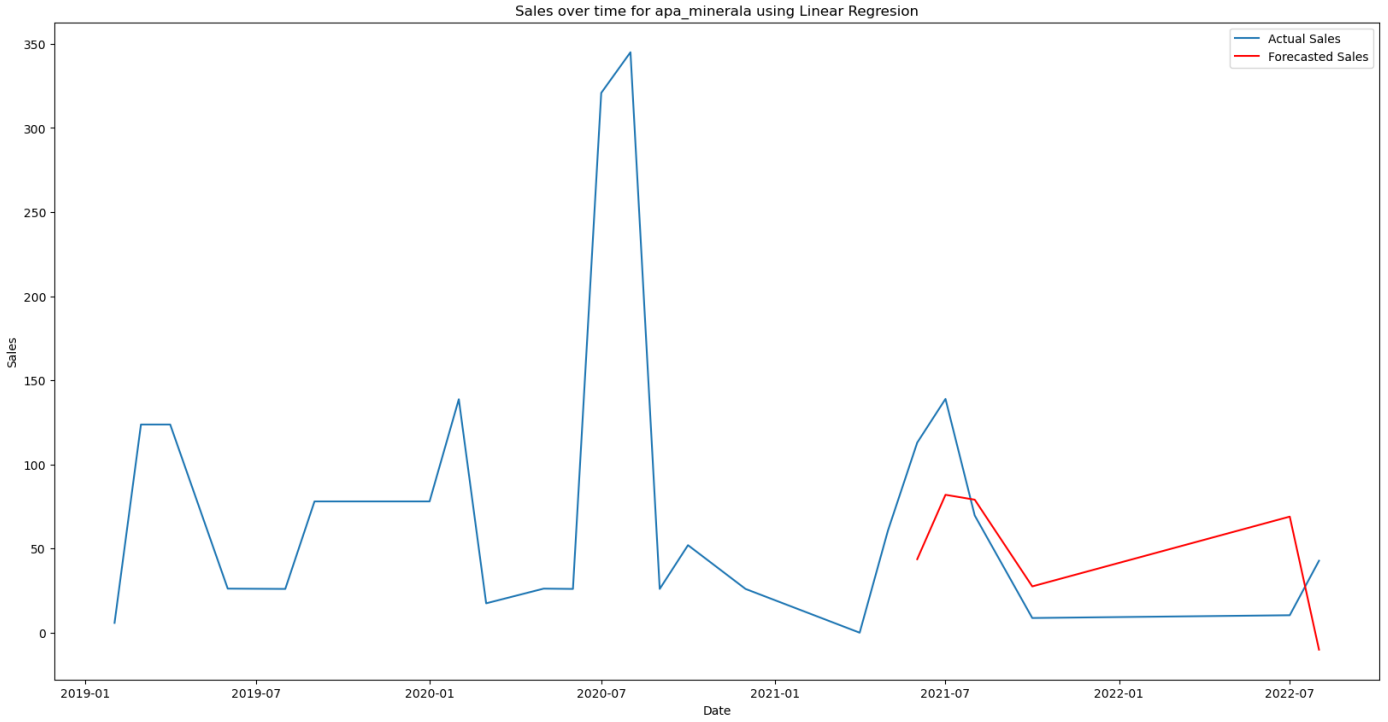


LR - apa_minerala

In [36]:
```
# Get the last 6 months of data
last_6_months = filter_df_sales.tail(6)

# Split the data into train and test sets
X_train = filter_df_sales.drop('sales', axis=1)
y_train = filter_df_sales['sales']
X_test = last_6_months.drop('sales', axis=1)
y_test = last_6_months['sales']

# Create a linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the error
lr_mae = mean_absolute_error(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)

print("Mean Absolute Error: ", lr_mae)
print("Mean Squared Error: ", lr_mse)
print("Root Mean Squared Error:", lr_rmse)

all_sales = filter_df_sales['sales']

# Get the index of the last 6 months of data
last_6_months_index = last_6_months.index

# Plot the entire period of sales data
plt.figure(figsize=(20,10))
plt.plot(all_sales)

# Make predictions on the last 6 months of data
```

```
y_pred = model.predict(X_test)

# Plot the forecasted sales on the same chart
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Linear Regresion')
plt.legend(['Actual Sales', 'Forecasted Sales'])

plt.show()
```

```
Mean Absolute Error:  44.3245347091121
Mean Squared Error:  2453.366899333518
Root Mean Squared Error: 49.53147382557396
```



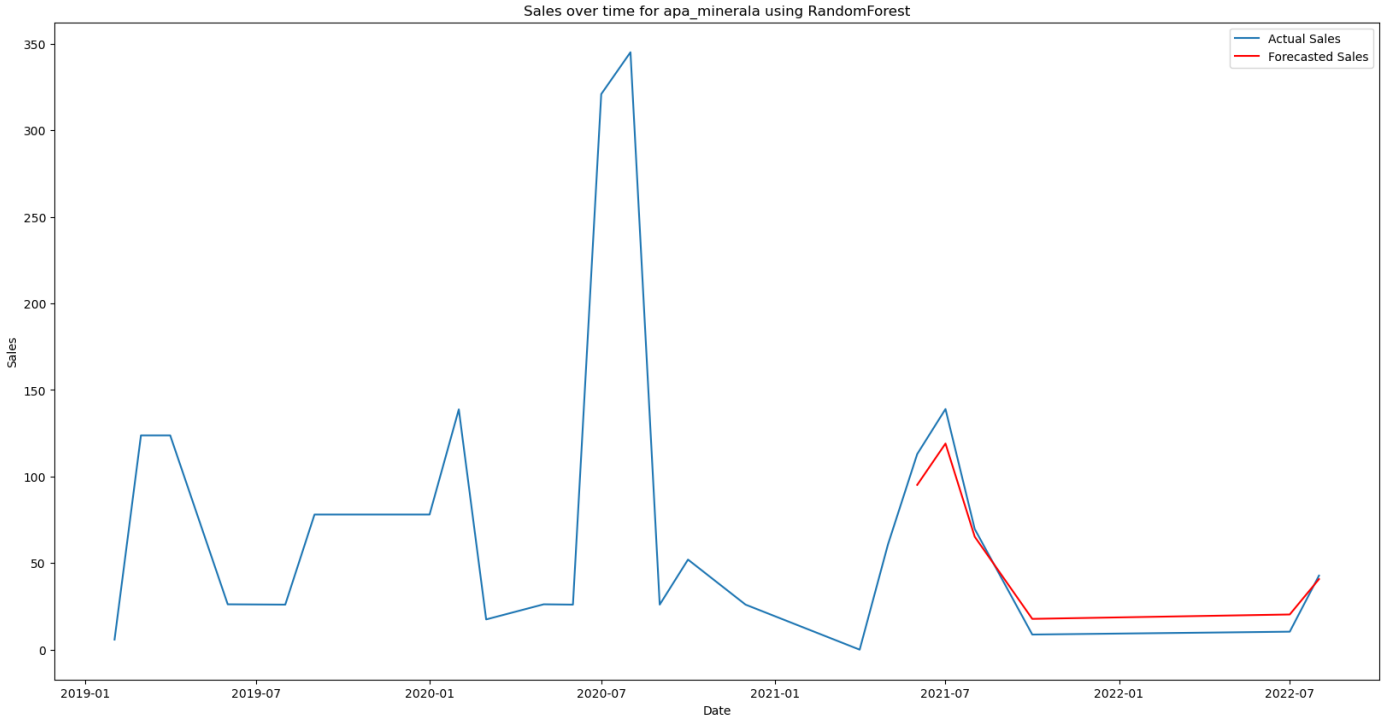
RF - apa_minerala

In [37]:

```
last_6_months = filter_df_sales.tail(6)

X_train = filter_df_sales.drop(columns=["sales"])
y_train = filter_df_sales["sales"]
X_test = last_6_months.drop(columns=["sales"])
y_test = last_6_months["sales"]

# Create a random forest model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Use the model to make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate the model's accuracy
accuracy = rf.score(X_test, y_test)
print("Accuracy: ", accuracy)

# Calculate error
rf_mae = mean_absolute_error(y_test, y_pred)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rf_mse)
```

```
print("Mean Absolute Error: ", rf_mae)
print("Mean Squared Error: ", rf_mse)
print("Root Mean Squared Error:", rf_rmse)

# Line graph for the forcast
plt.figure(figsize=(20,10))

plt.plot(all_sales)
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using RandomForest')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
Accuracy:  0.936337099048517
Mean Absolute Error:  10.53310000000004
Mean Squared Error:  153.08017011000155
Root Mean Squared Error: 12.372557137067565
```



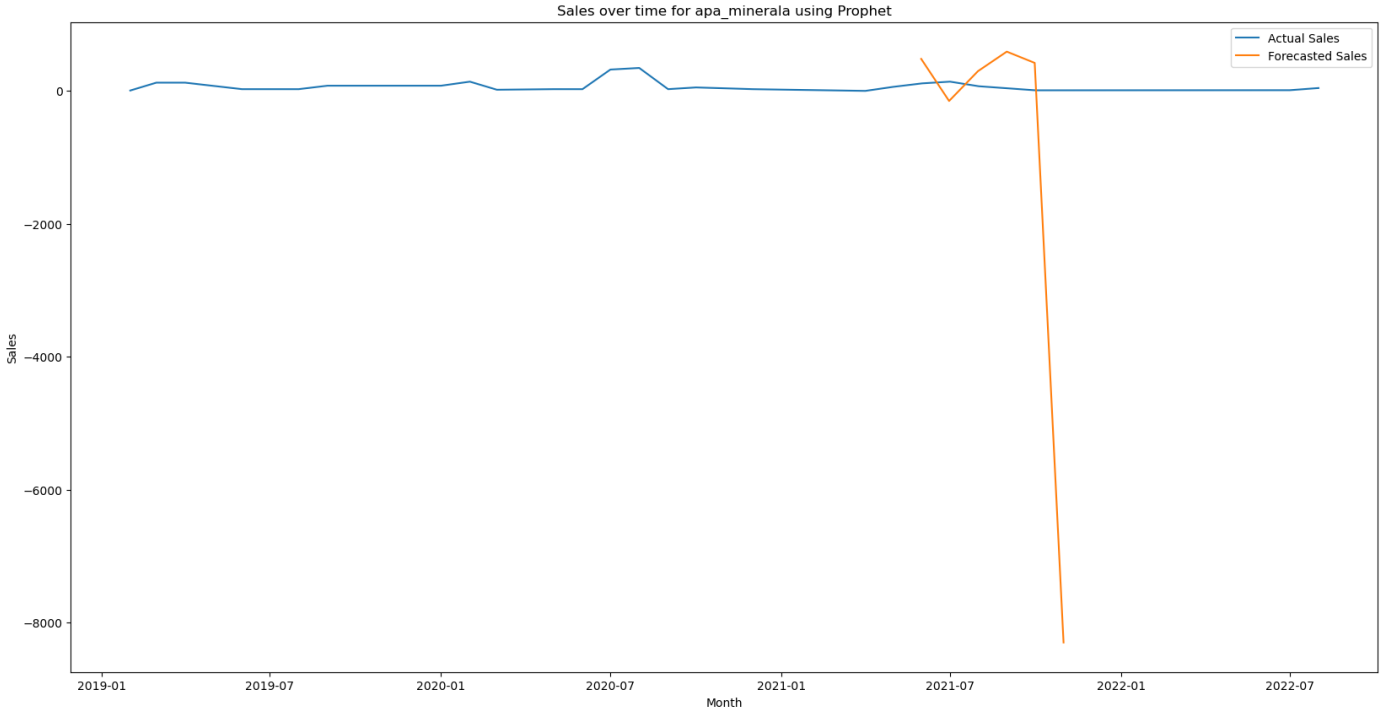
PR - apa_minerala

In [38]:

```
# create a copy of the dataframe with a column named 'ds' with the index values
filter_df_sales_prophet = filter_df_sales.reset_index().rename(columns={'month_id': 'ds'
filter_df_sales_prophet = filter_df_sales_prophet.rename(columns={'sales':'y'})

train_data = filter_df_sales_prophet[:-6]
test_data = filter_df_sales_prophet[-6:]

model = Prophet()

# Fit the model on the data
model.fit(train_data)

# Create a dataframe to hold the forecast
future_data = model.make_future_dataframe(periods=6, freq='m')

# Make predictions
forecast = model.predict(future_data)
forecast = forecast.tail(6)
```

```
# Calculate error
pr_mae = mean_absolute_error(test_data['y'], forecast[-6:]['yhat'])
pr_mse = mean_squared_error(test_data['y'], forecast[-6:]['yhat'])
pr_rmse = np.sqrt(pr_mse)
print("Mean Absolute Error:", pr_mae)
print("Mean Squared Error:", pr_mse)
print("Root Mean Squared Error:", pr_rmse)

# Plot the forecast
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index, filter_df_sales['sales'], label='Actual Sales')

# Plot the predicted values for the last six months
plt.plot(forecast.ds, forecast.yhat, label='Predicted Sales')

plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Prophet')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
21:34:47 - cmdstanpy - INFO - Chain [1] start processing
21:35:07 - cmdstanpy - INFO - Chain [1] done processing
Mean Absolute Error: 1703.2588808091807
Mean Squared Error: 11725043.17138497
Root Mean Squared Error: 3424.1850375505364
```



In [39]:
```
# Add the errors from all methods to the DataFrame
new_row_df =  pd.DataFrame({'product': [product], 'LR-MAE': [lr_mae], 'LR-MSE': [lr_mse]
errors_df = pd.concat([errors_df, new_row_df], ignore_index=True)
print(errors_df)
```

```
        product      LR-MAE         LR-MSE     LR-RMSE      RF-MAE  \
0     apa_plata  887.015829  988186.848307  994.075877  377.426267
1  apa_minerala   44.324535    2453.366899   49.531474   10.533100

          RF-MSE     RF-RMSE       PR-MAE        PR-MSE      PR-RMSE
0  200759.770117  448.062239  1107.910649  1.661801e+06  1289.108561
1     153.080170   12.372557  1703.258881  1.172504e+07  3424.185038
```

Bere DataFrame

In [40]:
```
# Filter the initial DataFrame
product = 'bere'
filter_df = df.loc[df['denumire_produs'] == product]
filter_df = filter_df.drop(columns=['colli', 'store_county'])
#print(apa_plata_df)

# Group by month_id and aggregate the other columns
filter_df_sales = filter_df.groupby("month_id").agg({'sales': 'sum','Inflatie anuala': '
filter_df_sales = filter_df_sales.sort_values(by='month_id', ascending=True)

# Print the apa_plata_sales dataframe
print(filter_df_sales)
```

month_id	sales	Inflatie anuala	unnemployment
2019-01-01	395.56	3.8	1.42
2019-02-01	295.26	3.8	3.32
2019-03-01	520.70	3.8	3.31
2019-04-01	385.67	3.8	3.19
2019-05-01	501.63	3.8	3.00
2019-06-01	909.31	3.8	2.95
2019-07-01	791.65	3.8	2.95
2019-08-01	700.06	3.8	3.01
2019-09-01	585.47	3.8	3.03
2019-10-01	404.72	3.8	3.00
2019-11-01	297.84	3.8	2.98
2019-12-01	493.94	3.8	2.98
2020-01-01	458.81	2.6	2.97
2020-02-01	291.28	2.6	2.98
2020-03-01	931.91	2.6	2.95
2020-04-01	570.13	2.6	2.88
2020-05-01	781.09	2.6	2.90
2020-06-01	528.85	2.6	2.87
2020-07-01	660.74	2.6	3.00
2020-08-01	694.19	2.6	3.02
2020-09-01	397.57	2.6	3.30
2020-10-01	334.42	2.6	3.26
2020-11-01	264.36	2.6	3.27
2020-12-01	485.00	2.6	3.32
2021-01-01	551.12	5.1	3.38
2021-02-01	348.82	5.1	3.34
2021-03-01	357.66	5.1	3.35
2021-04-01	200.96	5.1	3.33
2021-05-01	226.70	5.1	3.16
2021-06-01	613.72	5.1	3.06
2021-07-01	567.49	5.1	3.00
2021-08-01	638.82	5.1	2.96
2021-09-01	702.36	5.1	2.93
2021-10-01	619.81	5.1	2.85
2021-11-01	462.38	5.1	2.76
2021-12-01	850.56	5.1	2.72
2022-01-01	294.00	13.8	2.69
2022-02-01	881.15	13.8	2.68
2022-03-01	830.41	13.8	2.67
2022-04-01	1010.63	13.8	2.64
2022-05-01	1040.63	13.8	2.57
2022-06-01	1339.46	13.8	2.55
2022-07-01	1440.80	13.8	2.55
2022-08-01	1402.94	13.8	2.56
2022-09-01	1250.40	13.8	2.56
2022-10-01	765.58	13.8	2.88
2022-11-01	525.64	13.8	2.96

2022-12-01	487.94	13.8	3.04

month_id	Inflatie(de la o luna la alta)	net_average_salary	\
2019-01-01	0.83	2936.0	
2019-02-01	0.79	2933.0	
2019-03-01	0.49	3075.0	
2019-04-01	0.61	3115.0	
2019-05-01	0.46	3101.0	
2019-06-01	-0.23	3142.0	
2019-07-01	-0.20	3119.0	
2019-08-01	0.06	3044.0	
2019-09-01	0.09	3082.0	
2019-10-01	0.43	3116.0	
2019-11-01	0.23	3179.0	
2019-12-01	0.42	3340.0	
2020-01-01	0.41	3189.0	
2020-02-01	0.25	3202.0	
2020-03-01	0.50	3294.0	
2020-04-01	0.26	3182.0	
2020-05-01	0.05	3179.0	
2020-06-01	0.08	3298.0	
2020-07-01	0.00	3372.0	
2020-08-01	-0.05	3275.0	
2020-09-01	-0.14	3321.0	
2020-10-01	0.22	3343.0	
2020-11-01	0.13	3411.0	
2020-12-01	0.34	3620.0	
2021-01-01	1.33	3395.0	
2021-02-01	0.41	3365.0	
2021-03-01	0.38	3547.0	
2021-04-01	0.45	3561.0	
2021-05-01	0.53	3492.0	
2021-06-01	0.27	3541.0	
2021-07-01	0.97	3545.0	
2021-08-01	0.24	3487.0	
2021-09-01	0.84	3517.0	
2021-10-01	1.78	3544.0	
2021-11-01	0.00	3645.0	
2021-12-01	0.71	3879.0	
2022-01-01	1.48	3698.0	
2022-02-01	0.58	3721.0	
2022-03-01	1.88	3937.0	
2022-04-01	3.74	3967.0	
2022-05-01	1.18	3928.0	
2022-06-01	0.76	3977.0	
2022-07-01	0.89	3975.0	
2022-08-01	0.56	3933.0	
2022-09-01	1.33	4003.0	
2022-10-01	1.28	4008.0	
2022-11-01	1.25	4141.0	
2022-12-01	0.37	4141.0	

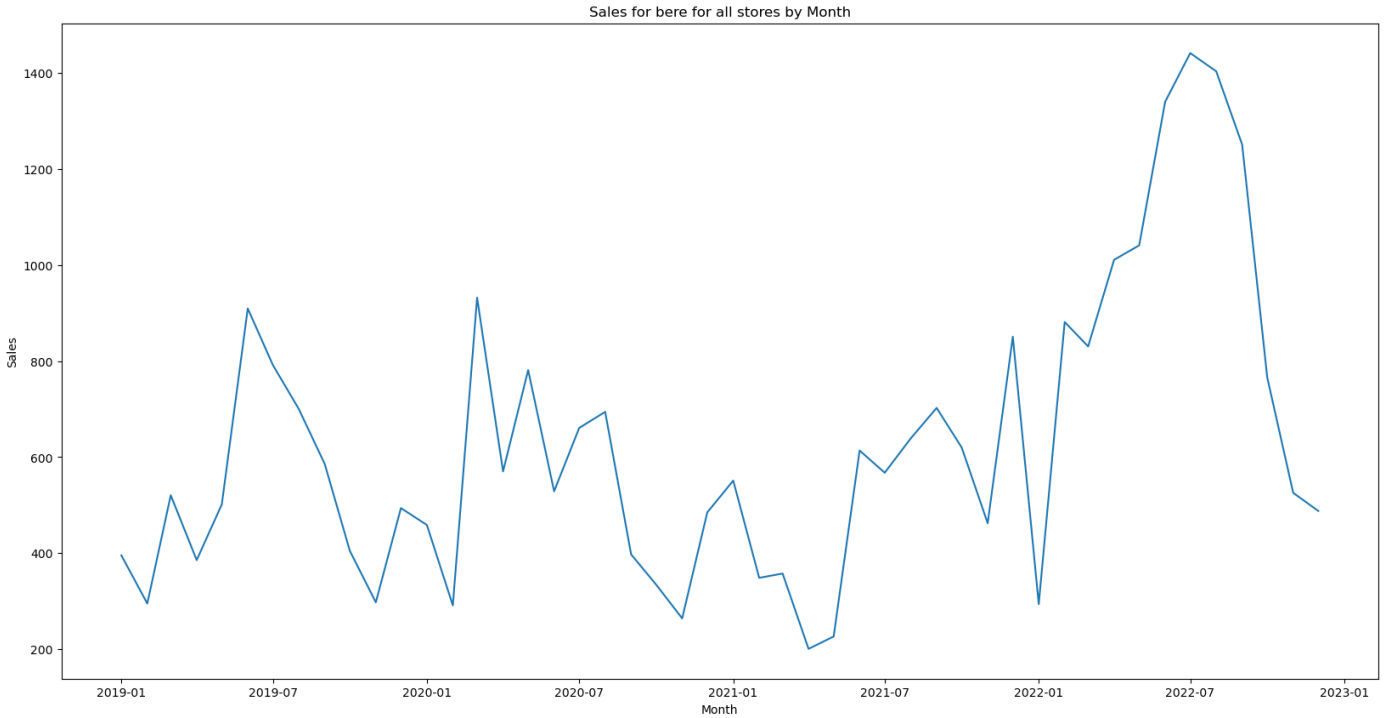
month_id	IPC Marfuri alimentare (%)	IPC Marfuri nealimentare (%)	\
2019-01-01	101.30	100.63	
2019-02-01	101.27	100.57	
2019-03-01	100.48	100.54	
2019-04-01	100.68	100.53	
2019-05-01	100.79	100.20	
2019-06-01	99.53	99.76	
2019-07-01	99.34	99.98	
2019-08-01	99.71	100.22	
2019-09-01	99.91	100.13	
2019-10-01	100.70	100.32	
2019-11-01	100.43	100.12	

2019-12-01	100.84	100.28
2020-01-01	100.99	100.02
2020-02-01	100.63	99.94
2020-03-01	101.46	99.91
2020-04-01	101.27	99.67
2020-05-01	100.34	99.82
2020-06-01	99.62	100.28
2020-07-01	99.55	100.19
2020-08-01	99.59	100.08
2020-09-01	99.45	99.99
2020-10-01	100.11	100.31
2020-11-01	99.92	100.29
2020-12-01	100.29	100.51
2021-01-01	100.63	102.24
2021-02-01	100.46	100.47
2021-03-01	100.37	100.46
2021-04-01	100.45	100.47
2021-05-01	101.10	100.28
2021-06-01	100.25	100.29
2021-07-01	99.71	102.02
2021-08-01	99.95	100.34
2021-09-01	100.95	100.73
2021-10-01	101.06	102.78
2021-11-01	100.73	99.47
2021-12-01	100.84	100.74
2022-01-01	101.15	101.73
2022-02-01	101.96	99.70
2022-03-01	102.54	101.86
2022-04-01	102.56	105.45
2022-05-01	101.73	101.00
2022-06-01	100.62	100.92
2022-07-01	100.92	100.87
2022-08-01	101.82	99.81
2022-09-01	101.72	101.28
2022-10-01	102.29	100.80
2022-11-01	101.54	101.03
2022-12-01	101.26	99.68

month_id	IPC Servicii (%)
2019-01-01	100.57
2019-02-01	100.55
2019-03-01	100.40
2019-04-01	100.72
2019-05-01	100.55
2019-06-01	100.17
2019-07-01	100.10
2019-08-01	100.25
2019-09-01	100.27
2019-10-01	100.25
2019-11-01	100.15
2019-12-01	100.12
2020-01-01	100.43
2020-02-01	100.39
2020-03-01	100.35
2020-04-01	100.00
2020-05-01	100.11
2020-06-01	100.34
2020-07-01	100.31
2020-08-01	100.21
2020-09-01	100.20
2020-10-01	100.22
2020-11-01	100.07
2020-12-01	100.04
2021-01-01	100.25
2021-02-01	100.20

```
2021-03-01            100.24
2021-04-01            100.40
2021-05-01            100.27
2021-06-01            100.23
2021-07-01            100.39
2021-08-01            100.43
2021-09-01            100.96
2021-10-01            100.42
2021-11-01            100.20
2021-12-01            100.42
2022-01-01            101.37
2022-02-01            100.60
2022-03-01            100.67
2022-04-01            100.94
2022-05-01            100.61
2022-06-01            100.54
2022-07-01            100.88
2022-08-01            100.37
2022-09-01            100.72
2022-10-01            100.70
2022-11-01            101.31
2022-12-01            100.67
```

In [41]:
```
# Line graph for apa_plata for all stores
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index.get_level_values(0), filter_df_sales['sales'])
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales for ' + product + ' for all stores by Month')
plt.show()
```

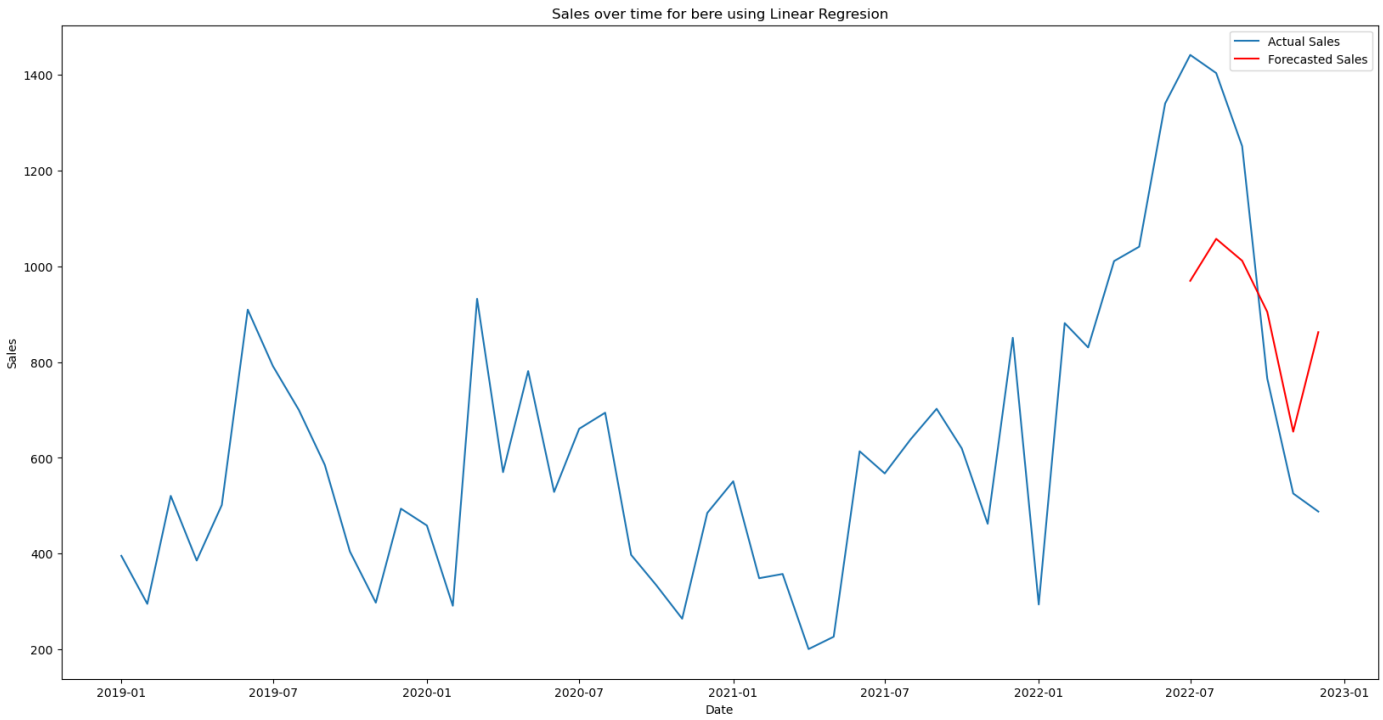


LR - bere

In [42]:
```
# Get the last 6 months of data
last_6_months = filter_df_sales.tail(6)

# Split the data into train and test sets
X_train = filter_df_sales.drop('sales', axis=1)
y_train = filter_df_sales['sales']
X_test = last_6_months.drop('sales', axis=1)
```

```
y_test = last_6_months['sales']

# Create a linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the error
lr_mae = mean_absolute_error(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)

print("Mean Absolute Error: ", lr_mae)
print("Mean Squared Error: ", lr_mse)
print("Root Mean Squared Error:", lr_rmse)

all_sales = filter_df_sales['sales']

# Get the index of the last 6 months of data
last_6_months_index = last_6_months.index

# Plot the entire period of sales data
plt.figure(figsize=(20,10))
plt.plot(all_sales)

# Make predictions on the last 6 months of data
y_pred = model.predict(X_test)

# Plot the forecasted sales on the same chart
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Linear Regresion')
plt.legend(['Actual Sales', 'Forecasted Sales'])

plt.show()
```

```
Mean Absolute Error:  283.10979168648987
Mean Squared Error:  95844.00265754924
Root Mean Squared Error: 309.58682571703406
```



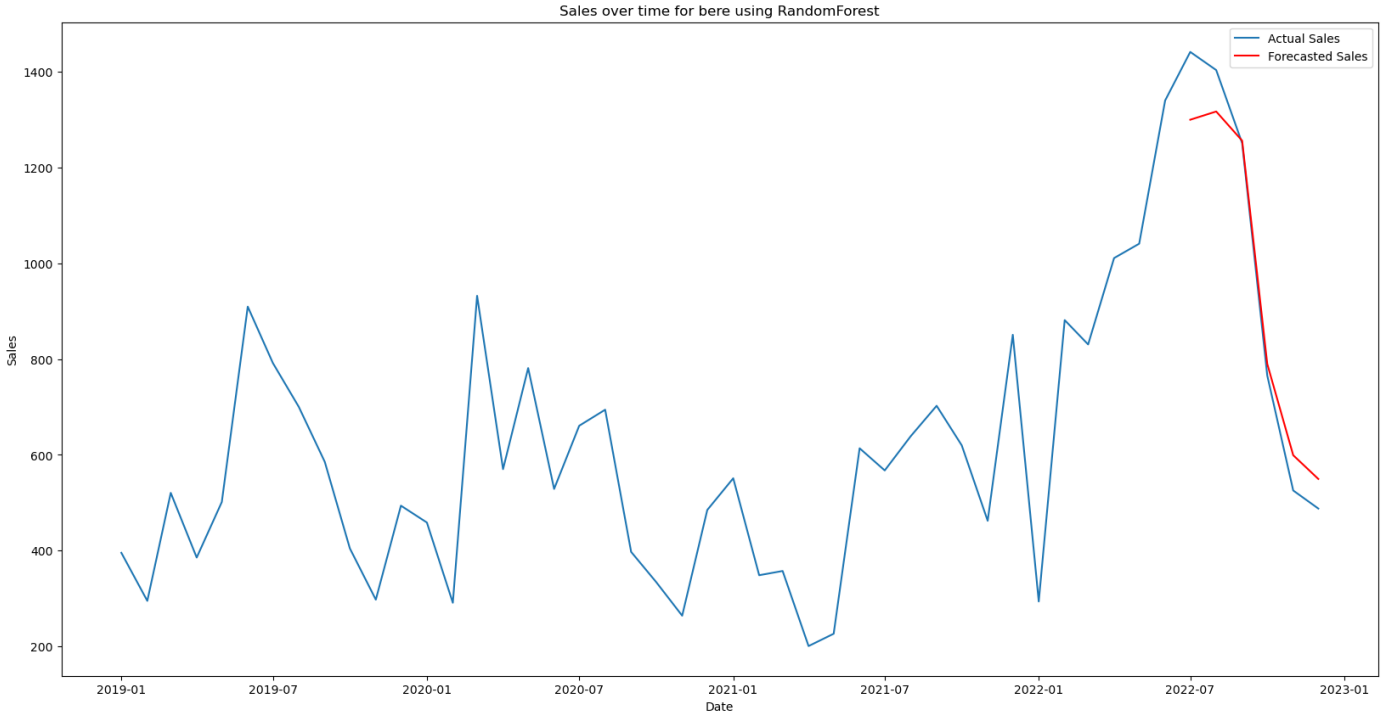
RF - bere

In [43]:

```
last_6_months = filter_df_sales.tail(6)

X_train = filter_df_sales.drop(columns=["sales"])
y_train = filter_df_sales["sales"]
X_test = last_6_months.drop(columns=["sales"])
y_test = last_6_months["sales"]

# Create a random forest model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Use the model to make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate the model's accuracy
accuracy = rf.score(X_test, y_test)
print("Accuracy: ", accuracy)

# Calculate error
rf_mae = mean_absolute_error(y_test, y_pred)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rf_mse)

print("Mean Absolute Error: ", rf_mae)
print("Mean Squared Error: ", rf_mse)
print("Root Mean Squared Error:", rf_rmse)

# Line graph for the forcast
plt.figure(figsize=(20,10))

plt.plot(all_sales)
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using RandomForest')
```

```
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
Accuracy:  0.9610675974137636
Mean Absolute Error:  65.44414999999942
Mean Squared Error:  6221.823279728219
Root Mean Squared Error: 78.87853497453042
```



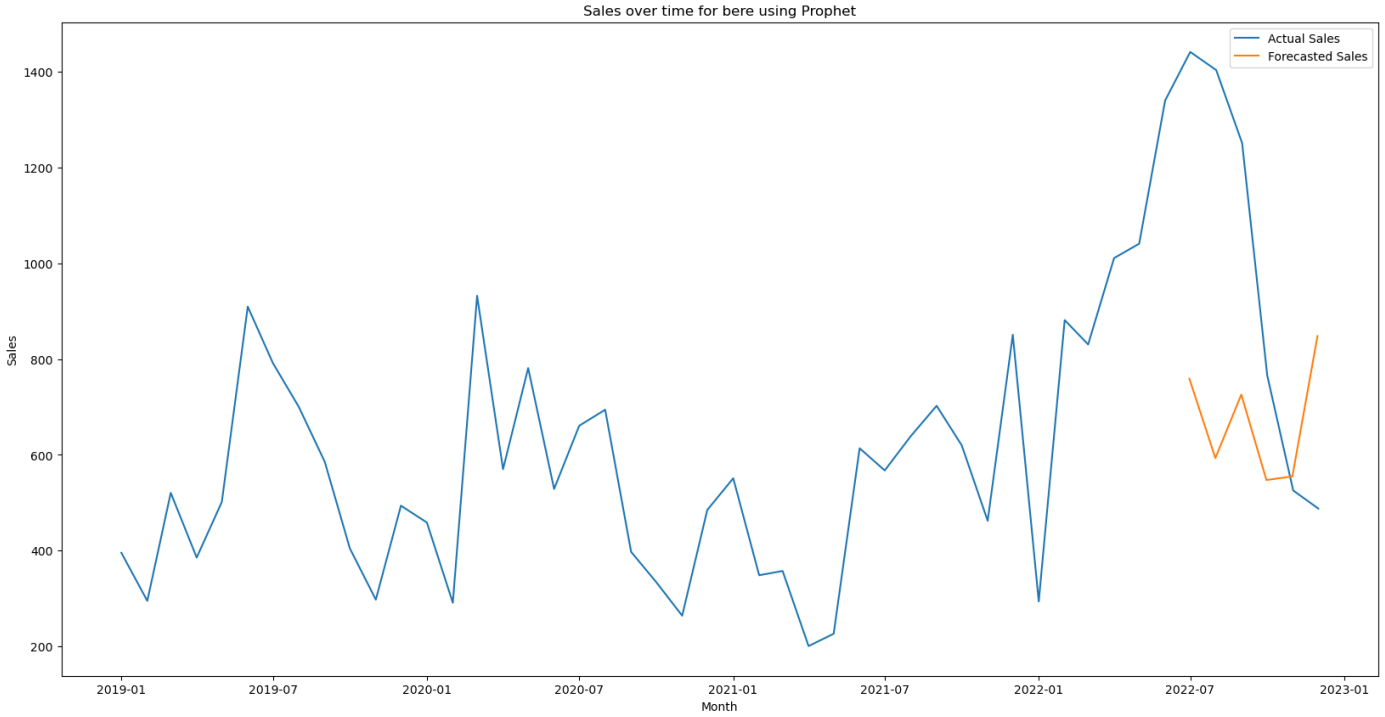
PR - bere

In [44]:

```
# create a copy of the dataframe with a column named 'ds' with the index values
filter_df_sales_prophet = filter_df_sales.reset_index().rename(columns={'month_id': 'ds'
filter_df_sales_prophet = filter_df_sales_prophet.rename(columns={'sales':'y'})

train_data = filter_df_sales_prophet[:-6]
test_data = filter_df_sales_prophet[-6:]

model = Prophet()

# Fit the model on the data
model.fit(train_data)

# Create a dataframe to hold the forecast
future_data = model.make_future_dataframe(periods=6, freq='m')

# Make predictions
forecast = model.predict(future_data)
forecast = forecast.tail(6)


# Calculate error
pr_mae = mean_absolute_error(test_data['y'], forecast[-6:]['yhat'])
pr_mse = mean_squared_error(test_data['y'], forecast[-6:]['yhat'])
pr_rmse = np.sqrt(pr_mse)
print("Mean Absolute Error:", pr_mae)
print("Mean Squared Error:", pr_mse)
print("Root Mean Squared Error:", pr_rmse)

# Plot the forecast
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index, filter_df_sales['sales'], label='Actual Sales')
```

```
# Plot the predicted values for the last six months
plt.plot(forecast.ds, forecast.yhat, label='Predicted Sales')

plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Prophet')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
21:35:08 - cmdstanpy - INFO - Chain [1] start processing
21:35:09 - cmdstanpy - INFO - Chain [1] done processing
Mean Absolute Error: 437.2978850529184
Mean Squared Error: 262332.512733538
Root Mean Squared Error: 512.184061381783
```



In [45]:

```
# Add the errors from all methods to the DataFrame
new_row_df =  pd.DataFrame({'product': [product], 'LR-MAE': [lr_mae], 'LR-MSE': [lr_mse]
errors_df = pd.concat([errors_df, new_row_df], ignore_index=True)
print(errors_df)
```

```
        product      LR-MAE         LR-MSE     LR-RMSE      RF-MAE  \
0     apa_plata  887.015829  988186.848307  994.075877  377.426267
1  apa_minerala   44.324535    2453.366899   49.531474   10.533100
2          bere  283.109792   95844.002658  309.586826   65.444150

          RF-MSE     RF-RMSE       PR-MAE        PR-MSE      PR-RMSE
0  200759.770117  448.062239  1107.910649  1.661801e+06  1289.108561
1     153.080170   12.372557  1703.258881  1.172504e+07  3424.185038
2    6221.823280   78.878535   437.297885  2.623325e+05   512.184061
```

Cafea DataFrame

In [46]:

```
# Filter the initial DataFrame
product = 'cafea'
filter_df = df.loc[df['denumire_produs'] == product]
filter_df = filter_df.drop(columns=['colli', 'store_county'])
#print(apa_plata_df)

# Group by month_id and aggregate the other columns
filter_df_sales = filter_df.groupby("month_id").agg({'sales': 'sum','Inflatie anuala': '
```

```
filter_df_sales = filter_df_sales.sort_values(by='month_id', ascending=True)

# Print the apa_plata_sales dataframe
print(filter_df_sales)
```

month_id	sales	Inflatie anuala	unnemployment	Inflatie(de la o luna la alta)	net_average_salary
2019-01-01	2896.35	3.8	1.42	0.83	2936.0
2019-02-01	2469.45	3.8	3.32	0.79	2933.0
2019-03-01	2624.91	3.8	3.31	0.49	3075.0
2019-04-01	2604.14	3.8	3.19	0.61	3115.0
2019-05-01	3141.86	3.8	3.00	0.46	3101.0
2019-06-01	3059.61	3.8	2.95	-0.23	3142.0
2019-07-01	2500.20	3.8	2.95	-0.20	3119.0
2019-08-01	3287.20	3.8	3.01	0.06	3044.0
2019-09-01	2696.14	3.8	3.03	0.00	3042.0
2019-10-01	2809.71	3.8	3.00	0.00	3042.0
2019-11-01	2692.70	3.8	2.98	0.00	3042.0
2019-12-01	2831.67	3.8	2.98	0.00	3042.0
2020-01-01	2672.82	2.6	2.97	-1.20	3006.0
2020-02-01	3126.66	2.6	2.98	0.00	3006.0
2020-03-01	2672.14	2.6	2.95	0.00	3006.0
2020-04-01	2262.21	2.6	2.88	0.00	3006.0
2020-05-01	2859.85	2.6	2.90	0.00	3006.0
2020-06-01	2971.40	2.6	2.87	0.00	3006.0
2020-07-01	2884.47	2.6	3.00	0.00	3006.0
2020-08-01	3209.29	2.6	3.02	0.00	3006.0
2020-09-01	3301.52	2.6	3.30	0.00	3006.0
2020-10-01	3663.44	2.6	3.26	0.00	3006.0
2020-11-01	2586.76	2.6	3.27	0.00	3006.0
2020-12-01	3424.42	2.6	3.32	0.00	3006.0
2021-01-01	3077.86	5.1	3.38	2.50	3081.0
2021-02-01	3334.40	5.1	3.34	0.00	3081.0
2021-03-01	2711.35	5.1	3.35	0.00	3081.0
2021-04-01	2851.03	5.1	3.33	0.00	3081.0
2021-05-01	3747.35	5.1	3.16	0.00	3081.0
2021-06-01	2412.76	5.1	3.06	0.00	3081.0
2021-07-01	2634.23	5.1	3.00	0.00	3081.0
2021-08-01	2937.70	5.1	2.96	0.00	3081.0
2021-09-01	2996.55	5.1	2.93	0.00	3081.0
2021-10-01	3490.56	5.1	2.85	0.00	3081.0
2021-11-01	2563.84	5.1	2.76	0.00	3081.0
2021-12-01	3542.28	5.1	2.72	0.00	3081.0
2022-01-01	2876.44	13.8	2.69	8.70	3348.0
2022-02-01	3723.59	13.8	2.68	0.00	3348.0
2022-03-01	3571.31	13.8	2.67	0.00	3348.0
2022-04-01	3414.72	13.8	2.64	0.00	3348.0
2022-05-01	3371.58	13.8	2.57	0.00	3348.0
2022-06-01	2912.40	13.8	2.55	0.00	3348.0
2022-07-01	3104.05	13.8	2.55	0.00	3348.0
2022-08-01	4054.42	13.8	2.56	0.00	3348.0
2022-09-01	3883.13	13.8	2.56	0.00	3348.0
2022-10-01	3627.83	13.8	2.88	0.00	3348.0
2022-11-01	3044.88	13.8	2.96	0.00	3348.0
2022-12-01	3688.86	13.8	3.04	0.00	3348.0

2019-09-01	0.09	3082.0
2019-10-01	0.43	3116.0
2019-11-01	0.23	3179.0
2019-12-01	0.42	3340.0
2020-01-01	0.41	3189.0
2020-02-01	0.25	3202.0
2020-03-01	0.50	3294.0
2020-04-01	0.26	3182.0
2020-05-01	0.05	3179.0
2020-06-01	0.08	3298.0
2020-07-01	0.00	3372.0
2020-08-01	-0.05	3275.0
2020-09-01	-0.14	3321.0
2020-10-01	0.22	3343.0
2020-11-01	0.13	3411.0
2020-12-01	0.34	3620.0
2021-01-01	1.33	3395.0
2021-02-01	0.41	3365.0
2021-03-01	0.38	3547.0
2021-04-01	0.45	3561.0
2021-05-01	0.53	3492.0
2021-06-01	0.27	3541.0
2021-07-01	0.97	3545.0
2021-08-01	0.24	3487.0
2021-09-01	0.84	3517.0
2021-10-01	1.78	3544.0
2021-11-01	0.00	3645.0
2021-12-01	0.71	3879.0
2022-01-01	1.48	3698.0
2022-02-01	0.58	3721.0
2022-03-01	1.88	3937.0
2022-04-01	3.74	3967.0
2022-05-01	1.18	3928.0
2022-06-01	0.76	3977.0
2022-07-01	0.89	3975.0
2022-08-01	0.56	3933.0
2022-09-01	1.33	4003.0
2022-10-01	1.28	4008.0
2022-11-01	1.25	4141.0
2022-12-01	0.37	4141.0

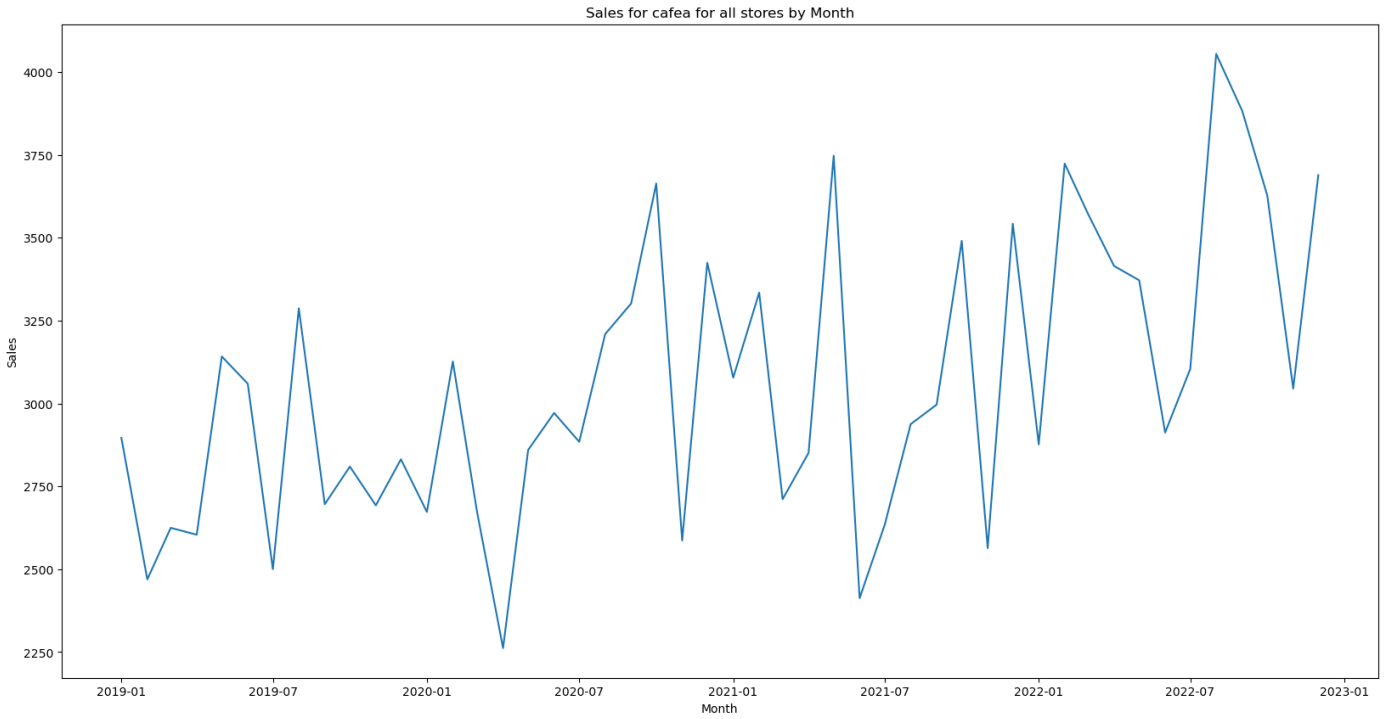
month_id	IPC Marfuri alimentare (%)	IPC Marfuri nealimentare (%)	\
2019-01-01	101.30	100.63	
2019-02-01	101.27	100.57	
2019-03-01	100.48	100.54	
2019-04-01	100.68	100.53	
2019-05-01	100.79	100.20	
2019-06-01	99.53	99.76	
2019-07-01	99.34	99.98	
2019-08-01	99.71	100.22	
2019-09-01	99.91	100.13	
2019-10-01	100.70	100.32	
2019-11-01	100.43	100.12	
2019-12-01	100.84	100.28	
2020-01-01	100.99	100.02	
2020-02-01	100.63	99.94	
2020-03-01	101.46	99.91	
2020-04-01	101.27	99.67	
2020-05-01	100.34	99.82	
2020-06-01	99.62	100.28	
2020-07-01	99.55	100.19	
2020-08-01	99.59	100.08	
2020-09-01	99.45	99.99	
2020-10-01	100.11	100.31	
2020-11-01	99.92	100.29	

2020-12-01	100.29	100.51
2021-01-01	100.63	102.24
2021-02-01	100.46	100.47
2021-03-01	100.37	100.46
2021-04-01	100.45	100.47
2021-05-01	101.10	100.28
2021-06-01	100.25	100.29
2021-07-01	99.71	102.02
2021-08-01	99.95	100.34
2021-09-01	100.95	100.73
2021-10-01	101.06	102.78
2021-11-01	100.73	99.47
2021-12-01	100.84	100.74
2022-01-01	101.15	101.73
2022-02-01	101.96	99.70
2022-03-01	102.54	101.86
2022-04-01	102.56	105.45
2022-05-01	101.73	101.00
2022-06-01	100.62	100.92
2022-07-01	100.92	100.87
2022-08-01	101.82	99.81
2022-09-01	101.72	101.28
2022-10-01	102.29	100.80
2022-11-01	101.54	101.03
2022-12-01	101.26	99.68

month_id	IPC Servicii (%)
2019-01-01	100.57
2019-02-01	100.55
2019-03-01	100.40
2019-04-01	100.72
2019-05-01	100.55
2019-06-01	100.17
2019-07-01	100.10
2019-08-01	100.25
2019-09-01	100.27
2019-10-01	100.25
2019-11-01	100.15
2019-12-01	100.12
2020-01-01	100.43
2020-02-01	100.39
2020-03-01	100.35
2020-04-01	100.00
2020-05-01	100.11
2020-06-01	100.34
2020-07-01	100.31
2020-08-01	100.21
2020-09-01	100.20
2020-10-01	100.22
2020-11-01	100.07
2020-12-01	100.04
2021-01-01	100.25
2021-02-01	100.20
2021-03-01	100.24
2021-04-01	100.40
2021-05-01	100.27
2021-06-01	100.23
2021-07-01	100.39
2021-08-01	100.43
2021-09-01	100.96
2021-10-01	100.42
2021-11-01	100.20
2021-12-01	100.42
2022-01-01	101.37
2022-02-01	100.60

```
2022-03-01            100.67
2022-04-01            100.94
2022-05-01            100.61
2022-06-01            100.54
2022-07-01            100.88
2022-08-01            100.37
2022-09-01            100.72
2022-10-01            100.70
2022-11-01            101.31
2022-12-01            100.67
```

In [47]:
```
# Line graph for apa_plata for all stores
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index.get_level_values(0), filter_df_sales['sales'])
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales for ' + product + ' for all stores by Month')
plt.show()
```

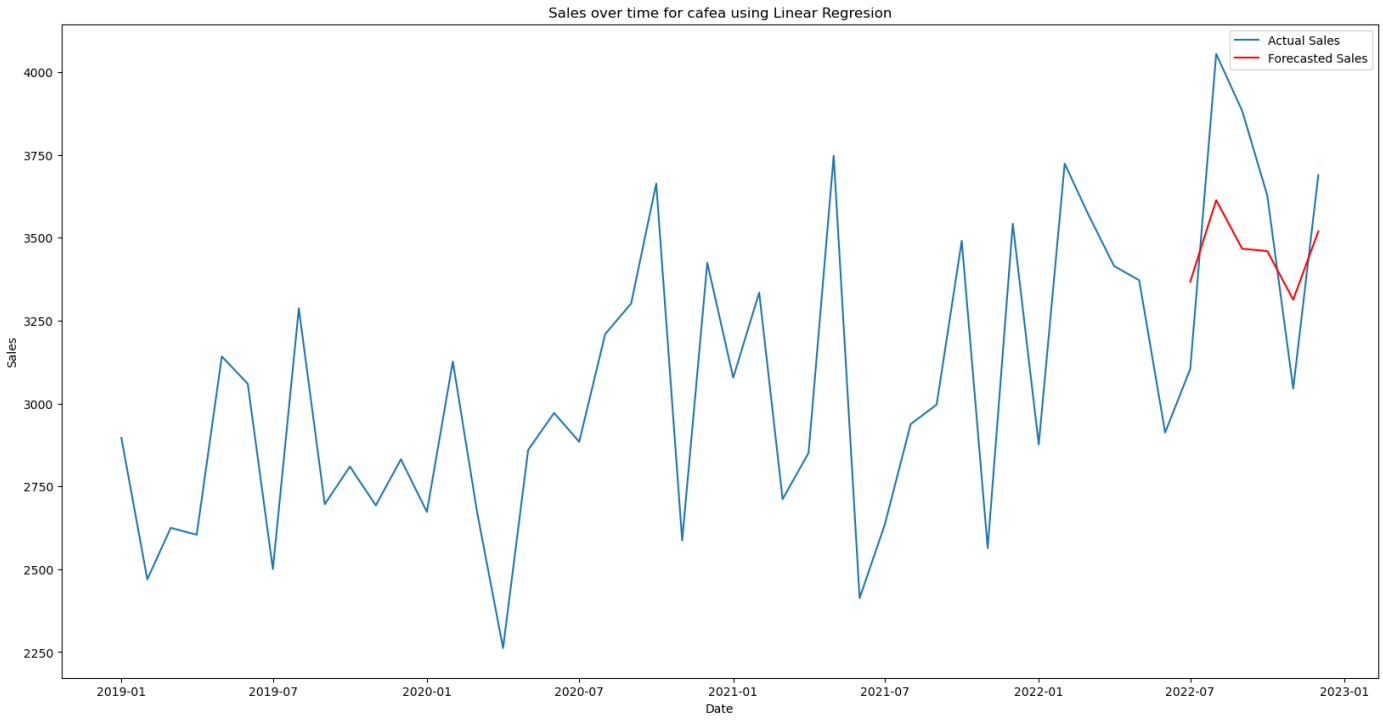


LR - cafea

In [48]:
```
# Get the last 6 months of data
last_6_months = filter_df_sales.tail(6)

# Split the data into train and test sets
X_train = filter_df_sales.drop('sales', axis=1)
y_train = filter_df_sales['sales']
X_test = last_6_months.drop('sales', axis=1)
y_test = last_6_months['sales']

# Create a linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the error
```

```
lr_mae = mean_absolute_error(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)

print("Mean Absolute Error: ", lr_mae)
print("Mean Squared Error: ", lr_mse)
print("Root Mean Squared Error:", lr_rmse)

all_sales = filter_df_sales['sales']

# Get the index of the last 6 months of data
last_6_months_index = last_6_months.index

# Plot the entire period of sales data
plt.figure(figsize=(20,10))
plt.plot(all_sales)

# Make predictions on the last 6 months of data
y_pred = model.predict(X_test)

# Plot the forecasted sales on the same chart
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Linear Regresion')
plt.legend(['Actual Sales', 'Forecasted Sales'])

plt.show()
```

```
Mean Absolute Error:  287.81826105650833
Mean Squared Error:  94385.21301339782
Root Mean Squared Error: 307.22176520129204
```



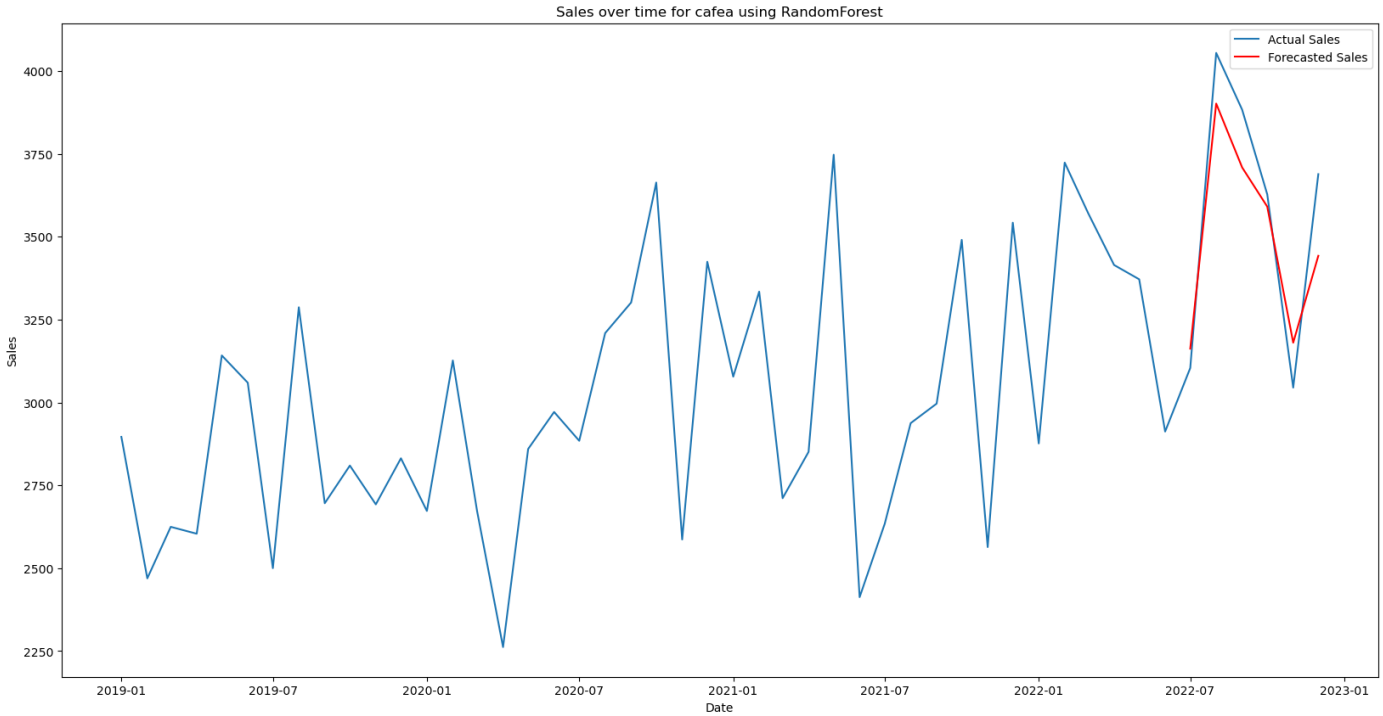
RF - cafea

In [49]:

```
last_6_months = filter_df_sales.tail(6)

X_train = filter_df_sales.drop(columns=["sales"])
y_train = filter_df_sales["sales"]
X_test = last_6_months.drop(columns=["sales"])
```

```
y_test = last_6_months["sales"]

# Create a random forest model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Use the model to make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate the model's accuracy
accuracy = rf.score(X_test, y_test)
print("Accuracy: ", accuracy)

# Calculate error
rf_mae = mean_absolute_error(y_test, y_pred)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rf_mse)

print("Mean Absolute Error: ", rf_mae)
print("Mean Squared Error: ", rf_mse)
print("Root Mean Squared Error:", rf_rmse)

# Line graph for the forcast
plt.figure(figsize=(20,10))

plt.plot(all_sales)
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using RandomForest')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

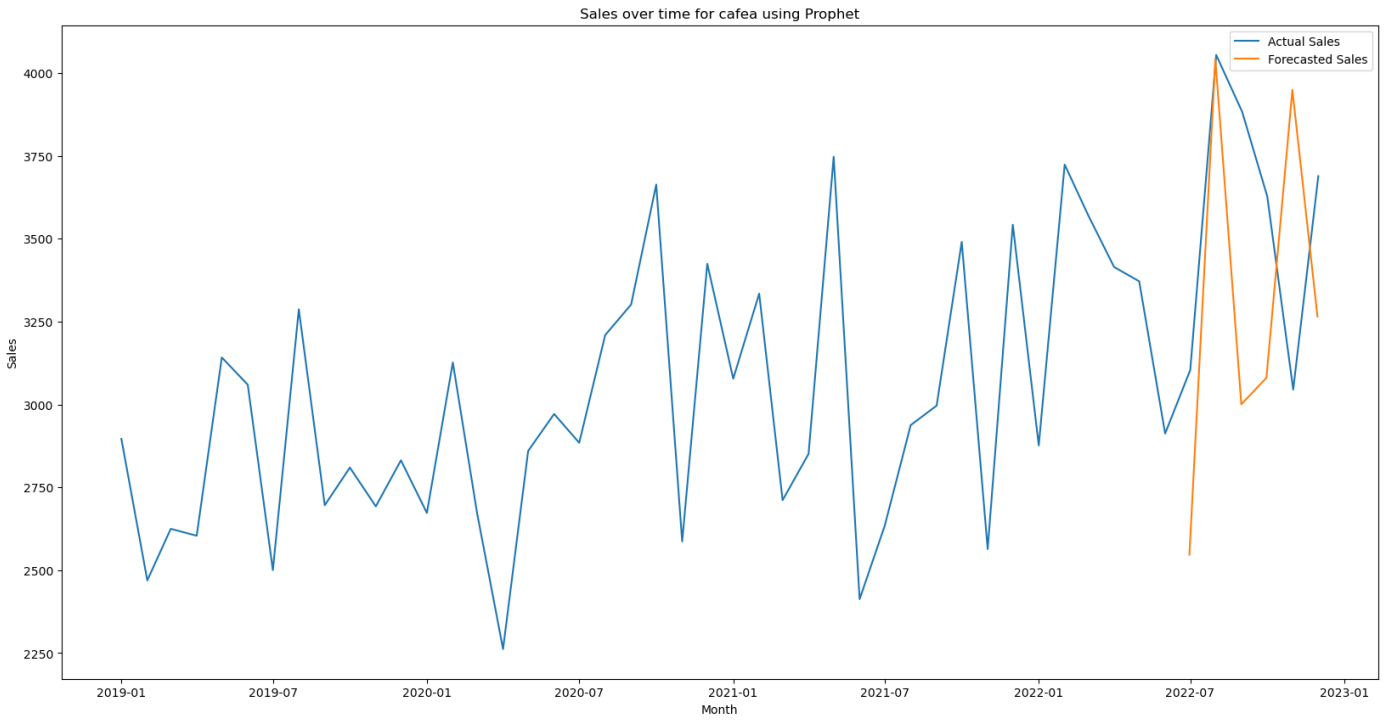
```
Accuracy:  0.8368601008016994
Mean Absolute Error:  134.03549999999981
Mean Squared Error:  22921.110988843455
Root Mean Squared Error: 151.39719610628018
```



PR - cafea

In [50]:
```
# create a copy of the dataframe with a column named 'ds' with the index values
filter_df_sales_prophet = filter_df_sales.reset_index().rename(columns={'month_id': 'ds'
filter_df_sales_prophet = filter_df_sales_prophet.rename(columns={'sales':'y'})

train_data = filter_df_sales_prophet[:-6]
test_data = filter_df_sales_prophet[-6:]

model = Prophet()

# Fit the model on the data
model.fit(train_data)

# Create a dataframe to hold the forecast
future_data = model.make_future_dataframe(periods=6, freq='m')

# Make predictions
forecast = model.predict(future_data)
forecast = forecast.tail(6)


# Calculate error
pr_mae = mean_absolute_error(test_data['y'], forecast[-6:]['yhat'])
pr_mse = mean_squared_error(test_data['y'], forecast[-6:]['yhat'])
pr_rmse = np.sqrt(pr_mse)
print("Mean Absolute Error:", pr_mae)
print("Mean Squared Error:", pr_mse)
print("Root Mean Squared Error:", pr_rmse)

# Plot the forecast
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index, filter_df_sales['sales'], label='Actual Sales')

# Plot the predicted values for the last six months
plt.plot(forecast.ds, forecast.yhat, label='Predicted Sales')

plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Prophet')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
21:35:10 - cmdstanpy - INFO - Chain [1] start processing
21:35:10 - cmdstanpy - INFO - Chain [1] done processing
Mean Absolute Error: 554.5671750212072
Mean Squared Error: 397664.1044723328
Root Mean Squared Error: 630.6061405285654
```



In [51]:
```
# Add the errors from all methods to the DataFrame
new_row_df =  pd.DataFrame({'product': [product], 'LR-MAE': [lr_mae], 'LR-MSE': [lr_mse]
errors_df = pd.concat([errors_df, new_row_df], ignore_index=True)
print(errors_df)
```

```
        product      LR-MAE         LR-MSE     LR-RMSE      RF-MAE  \
0     apa_plata  887.015829  988186.848307  994.075877  377.426267
1  apa_minerala   44.324535    2453.366899   49.531474   10.533100
2          bere  283.109792   95844.002658  309.586826   65.444150
3         cafea  287.818261   94385.213013  307.221765  134.035500

          RF-MSE     RF-RMSE       PR-MAE        PR-MSE      PR-RMSE
0  200759.770117  448.062239  1107.910649  1.661801e+06  1289.108561
1     153.080170   12.372557  1703.258881  1.172504e+07  3424.185038
2    6221.823280   78.878535   437.297885  2.623325e+05   512.184061
3   22921.110989  151.397196   554.567175  3.976641e+05   630.606141
```

Carnati DataFrame

In [52]:
```
# Filter the initial DataFrame
product = 'carnati'
filter_df = df.loc[df['denumire_produs'] == product]
filter_df = filter_df.drop(columns=['colli', 'store_county'])
#print(apa_plata_df)

# Group by month_id and aggregate the other columns
filter_df_sales = filter_df.groupby("month_id").agg({'sales': 'sum','Inflatie anuala': '
filter_df_sales = filter_df_sales.sort_values(by='month_id', ascending=True)

# Print the apa_plata_sales dataframe
print(filter_df_sales)
```

```
              sales  Inflatie anuala  unnemployment  \
month_id
2019-01-01  1017.35              3.8           1.42
2019-02-01  1052.80              3.8           3.32
2019-03-01  1522.25              3.8           3.31
2019-04-01  1353.29              3.8           3.19
2019-05-01  1948.06              3.8           3.00
2019-06-01  1916.72              3.8           2.95
```

2019-07-01	2473.80	3.8	2.95
2019-08-01	1558.64	3.8	3.01
2019-09-01	2029.26	3.8	3.03
2019-10-01	1875.39	3.8	3.00
2019-11-01	2263.74	3.8	2.98
2019-12-01	2714.40	3.8	2.98
2020-01-01	2258.95	2.6	2.97
2020-02-01	1788.19	2.6	2.98
2020-03-01	2141.81	2.6	2.95
2020-04-01	1687.56	2.6	2.88
2020-05-01	2206.91	2.6	2.90
2020-06-01	1969.11	2.6	2.87
2020-07-01	2674.57	2.6	3.00
2020-08-01	2264.94	2.6	3.02
2020-09-01	2299.24	2.6	3.30
2020-10-01	2700.27	2.6	3.26
2020-11-01	2398.47	2.6	3.27
2020-12-01	2235.57	2.6	3.32
2021-01-01	1473.87	5.1	3.38
2021-02-01	1250.65	5.1	3.34
2021-03-01	1831.42	5.1	3.35
2021-04-01	2240.43	5.1	3.33
2021-05-01	1692.62	5.1	3.16
2021-06-01	2408.03	5.1	3.06
2021-07-01	2152.32	5.1	3.00
2021-08-01	2324.38	5.1	2.96
2021-09-01	2749.40	5.1	2.93
2021-10-01	2363.18	5.1	2.85
2021-11-01	2086.23	5.1	2.76
2021-12-01	2663.56	5.1	2.72
2022-01-01	1479.24	13.8	2.69
2022-02-01	1701.96	13.8	2.68
2022-03-01	3062.63	13.8	2.67
2022-04-01	2234.96	13.8	2.64
2022-05-01	3032.58	13.8	2.57
2022-06-01	2225.86	13.8	2.55
2022-07-01	2875.09	13.8	2.55
2022-08-01	2443.74	13.8	2.56
2022-09-01	2723.38	13.8	2.56
2022-10-01	2093.80	13.8	2.88
2022-11-01	3230.12	13.8	2.96
2022-12-01	3101.97	13.8	3.04

month_id	Inflatie(de la o luna la alta)	net_average_salary	\
2019-01-01	0.83	2936.0	
2019-02-01	0.79	2933.0	
2019-03-01	0.49	3075.0	
2019-04-01	0.61	3115.0	
2019-05-01	0.46	3101.0	
2019-06-01	-0.23	3142.0	
2019-07-01	-0.20	3119.0	
2019-08-01	0.06	3044.0	
2019-09-01	0.09	3082.0	
2019-10-01	0.43	3116.0	
2019-11-01	0.23	3179.0	
2019-12-01	0.42	3340.0	
2020-01-01	0.41	3189.0	
2020-02-01	0.25	3202.0	
2020-03-01	0.50	3294.0	
2020-04-01	0.26	3182.0	
2020-05-01	0.05	3179.0	
2020-06-01	0.08	3298.0	
2020-07-01	0.00	3372.0	
2020-08-01	-0.05	3275.0	
2020-09-01	-0.14	3321.0	

2020-10-01	0.22	3343.0
2020-11-01	0.13	3411.0
2020-12-01	0.34	3620.0
2021-01-01	1.33	3395.0
2021-02-01	0.41	3365.0
2021-03-01	0.38	3547.0
2021-04-01	0.45	3561.0
2021-05-01	0.53	3492.0
2021-06-01	0.27	3541.0
2021-07-01	0.97	3545.0
2021-08-01	0.24	3487.0
2021-09-01	0.84	3517.0
2021-10-01	1.78	3544.0
2021-11-01	0.00	3645.0
2021-12-01	0.71	3879.0
2022-01-01	1.48	3698.0
2022-02-01	0.58	3721.0
2022-03-01	1.88	3937.0
2022-04-01	3.74	3967.0
2022-05-01	1.18	3928.0
2022-06-01	0.76	3977.0
2022-07-01	0.89	3975.0
2022-08-01	0.56	3933.0
2022-09-01	1.33	4003.0
2022-10-01	1.28	4008.0
2022-11-01	1.25	4141.0
2022-12-01	0.37	4141.0

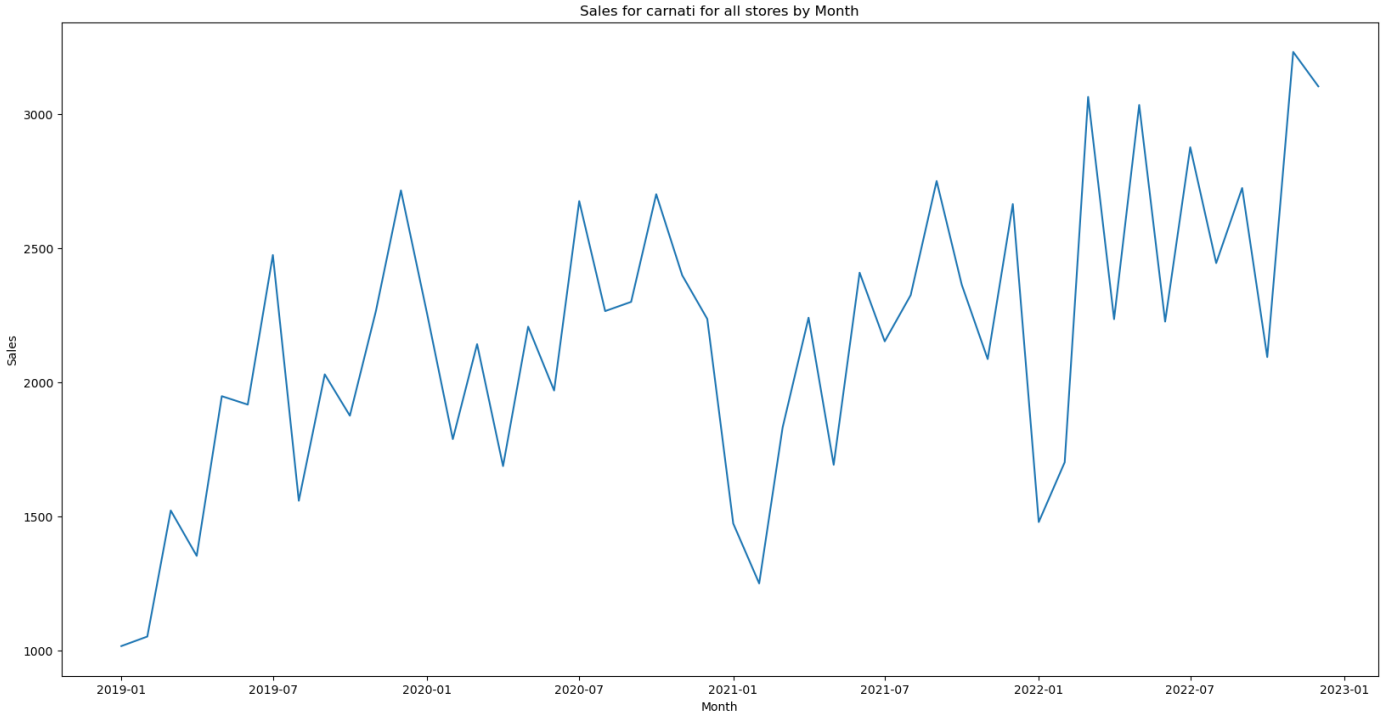
month_id	IPC Marfuri alimentare (%)	IPC Marfuri nealimentare (%)	\
2019-01-01	101.30	100.63	
2019-02-01	101.27	100.57	
2019-03-01	100.48	100.54	
2019-04-01	100.68	100.53	
2019-05-01	100.79	100.20	
2019-06-01	99.53	99.76	
2019-07-01	99.34	99.98	
2019-08-01	99.71	100.22	
2019-09-01	99.91	100.13	
2019-10-01	100.70	100.32	
2019-11-01	100.43	100.12	
2019-12-01	100.84	100.28	
2020-01-01	100.99	100.02	
2020-02-01	100.63	99.94	
2020-03-01	101.46	99.91	
2020-04-01	101.27	99.67	
2020-05-01	100.34	99.82	
2020-06-01	99.62	100.28	
2020-07-01	99.55	100.19	
2020-08-01	99.59	100.08	
2020-09-01	99.45	99.99	
2020-10-01	100.11	100.31	
2020-11-01	99.92	100.29	
2020-12-01	100.29	100.51	
2021-01-01	100.63	102.24	
2021-02-01	100.46	100.47	
2021-03-01	100.37	100.46	
2021-04-01	100.45	100.47	
2021-05-01	101.10	100.28	
2021-06-01	100.25	100.29	
2021-07-01	99.71	102.02	
2021-08-01	99.95	100.34	
2021-09-01	100.95	100.73	
2021-10-01	101.06	102.78	
2021-11-01	100.73	99.47	
2021-12-01	100.84	100.74	

```
2022-01-01                    101.15                    101.73
2022-02-01                    101.96                     99.70
2022-03-01                    102.54                    101.86
2022-04-01                    102.56                    105.45
2022-05-01                    101.73                    101.00
2022-06-01                    100.62                    100.92
2022-07-01                    100.92                    100.87
2022-08-01                    101.82                     99.81
2022-09-01                    101.72                    101.28
2022-10-01                    102.29                    100.80
2022-11-01                    101.54                    101.03
2022-12-01                    101.26                     99.68
```

```
            IPC Servicii (%)
month_id
2019-01-01            100.57
2019-02-01            100.55
2019-03-01            100.40
2019-04-01            100.72
2019-05-01            100.55
2019-06-01            100.17
2019-07-01            100.10
2019-08-01            100.25
2019-09-01            100.27
2019-10-01            100.25
2019-11-01            100.15
2019-12-01            100.12
2020-01-01            100.43
2020-02-01            100.39
2020-03-01            100.35
2020-04-01            100.00
2020-05-01            100.11
2020-06-01            100.34
2020-07-01            100.31
2020-08-01            100.21
2020-09-01            100.20
2020-10-01            100.22
2020-11-01            100.07
2020-12-01            100.04
2021-01-01            100.25
2021-02-01            100.20
2021-03-01            100.24
2021-04-01            100.40
2021-05-01            100.27
2021-06-01            100.23
2021-07-01            100.39
2021-08-01            100.43
2021-09-01            100.96
2021-10-01            100.42
2021-11-01            100.20
2021-12-01            100.42
2022-01-01            101.37
2022-02-01            100.60
2022-03-01            100.67
2022-04-01            100.94
2022-05-01            100.61
2022-06-01            100.54
2022-07-01            100.88
2022-08-01            100.37
2022-09-01            100.72
2022-10-01            100.70
2022-11-01            101.31
2022-12-01            100.67
```

In [53]:
```
# Line graph for apa_plata for all stores
```

```
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index.get_level_values(0), filter_df_sales['sales'])
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales for ' + product + ' for all stores by Month')
plt.show()
```



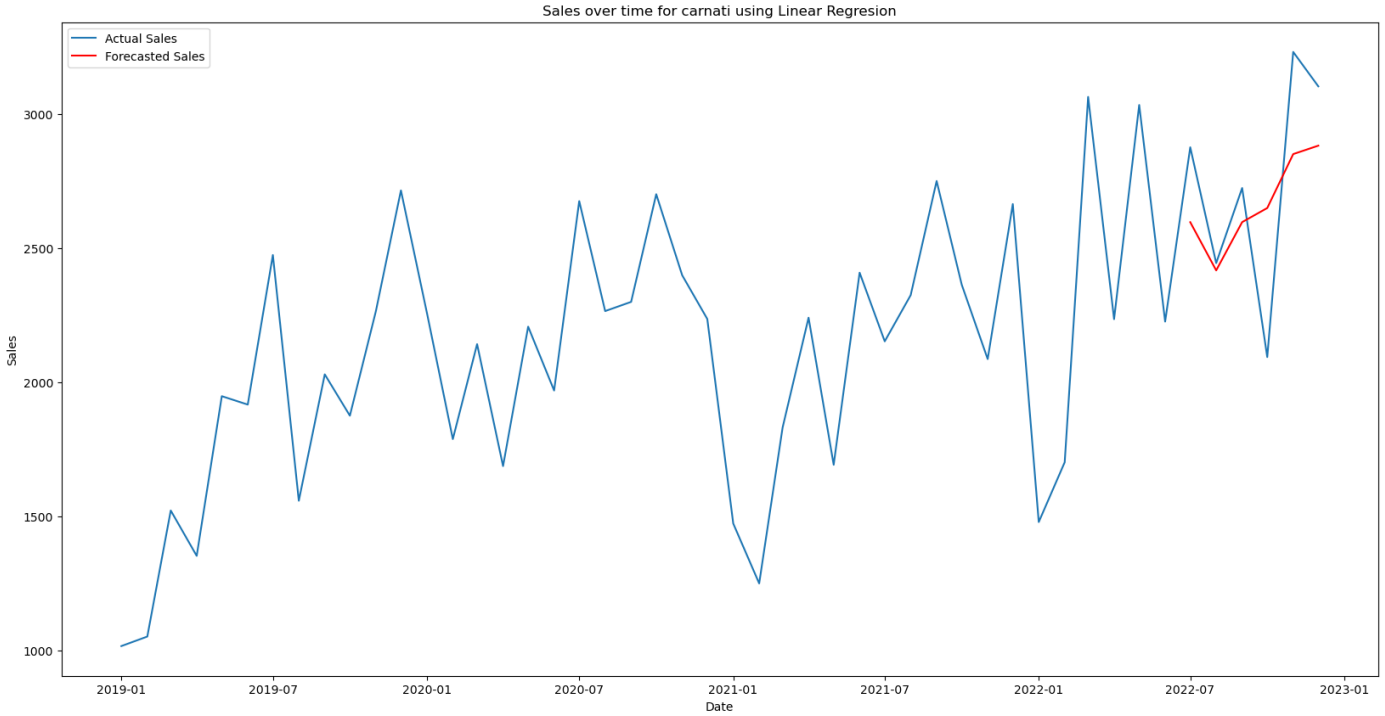
LR - carnati

In [54]:

```
# Get the last 6 months of data
last_6_months = filter_df_sales.tail(6)

# Split the data into train and test sets
X_train = filter_df_sales.drop('sales', axis=1)
y_train = filter_df_sales['sales']
X_test = last_6_months.drop('sales', axis=1)
y_test = last_6_months['sales']

# Create a linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the error
lr_mae = mean_absolute_error(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)

print("Mean Absolute Error: ", lr_mae)
print("Mean Squared Error: ", lr_mse)
print("Root Mean Squared Error:", lr_rmse)

all_sales = filter_df_sales['sales']

# Get the index of the last 6 months of data
last_6_months_index = last_6_months.index
```

```
# Plot the entire period of sales data
plt.figure(figsize=(20,10))
plt.plot(all_sales)

# Make predictions on the last 6 months of data
y_pred = model.predict(X_test)

# Plot the forecasted sales on the same chart
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Linear Regresion')
plt.legend(['Actual Sales', 'Forecasted Sales'])

plt.show()
```

```
Mean Absolute Error:  264.93659007545403
Mean Squared Error:  99377.94753383241
Root Mean Squared Error: 315.24268038105566
```



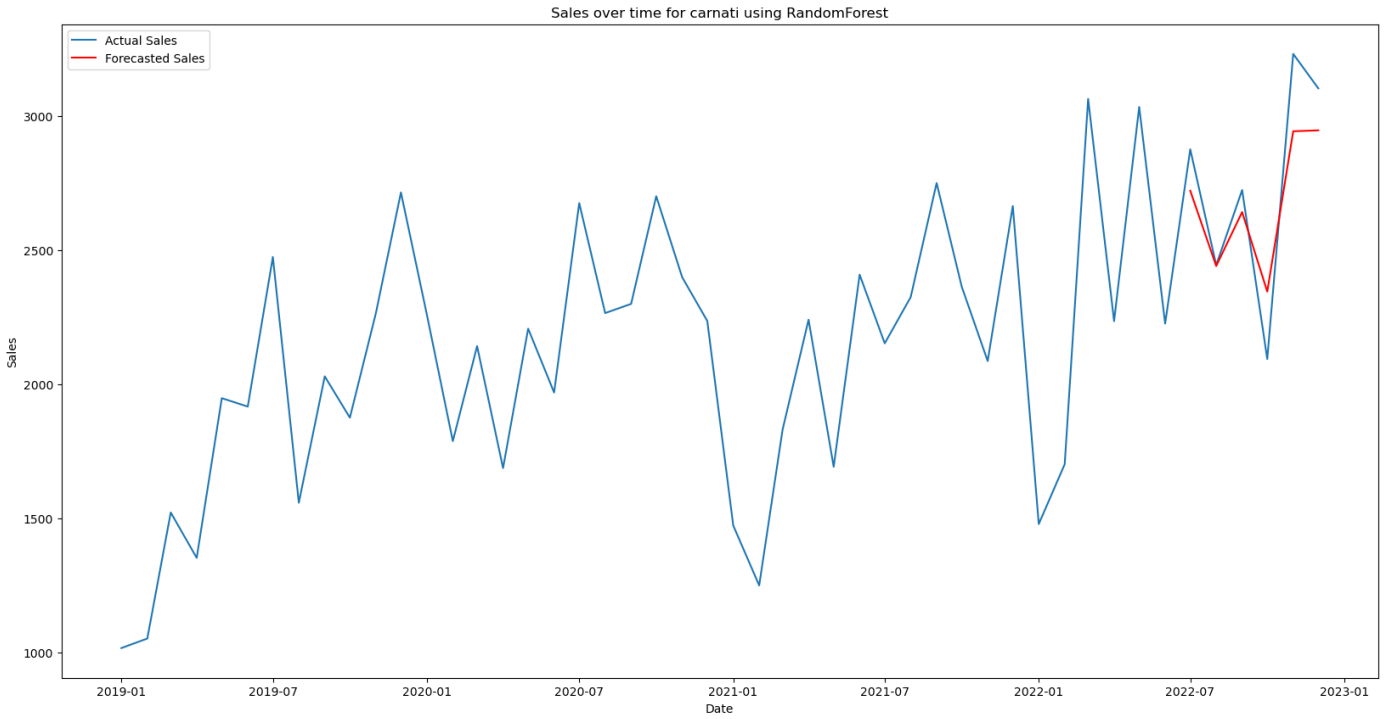
RF - carnati

In [55]:

```
last_6_months = filter_df_sales.tail(6)

X_train = filter_df_sales.drop(columns=["sales"])
y_train = filter_df_sales["sales"]
X_test = last_6_months.drop(columns=["sales"])
y_test = last_6_months["sales"]

# Create a random forest model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Use the model to make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate the model's accuracy
accuracy = rf.score(X_test, y_test)
print("Accuracy: ", accuracy)
```

```
# Calculate error
rf_mae = mean_absolute_error(y_test, y_pred)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rf_mse)

print("Mean Absolute Error: ", rf_mae)
print("Mean Squared Error: ", rf_mse)
print("Root Mean Squared Error:", rf_rmse)

# Line graph for the forcast
plt.figure(figsize=(20,10))

plt.plot(all_sales)
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using RandomForest')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
Accuracy:  0.7752580673322235
Mean Absolute Error:  155.93400000000088
Mean Squared Error:  33523.19040661016
Root Mean Squared Error: 183.09339258042647
```



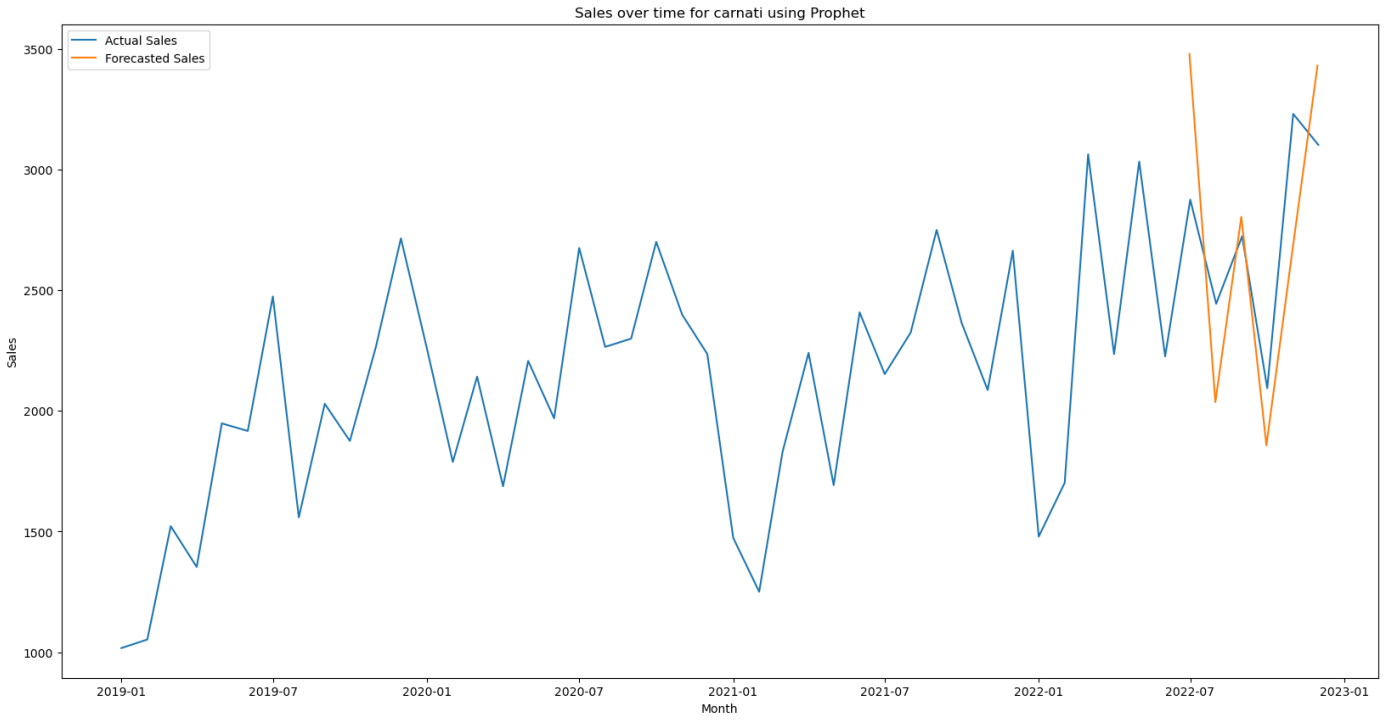
PR - carnati

In [56]:

```
# create a copy of the dataframe with a column named 'ds' with the index values
filter_df_sales_prophet = filter_df_sales.reset_index().rename(columns={'month_id': 'ds'
filter_df_sales_prophet = filter_df_sales_prophet.rename(columns={'sales':'y'})

train_data = filter_df_sales_prophet[:-6]
test_data = filter_df_sales_prophet[-6:]

model = Prophet()

# Fit the model on the data
model.fit(train_data)
```

```
# Create a dataframe to hold the forecast
future_data = model.make_future_dataframe(periods=6, freq='m')

# Make predictions
forecast = model.predict(future_data)
forecast = forecast.tail(6)


# Calculate error
pr_mae = mean_absolute_error(test_data['y'], forecast[-6:]['yhat'])
pr_mse = mean_squared_error(test_data['y'], forecast[-6:]['yhat'])
pr_rmse = np.sqrt(pr_mse)
print("Mean Absolute Error:", pr_mae)
print("Mean Squared Error:", pr_mse)
print("Root Mean Squared Error:", pr_rmse)

# Plot the forecast
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index, filter_df_sales['sales'], label='Actual Sales')

# Plot the predicted values for the last six months
plt.plot(forecast.ds, forecast.yhat, label='Predicted Sales')

plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Prophet')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
21:35:11 - cmdstanpy - INFO - Chain [1] start processing
21:35:12 - cmdstanpy - INFO - Chain [1] done processing
Mean Absolute Error: 370.439784779538
Mean Squared Error: 170275.97382631936
Root Mean Squared Error: 412.6450942714809
```



In [57]:

```
# Add the errors from all methods to the DataFrame
new_row_df =  pd.DataFrame({'product': [product], 'LR-MAE': [lr_mae], 'LR-MSE': [lr_mse]
errors_df = pd.concat([errors_df, new_row_df], ignore_index=True)
print(errors_df)
```

```
        product      LR-MAE         LR-MSE     LR-RMSE      RF-MAE  \
0     apa_plata  887.015829  988186.848307  994.075877  377.426267
1  apa_minerala   44.324535    2453.366899   49.531474   10.533100
```

```
2          bere  283.109792   95844.002658  309.586826   65.444150
3         cafea  287.818261   94385.213013  307.221765  134.035500
4       carnati  264.936590   99377.947534  315.242680  155.934000

          RF-MSE     RF-RMSE       PR-MAE        PR-MSE      PR-RMSE
0  200759.770117  448.062239  1107.910649  1.661801e+06  1289.108561
1     153.080170   12.372557  1703.258881  1.172504e+07  3424.185038
2    6221.823280   78.878535   437.297885  2.623325e+05   512.184061
3   22921.110989  151.397196   554.567175  3.976641e+05   630.606141
4   33523.190407  183.093393   370.439785  1.702760e+05   412.645094
```

Frigider DataFrame

In [58]:
```
# Filter the initial DataFrame
product = 'frigider'
filter_df = df.loc[df['denumire_produs'] == product]
filter_df = filter_df.drop(columns=['colli', 'store_county'])
#print(apa_plata_df)

# Group by month_id and aggregate the other columns
filter_df_sales = filter_df.groupby("month_id").agg({'sales': 'sum','Inflatie anuala': '
filter_df_sales = filter_df_sales.sort_values(by='month_id', ascending=True)

# Print the apa_plata_sales dataframe
print(filter_df_sales)
```

```
             sales  Inflatie anuala  unnemployment  \
month_id
2019-01-01   13.84              3.8           1.42
2019-02-01   53.45              3.8           3.32
2019-03-01   34.24              3.8           3.31
2019-04-01   17.90              3.8           3.19
2019-05-01   27.68              3.8           3.00
2019-06-01    6.92              3.8           2.95
2019-08-01   27.68              3.8           3.01
2019-09-01   20.04              3.8           3.03
2019-10-01   61.92              3.8           3.00
2019-11-01  121.63              3.8           2.98
2019-12-01   26.60              3.8           2.98
2020-01-01   13.84              2.6           2.97
2020-02-01   26.96              2.6           2.98
2020-03-01   13.84              2.6           2.95
2020-04-01   13.48              2.6           2.88
2020-05-01    6.92              2.6           2.90
2020-06-01   26.82              2.6           2.87
2020-07-01    6.92              2.6           3.00
2020-08-01   42.63              2.6           3.02
2020-09-01    6.92              2.6           3.30
2020-10-01   13.84              2.6           3.26
2020-12-01    6.92              2.6           3.32
2021-01-01   35.62              5.1           3.38
2021-02-01   50.14              5.1           3.34
2021-03-01   16.87              5.1           3.35
2021-04-01    6.92              5.1           3.33
2021-05-01    8.52              5.1           3.16
2021-06-01   26.85              5.1           3.06
2021-07-01   44.20              5.1           3.00
2021-08-01   19.64              5.1           2.96
2021-09-01   35.36              5.1           2.93
2021-10-01   39.28              5.1           2.85
2021-12-01    8.52              5.1           2.72
2022-06-01   70.72             13.8           2.55
2022-11-01   10.91             13.8           2.96
2022-12-01   19.64             13.8           3.04
```

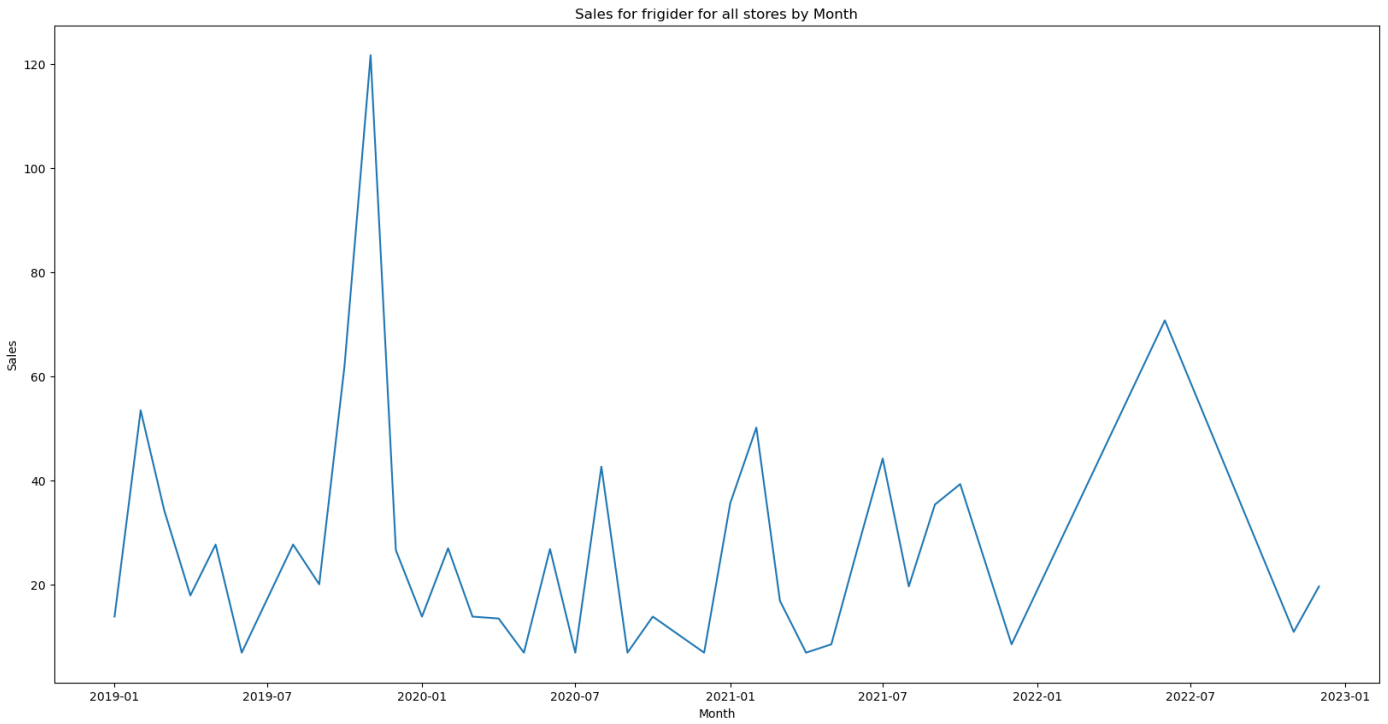
month_id	Inflatie(de la o luna la alta)	net_average_salary	\
2019-01-01	0.83	2936.0	
2019-02-01	0.79	2933.0	
2019-03-01	0.49	3075.0	
2019-04-01	0.61	3115.0	
2019-05-01	0.46	3101.0	
2019-06-01	-0.23	3142.0	
2019-08-01	0.06	3044.0	
2019-09-01	0.09	3082.0	
2019-10-01	0.43	3116.0	
2019-11-01	0.23	3179.0	
2019-12-01	0.42	3340.0	
2020-01-01	0.41	3189.0	
2020-02-01	0.25	3202.0	
2020-03-01	0.50	3294.0	
2020-04-01	0.26	3182.0	
2020-05-01	0.05	3179.0	
2020-06-01	0.08	3298.0	
2020-07-01	0.00	3372.0	
2020-08-01	-0.05	3275.0	
2020-09-01	-0.14	3321.0	
2020-10-01	0.22	3343.0	
2020-12-01	0.34	3620.0	
2021-01-01	1.33	3395.0	
2021-02-01	0.41	3365.0	
2021-03-01	0.38	3547.0	
2021-04-01	0.45	3561.0	
2021-05-01	0.53	3492.0	
2021-06-01	0.27	3541.0	
2021-07-01	0.97	3545.0	
2021-08-01	0.24	3487.0	
2021-09-01	0.84	3517.0	
2021-10-01	1.78	3544.0	
2021-12-01	0.71	3879.0	
2022-06-01	0.76	3977.0	
2022-11-01	1.25	4141.0	
2022-12-01	0.37	4141.0	

month_id	IPC Marfuri alimentare (%)	IPC Marfuri nealimentare (%)	\
2019-01-01	101.30	100.63	
2019-02-01	101.27	100.57	
2019-03-01	100.48	100.54	
2019-04-01	100.68	100.53	
2019-05-01	100.79	100.20	
2019-06-01	99.53	99.76	
2019-08-01	99.71	100.22	
2019-09-01	99.91	100.13	
2019-10-01	100.70	100.32	
2019-11-01	100.43	100.12	
2019-12-01	100.84	100.28	
2020-01-01	100.99	100.02	
2020-02-01	100.63	99.94	
2020-03-01	101.46	99.91	
2020-04-01	101.27	99.67	
2020-05-01	100.34	99.82	
2020-06-01	99.62	100.28	
2020-07-01	99.55	100.19	
2020-08-01	99.59	100.08	
2020-09-01	99.45	99.99	
2020-10-01	100.11	100.31	
2020-12-01	100.29	100.51	
2021-01-01	100.63	102.24	
2021-02-01	100.46	100.47	

```
2021-03-01                    100.37                    100.46
2021-04-01                    100.45                    100.47
2021-05-01                    101.10                    100.28
2021-06-01                    100.25                    100.29
2021-07-01                     99.71                    102.02
2021-08-01                     99.95                    100.34
2021-09-01                    100.95                    100.73
2021-10-01                    101.06                    102.78
2021-12-01                    100.84                    100.74
2022-06-01                    100.62                    100.92
2022-11-01                    101.54                    101.03
2022-12-01                    101.26                     99.68
```

```
            IPC Servicii (%)
month_id
2019-01-01             100.57
2019-02-01             100.55
2019-03-01             100.40
2019-04-01             100.72
2019-05-01             100.55
2019-06-01             100.17
2019-08-01             100.25
2019-09-01             100.27
2019-10-01             100.25
2019-11-01             100.15
2019-12-01             100.12
2020-01-01             100.43
2020-02-01             100.39
2020-03-01             100.35
2020-04-01             100.00
2020-05-01             100.11
2020-06-01             100.34
2020-07-01             100.31
2020-08-01             100.21
2020-09-01             100.20
2020-10-01             100.22
2020-12-01             100.04
2021-01-01             100.25
2021-02-01             100.20
2021-03-01             100.24
2021-04-01             100.40
2021-05-01             100.27
2021-06-01             100.23
2021-07-01             100.39
2021-08-01             100.43
2021-09-01             100.96
2021-10-01             100.42
2021-12-01             100.42
2022-06-01             100.54
2022-11-01             101.31
2022-12-01             100.67
```

In [59]:
```
# Line graph for apa_plata for all stores
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index.get_level_values(0), filter_df_sales['sales'])
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales for ' + product + ' for all stores by Month')
plt.show()
```

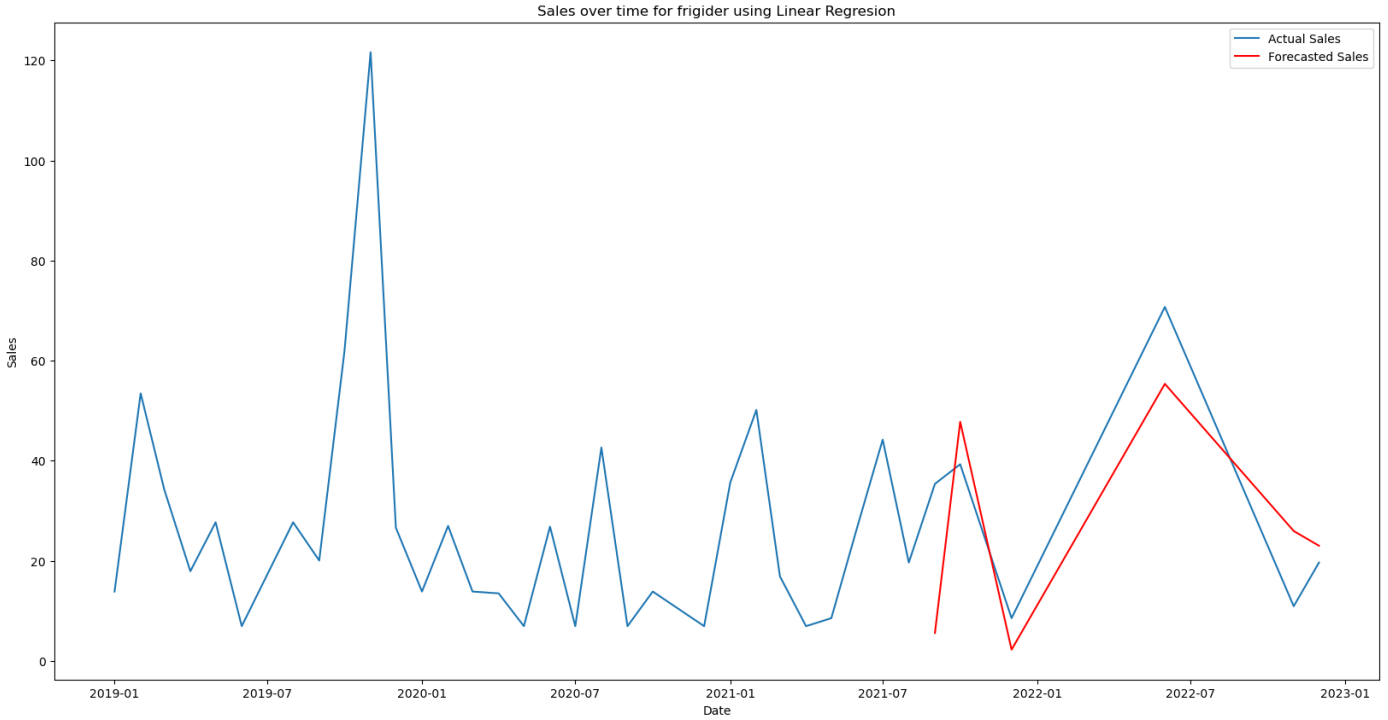


LR - frigider

In [60]:
```
# Get the last 6 months of data
last_6_months = filter_df_sales.tail(6)

# Split the data into train and test sets
X_train = filter_df_sales.drop('sales', axis=1)
y_train = filter_df_sales['sales']
X_test = last_6_months.drop('sales', axis=1)
y_test = last_6_months['sales']

# Create a linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the error
lr_mae = mean_absolute_error(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)

print("Mean Absolute Error: ", lr_mae)
print("Mean Squared Error: ", lr_mse)
print("Root Mean Squared Error:", lr_rmse)

all_sales = filter_df_sales['sales']

# Get the index of the last 6 months of data
last_6_months_index = last_6_months.index

# Plot the entire period of sales data
plt.figure(figsize=(20,10))
plt.plot(all_sales)

# Make predictions on the last 6 months of data
```

```
y_pred = model.predict(X_test)

# Plot the forecasted sales on the same chart
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Linear Regresion')
plt.legend(['Actual Sales', 'Forecasted Sales'])

plt.show()
```

```
Mean Absolute Error:  13.05462556894956
Mean Squared Error:  245.44180321442548
Root Mean Squared Error: 15.666582371864818
```



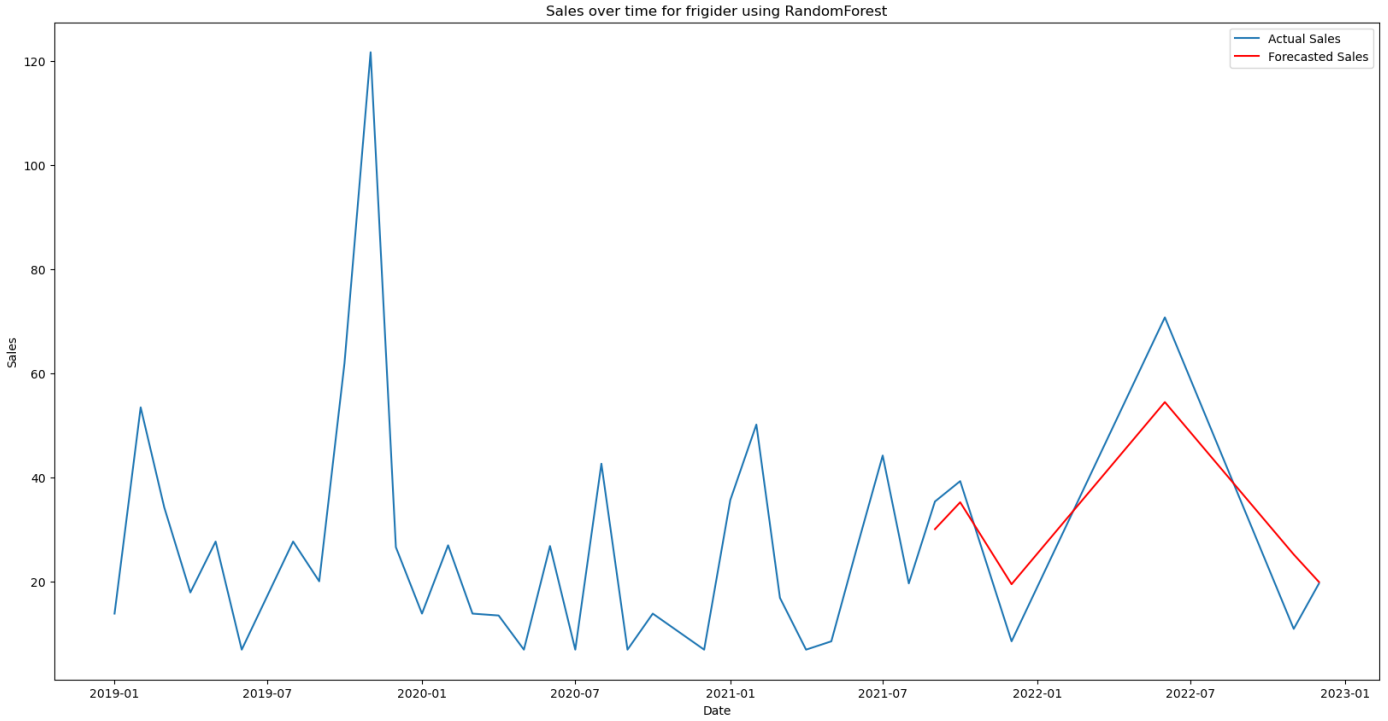
RF - frigider

In [61]:

```
last_6_months = filter_df_sales.tail(6)

X_train = filter_df_sales.drop(columns=["sales"])
y_train = filter_df_sales["sales"]
X_test = last_6_months.drop(columns=["sales"])
y_test = last_6_months["sales"]

# Create a random forest model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Use the model to make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate the model's accuracy
accuracy = rf.score(X_test, y_test)
print("Accuracy: ", accuracy)

# Calculate error
rf_mae = mean_absolute_error(y_test, y_pred)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rf_mse)
```

```
print("Mean Absolute Error: ", rf_mae)
print("Mean Squared Error: ", rf_mse)
print("Root Mean Squared Error:", rf_rmse)

# Line graph for the forcast
plt.figure(figsize=(20,10))

plt.plot(all_sales)
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using RandomForest')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
Accuracy:  0.7656136168708394
Mean Absolute Error:  8.523899999999992
Mean Squared Error:  105.58495918333308
Root Mean Squared Error: 10.275454208127886
```



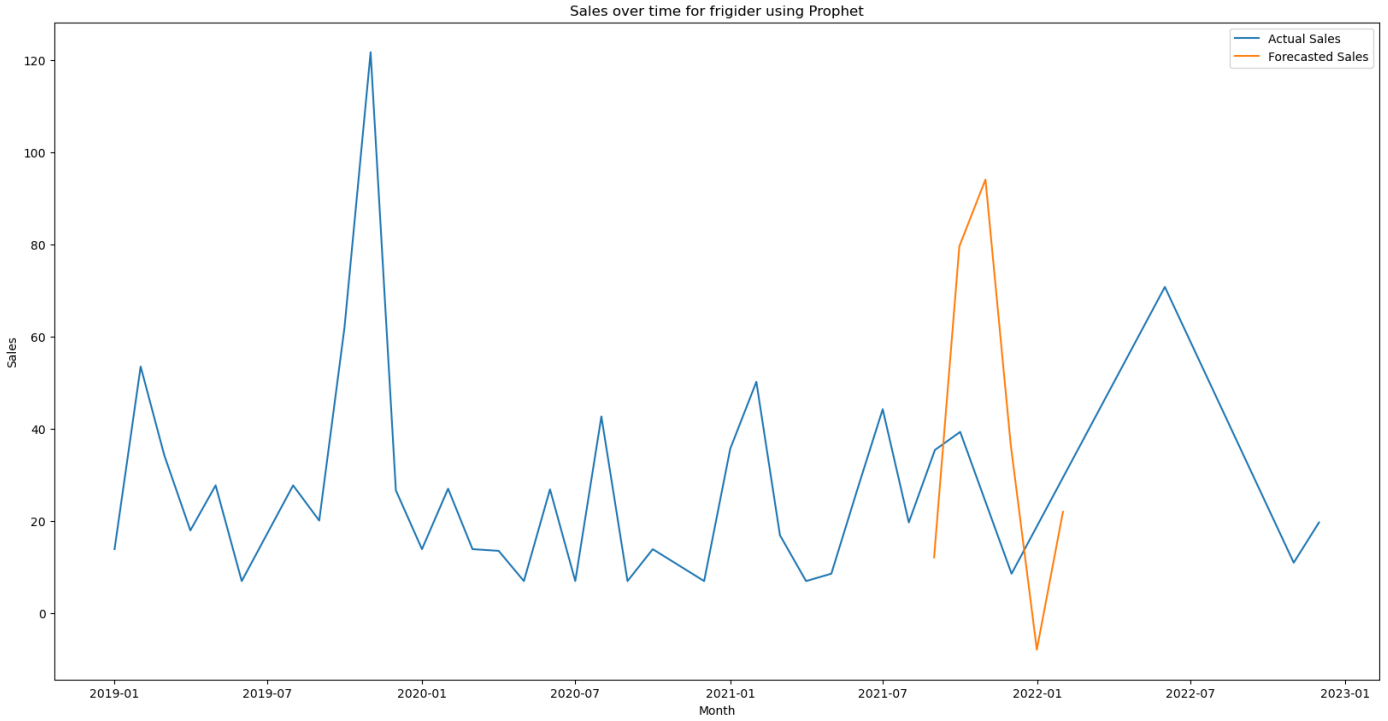
PR - frigider

In [62]:

```
# create a copy of the dataframe with a column named 'ds' with the index values
filter_df_sales_prophet = filter_df_sales.reset_index().rename(columns={'month_id': 'ds'
filter_df_sales_prophet = filter_df_sales_prophet.rename(columns={'sales':'y'})

train_data = filter_df_sales_prophet[:-6]
test_data = filter_df_sales_prophet[-6:]

model = Prophet()

# Fit the model on the data
model.fit(train_data)

# Create a dataframe to hold the forecast
future_data = model.make_future_dataframe(periods=6, freq='m')

# Make predictions
forecast = model.predict(future_data)
forecast = forecast.tail(6)
```

```
# Calculate error
pr_mae = mean_absolute_error(test_data['y'], forecast[-6:]['yhat'])
pr_mse = mean_squared_error(test_data['y'], forecast[-6:]['yhat'])
pr_rmse = np.sqrt(pr_mse)
print("Mean Absolute Error:", pr_mae)
print("Mean Squared Error:", pr_mse)
print("Root Mean Squared Error:", pr_rmse)

# Plot the forecast
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index, filter_df_sales['sales'], label='Actual Sales')

# Plot the predicted values for the last six months
plt.plot(forecast.ds, forecast.yhat, label='Predicted Sales')

plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Prophet')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
21:35:13 - cmdstanpy - INFO - Chain [1] start processing
21:35:13 - cmdstanpy - INFO - Chain [1] done processing
Mean Absolute Error: 34.0755146730762
Mean Squared Error: 1834.1035951063259
Root Mean Squared Error: 42.826435703970574
```



In [63]:

```
# Add the errors from all methods to the DataFrame
new_row_df =  pd.DataFrame({'product': [product], 'LR-MAE': [lr_mae], 'LR-MSE': [lr_mse]
errors_df = pd.concat([errors_df, new_row_df], ignore_index=True)
print(errors_df)
```

```
        product      LR-MAE         LR-MSE     LR-RMSE      RF-MAE  \
0     apa_plata  887.015829  988186.848307  994.075877  377.426267
1  apa_minerala   44.324535    2453.366899   49.531474   10.533100
2          bere  283.109792   95844.002658  309.586826   65.444150
3         cafea  287.818261   94385.213013  307.221765  134.035500
4       carnati  264.936590   99377.947534  315.242680  155.934000
5      frigider   13.054626     245.441803   15.666582    8.523900

          RF-MSE     RF-RMSE       PR-MAE         PR-MSE      PR-RMSE
```

0	200759.770117	448.062239	1107.910649	1.661801e+06	1289.108561
1	153.080170	12.372557	1703.258881	1.172504e+07	3424.185038
2	6221.823280	78.878535	437.297885	2.623325e+05	512.184061
3	22921.110989	151.397196	554.567175	3.976641e+05	630.606141
4	33523.190407	183.093393	370.439785	1.702760e+05	412.645094
5	105.584959	10.275454	34.075515	1.834104e+03	42.826436

Lapte DataFrame

In [64]:
```
# Filter the initial DataFrame
product = 'lapte'
filter_df = df.loc[df['denumire_produs'] == product]
filter_df = filter_df.drop(columns=['colli', 'store_county'])
#print(apa_plata_df)

# Group by month_id and aggregate the other columns
filter_df_sales = filter_df.groupby("month_id").agg({'sales': 'sum','Inflatie anuala': '
filter_df_sales = filter_df_sales.sort_values(by='month_id', ascending=True)

# Print the apa_plata_sales dataframe
print(filter_df_sales)
```

month_id	sales	Inflatie anuala	unnemployment	\
2019-01-01	5339.90	3.8	1.42	
2019-02-01	5734.66	3.8	3.32	
2019-03-01	5106.57	3.8	3.31	
2019-04-01	5862.41	3.8	3.19	
2019-05-01	4498.25	3.8	3.00	
2019-06-01	5579.20	3.8	2.95	
2019-07-01	5223.45	3.8	2.95	
2019-08-01	5751.19	3.8	3.01	
2019-09-01	5388.31	3.8	3.03	
2019-10-01	5697.71	3.8	3.00	
2019-11-01	5593.19	3.8	2.98	
2019-12-01	6550.13	3.8	2.98	
2020-01-01	4532.37	2.6	2.97	
2020-02-01	5420.25	2.6	2.98	
2020-03-01	6329.42	2.6	2.95	
2020-04-01	5663.01	2.6	2.88	
2020-05-01	5085.59	2.6	2.90	
2020-06-01	5219.27	2.6	2.87	
2020-07-01	5204.04	2.6	3.00	
2020-08-01	5924.29	2.6	3.02	
2020-09-01	4475.41	2.6	3.30	
2020-10-01	5586.64	2.6	3.26	
2020-11-01	5458.23	2.6	3.27	
2020-12-01	6550.81	2.6	3.32	
2021-01-01	6151.31	5.1	3.38	
2021-02-01	5474.41	5.1	3.34	
2021-03-01	5775.29	5.1	3.35	
2021-04-01	6284.54	5.1	3.33	
2021-05-01	6109.57	5.1	3.16	
2021-06-01	5899.03	5.1	3.06	
2021-07-01	6422.77	5.1	3.00	
2021-08-01	6887.75	5.1	2.96	
2021-09-01	5517.82	5.1	2.93	
2021-10-01	5604.55	5.1	2.85	
2021-11-01	6142.85	5.1	2.76	
2021-12-01	7849.76	5.1	2.72	
2022-01-01	6326.73	13.8	2.69	
2022-02-01	6222.73	13.8	2.68	
2022-03-01	7627.41	13.8	2.67	
2022-04-01	6642.49	13.8	2.64	

2022-05-01	5652.24	13.8	2.57
2022-06-01	5861.33	13.8	2.55
2022-07-01	5754.60	13.8	2.55
2022-08-01	5778.00	13.8	2.56
2022-09-01	5501.77	13.8	2.56
2022-10-01	5580.82	13.8	2.88
2022-11-01	5647.67	13.8	2.96
2022-12-01	6546.15	13.8	3.04

month_id	Inflatie(de la o luna la alta)	net_average_salary
2019-01-01	0.83	2936.0
2019-02-01	0.79	2933.0
2019-03-01	0.49	3075.0
2019-04-01	0.61	3115.0
2019-05-01	0.46	3101.0
2019-06-01	-0.23	3142.0
2019-07-01	-0.20	3119.0
2019-08-01	0.06	3044.0
2019-09-01	0.09	3082.0
2019-10-01	0.43	3116.0
2019-11-01	0.23	3179.0
2019-12-01	0.42	3340.0
2020-01-01	0.41	3189.0
2020-02-01	0.25	3202.0
2020-03-01	0.50	3294.0
2020-04-01	0.26	3182.0
2020-05-01	0.05	3179.0
2020-06-01	0.08	3298.0
2020-07-01	0.00	3372.0
2020-08-01	-0.05	3275.0
2020-09-01	-0.14	3321.0
2020-10-01	0.22	3343.0
2020-11-01	0.13	3411.0
2020-12-01	0.34	3620.0
2021-01-01	1.33	3395.0
2021-02-01	0.41	3365.0
2021-03-01	0.38	3547.0
2021-04-01	0.45	3561.0
2021-05-01	0.53	3492.0
2021-06-01	0.27	3541.0
2021-07-01	0.97	3545.0
2021-08-01	0.24	3487.0
2021-09-01	0.84	3517.0
2021-10-01	1.78	3544.0
2021-11-01	0.00	3645.0
2021-12-01	0.71	3879.0
2022-01-01	1.48	3698.0
2022-02-01	0.58	3721.0
2022-03-01	1.88	3937.0
2022-04-01	3.74	3967.0
2022-05-01	1.18	3928.0
2022-06-01	0.76	3977.0
2022-07-01	0.89	3975.0
2022-08-01	0.56	3933.0
2022-09-01	1.33	4003.0
2022-10-01	1.28	4008.0
2022-11-01	1.25	4141.0
2022-12-01	0.37	4141.0

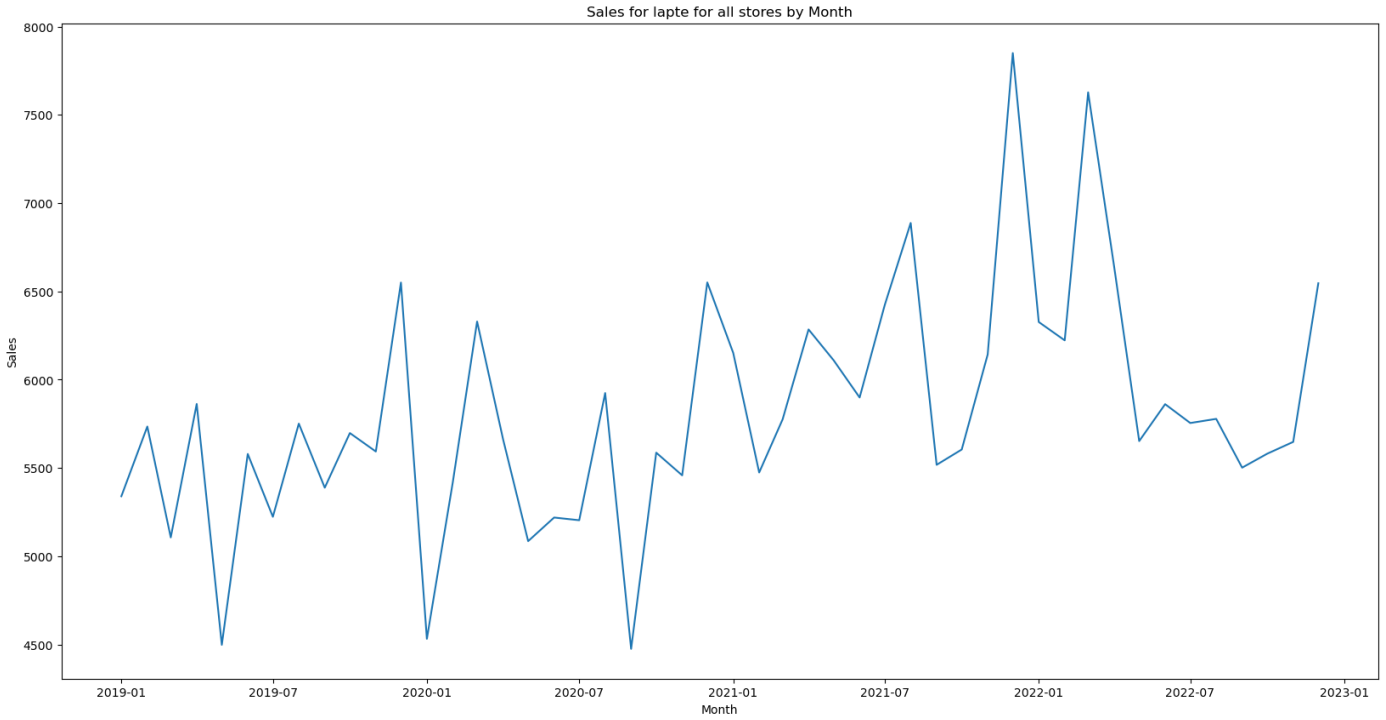
month_id	IPC Marfuri alimentare (%)	IPC Marfuri nealimentare (%)
2019-01-01	101.30	100.63
2019-02-01	101.27	100.57
2019-03-01	100.48	100.54
2019-04-01	100.68	100.53

2019-05-01	100.79	100.20
2019-06-01	99.53	99.76
2019-07-01	99.34	99.98
2019-08-01	99.71	100.22
2019-09-01	99.91	100.13
2019-10-01	100.70	100.32
2019-11-01	100.43	100.12
2019-12-01	100.84	100.28
2020-01-01	100.99	100.02
2020-02-01	100.63	99.94
2020-03-01	101.46	99.91
2020-04-01	101.27	99.67
2020-05-01	100.34	99.82
2020-06-01	99.62	100.28
2020-07-01	99.55	100.19
2020-08-01	99.59	100.08
2020-09-01	99.45	99.99
2020-10-01	100.11	100.31
2020-11-01	99.92	100.29
2020-12-01	100.29	100.51
2021-01-01	100.63	102.24
2021-02-01	100.46	100.47
2021-03-01	100.37	100.46
2021-04-01	100.45	100.47
2021-05-01	101.10	100.28
2021-06-01	100.25	100.29
2021-07-01	99.71	102.02
2021-08-01	99.95	100.34
2021-09-01	100.95	100.73
2021-10-01	101.06	102.78
2021-11-01	100.73	99.47
2021-12-01	100.84	100.74
2022-01-01	101.15	101.73
2022-02-01	101.96	99.70
2022-03-01	102.54	101.86
2022-04-01	102.56	105.45
2022-05-01	101.73	101.00
2022-06-01	100.62	100.92
2022-07-01	100.92	100.87
2022-08-01	101.82	99.81
2022-09-01	101.72	101.28
2022-10-01	102.29	100.80
2022-11-01	101.54	101.03
2022-12-01	101.26	99.68

month_id	IPC Servicii (%)
2019-01-01	100.57
2019-02-01	100.55
2019-03-01	100.40
2019-04-01	100.72
2019-05-01	100.55
2019-06-01	100.17
2019-07-01	100.10
2019-08-01	100.25
2019-09-01	100.27
2019-10-01	100.25
2019-11-01	100.15
2019-12-01	100.12
2020-01-01	100.43
2020-02-01	100.39
2020-03-01	100.35
2020-04-01	100.00
2020-05-01	100.11
2020-06-01	100.34
2020-07-01	100.31

```
2020-08-01              100.21
2020-09-01              100.20
2020-10-01              100.22
2020-11-01              100.07
2020-12-01              100.04
2021-01-01              100.25
2021-02-01              100.20
2021-03-01              100.24
2021-04-01              100.40
2021-05-01              100.27
2021-06-01              100.23
2021-07-01              100.39
2021-08-01              100.43
2021-09-01              100.96
2021-10-01              100.42
2021-11-01              100.20
2021-12-01              100.42
2022-01-01              101.37
2022-02-01              100.60
2022-03-01              100.67
2022-04-01              100.94
2022-05-01              100.61
2022-06-01              100.54
2022-07-01              100.88
2022-08-01              100.37
2022-09-01              100.72
2022-10-01              100.70
2022-11-01              101.31
2022-12-01              100.67
```

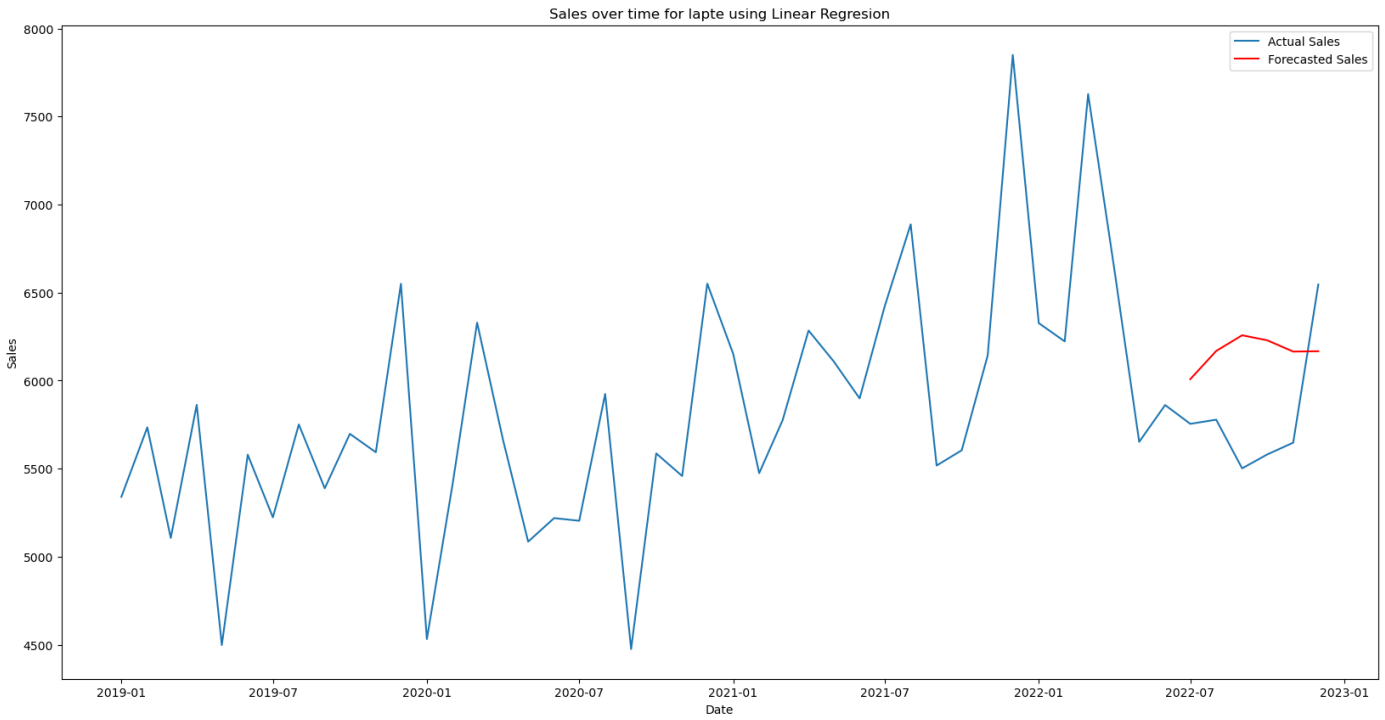
In [65]:
```
# Line graph for apa_plata for all stores
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index.get_level_values(0), filter_df_sales['sales'])
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales for ' + product + ' for all stores by Month')
plt.show()
```



LR - lapte

In [66]:
```
# Get the last 6 months of data
last_6_months = filter_df_sales.tail(6)

# Split the data into train and test sets
X_train = filter_df_sales.drop('sales', axis=1)
y_train = filter_df_sales['sales']
X_test = last_6_months.drop('sales', axis=1)
y_test = last_6_months['sales']

# Create a linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the error
lr_mae = mean_absolute_error(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)

print("Mean Absolute Error: ", lr_mae)
print("Mean Squared Error: ", lr_mse)
print("Root Mean Squared Error:", lr_rmse)

all_sales = filter_df_sales['sales']

# Get the index of the last 6 months of data
last_6_months_index = last_6_months.index

# Plot the entire period of sales data
plt.figure(figsize=(20,10))
plt.plot(all_sales)

# Make predictions on the last 6 months of data
y_pred = model.predict(X_test)

# Plot the forecasted sales on the same chart
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Linear Regresion')
plt.legend(['Actual Sales', 'Forecasted Sales'])

plt.show()
```

```
Mean Absolute Error:  490.76916398355723
Mean Squared Error:  269944.5578677117
Root Mean Squared Error: 519.561890315015
```

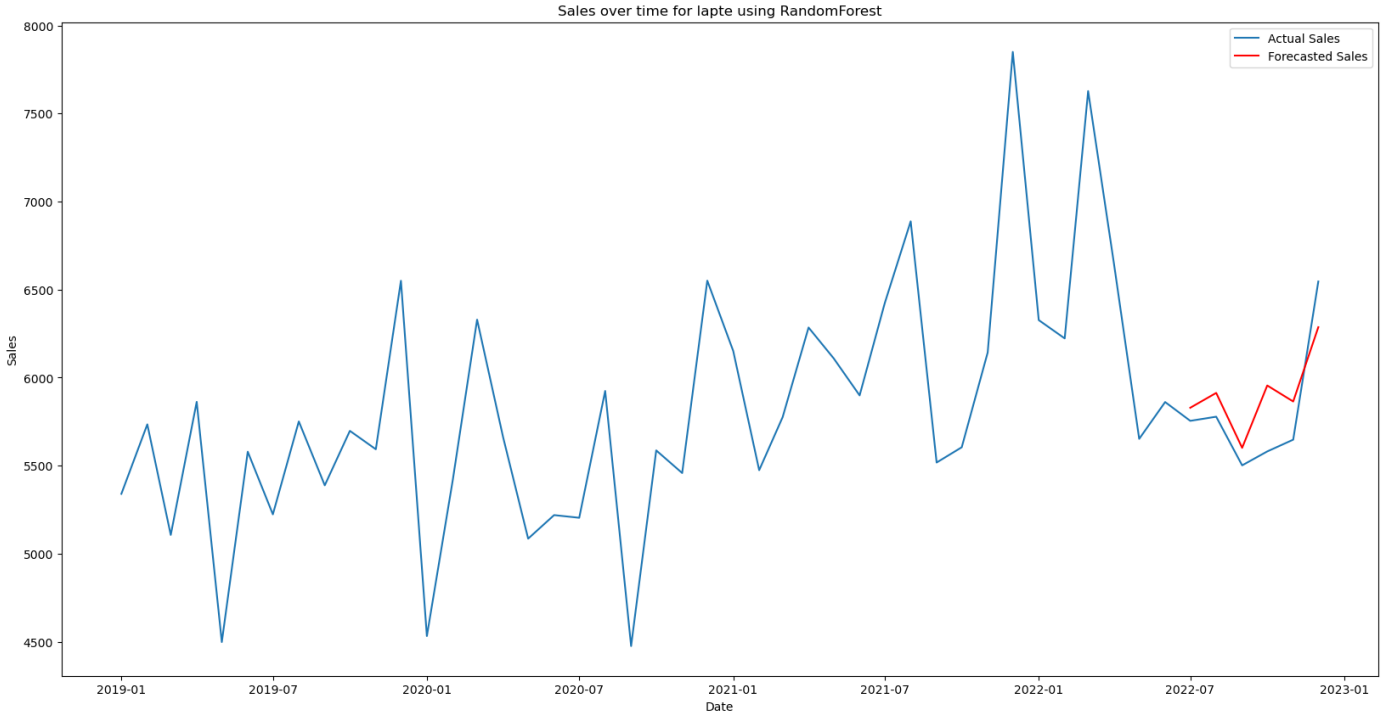


RF - lapte

In [67]:
```
last_6_months = filter_df_sales.tail(6)

X_train = filter_df_sales.drop(columns=["sales"])
y_train = filter_df_sales["sales"]
X_test = last_6_months.drop(columns=["sales"])
y_test = last_6_months["sales"]

# Create a random forest model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Use the model to make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate the model's accuracy
accuracy = rf.score(X_test, y_test)
print("Accuracy: ", accuracy)

# Calculate error
rf_mae = mean_absolute_error(y_test, y_pred)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rf_mse)

print("Mean Absolute Error: ", rf_mae)
print("Mean Squared Error: ", rf_mse)
print("Root Mean Squared Error:", rf_rmse)

# Line graph for the forcast
plt.figure(figsize=(20,10))

plt.plot(all_sales)
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using RandomForest')
```

```
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
Accuracy:  0.5991833430430953
Mean Absolute Error:  193.30963333333148
Mean Squared Error:  48061.717357505986
Root Mean Squared Error: 219.22982770942914
```



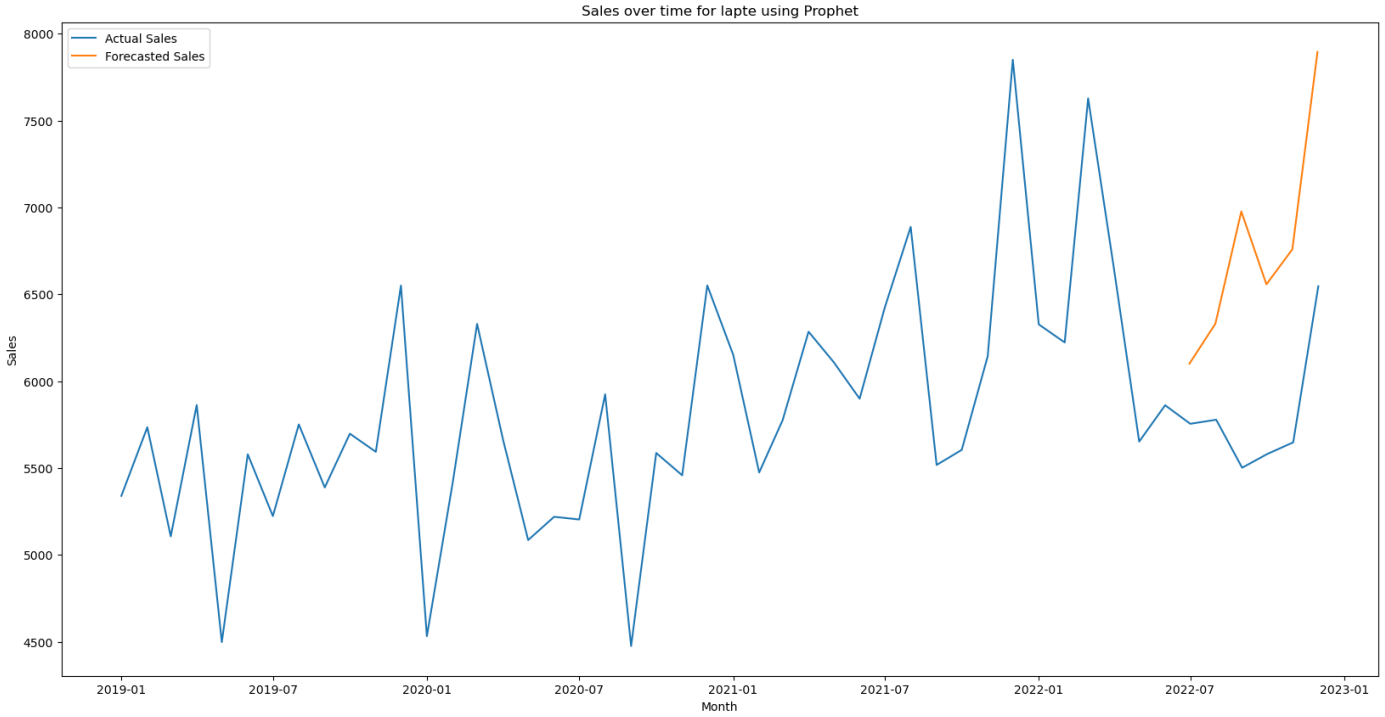
PR - lapte

In [68]:

```
# create a copy of the dataframe with a column named 'ds' with the index values
filter_df_sales_prophet = filter_df_sales.reset_index().rename(columns={'month_id': 'ds'
filter_df_sales_prophet = filter_df_sales_prophet.rename(columns={'sales':'y'})

train_data = filter_df_sales_prophet[:-6]
test_data = filter_df_sales_prophet[-6:]

model = Prophet()

# Fit the model on the data
model.fit(train_data)

# Create a dataframe to hold the forecast
future_data = model.make_future_dataframe(periods=6, freq='m')

# Make predictions
forecast = model.predict(future_data)
forecast = forecast.tail(6)


# Calculate error
pr_mae = mean_absolute_error(test_data['y'], forecast[-6:]['yhat'])
pr_mse = mean_squared_error(test_data['y'], forecast[-6:]['yhat'])
pr_rmse = np.sqrt(pr_mse)
print("Mean Absolute Error:", pr_mae)
print("Mean Squared Error:", pr_mse)
print("Root Mean Squared Error:", pr_rmse)

# Plot the forecast
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index, filter_df_sales['sales'], label='Actual Sales')
```

```
# Plot the predicted values for the last six months
plt.plot(forecast.ds, forecast.yhat, label='Predicted Sales')

plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Prophet')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
21:35:14 - cmdstanpy - INFO - Chain [1] start processing
21:35:15 - cmdstanpy - INFO - Chain [1] done processing
Mean Absolute Error: 968.1818397915512
Mean Squared Error: 1101084.5178032739
Root Mean Squared Error: 1049.3257443726775
```



In [69]:

```
# Add the errors from all methods to the DataFrame
new_row_df =  pd.DataFrame({'product': [product], 'LR-MAE': [lr_mae], 'LR-MSE': [lr_mse]
errors_df = pd.concat([errors_df, new_row_df], ignore_index=True)
print(errors_df)
```

```
        product      LR-MAE         LR-MSE     LR-RMSE      RF-MAE  \
0     apa_plata  887.015829  988186.848307  994.075877  377.426267
1  apa_minerala   44.324535    2453.366899   49.531474   10.533100
2          bere  283.109792   95844.002658  309.586826   65.444150
3         cafea  287.818261   94385.213013  307.221765  134.035500
4       carnati  264.936590   99377.947534  315.242680  155.934000
5      frigider   13.054626     245.441803   15.666582    8.523900
6         lapte  490.769164  269944.557868  519.561890  193.309633

           RF-MSE     RF-RMSE       PR-MAE        PR-MSE      PR-RMSE
0  200759.770117  448.062239  1107.910649  1.661801e+06  1289.108561
1     153.080170   12.372557  1703.258881  1.172504e+07  3424.185038
2    6221.823280   78.878535   437.297885  2.623325e+05   512.184061
3   22921.110989  151.397196   554.567175  3.976641e+05   630.606141
4   33523.190407  183.093393   370.439785  1.702760e+05   412.645094
5     105.584959   10.275454    34.075515  1.834104e+03    42.826436
6   48061.717358  219.229828   968.181840  1.101085e+06  1049.325744
```

Legume DataFrame

In [70]:
```
# Filter the initial DataFrame
product = 'legume'
filter_df = df.loc[df['denumire_produs'] == product]
filter_df = filter_df.drop(columns=['colli', 'store_county'])
#print(apa_plata_df)

# Group by month_id and aggregate the other columns
filter_df_sales = filter_df.groupby("month_id").agg({'sales': 'sum','Inflatie anuala': '
filter_df_sales = filter_df_sales.sort_values(by='month_id', ascending=True)

# Print the apa_plata_sales dataframe
print(filter_df_sales)
```

month_id	sales	Inflatie anuala	unnemployment	\
2019-01-01	1802.61	3.8	1.42	
2019-02-01	2377.92	3.8	3.32	
2019-03-01	2387.53	3.8	3.31	
2019-04-01	2412.19	3.8	3.19	
2019-05-01	2598.18	3.8	3.00	
2019-06-01	2114.99	3.8	2.95	
2019-07-01	2783.45	3.8	2.95	
2019-08-01	2426.90	3.8	3.01	
2019-09-01	2067.41	3.8	3.03	
2019-10-01	2544.82	3.8	3.00	
2019-11-01	2660.63	3.8	2.98	
2019-12-01	1985.84	3.8	2.98	
2020-01-01	2220.37	2.6	2.97	
2020-02-01	2611.80	2.6	2.98	
2020-03-01	3714.89	2.6	2.95	
2020-04-01	2198.82	2.6	2.88	
2020-05-01	2079.80	2.6	2.90	
2020-06-01	2121.61	2.6	2.87	
2020-07-01	1941.70	2.6	3.00	
2020-08-01	3198.25	2.6	3.02	
2020-09-01	2904.42	2.6	3.30	
2020-10-01	2409.46	2.6	3.26	
2020-11-01	2831.35	2.6	3.27	
2020-12-01	2885.13	2.6	3.32	
2021-01-01	2335.98	5.1	3.38	
2021-02-01	2172.77	5.1	3.34	
2021-03-01	3647.20	5.1	3.35	
2021-04-01	2920.70	5.1	3.33	
2021-05-01	2935.86	5.1	3.16	
2021-06-01	2242.22	5.1	3.06	
2021-07-01	2044.64	5.1	3.00	
2021-08-01	3235.97	5.1	2.96	
2021-09-01	3013.13	5.1	2.93	
2021-10-01	3953.80	5.1	2.85	
2021-11-01	2791.99	5.1	2.76	
2021-12-01	3265.94	5.1	2.72	
2022-01-01	2373.78	13.8	2.69	
2022-02-01	2814.53	13.8	2.68	
2022-03-01	4629.14	13.8	2.67	
2022-04-01	3329.30	13.8	2.64	
2022-05-01	2829.53	13.8	2.57	
2022-06-01	2487.96	13.8	2.55	
2022-07-01	3224.30	13.8	2.55	
2022-08-01	2707.70	13.8	2.56	
2022-09-01	2835.15	13.8	2.56	
2022-10-01	2979.84	13.8	2.88	
2022-11-01	3256.82	13.8	2.96	
2022-12-01	2209.25	13.8	3.04	

month_id	Inflatie(de la o luna la alta)	net_average_salary	\

month_id		
2019-01-01	0.83	2936.0
2019-02-01	0.79	2933.0
2019-03-01	0.49	3075.0
2019-04-01	0.61	3115.0
2019-05-01	0.46	3101.0
2019-06-01	-0.23	3142.0
2019-07-01	-0.20	3119.0
2019-08-01	0.06	3044.0
2019-09-01	0.09	3082.0
2019-10-01	0.43	3116.0
2019-11-01	0.23	3179.0
2019-12-01	0.42	3340.0
2020-01-01	0.41	3189.0
2020-02-01	0.25	3202.0
2020-03-01	0.50	3294.0
2020-04-01	0.26	3182.0
2020-05-01	0.05	3179.0
2020-06-01	0.08	3298.0
2020-07-01	0.00	3372.0
2020-08-01	-0.05	3275.0
2020-09-01	-0.14	3321.0
2020-10-01	0.22	3343.0
2020-11-01	0.13	3411.0
2020-12-01	0.34	3620.0
2021-01-01	1.33	3395.0
2021-02-01	0.41	3365.0
2021-03-01	0.38	3547.0
2021-04-01	0.45	3561.0
2021-05-01	0.53	3492.0
2021-06-01	0.27	3541.0
2021-07-01	0.97	3545.0
2021-08-01	0.24	3487.0
2021-09-01	0.84	3517.0
2021-10-01	1.78	3544.0
2021-11-01	0.00	3645.0
2021-12-01	0.71	3879.0
2022-01-01	1.48	3698.0
2022-02-01	0.58	3721.0
2022-03-01	1.88	3937.0
2022-04-01	3.74	3967.0
2022-05-01	1.18	3928.0
2022-06-01	0.76	3977.0
2022-07-01	0.89	3975.0
2022-08-01	0.56	3933.0
2022-09-01	1.33	4003.0
2022-10-01	1.28	4008.0
2022-11-01	1.25	4141.0
2022-12-01	0.37	4141.0

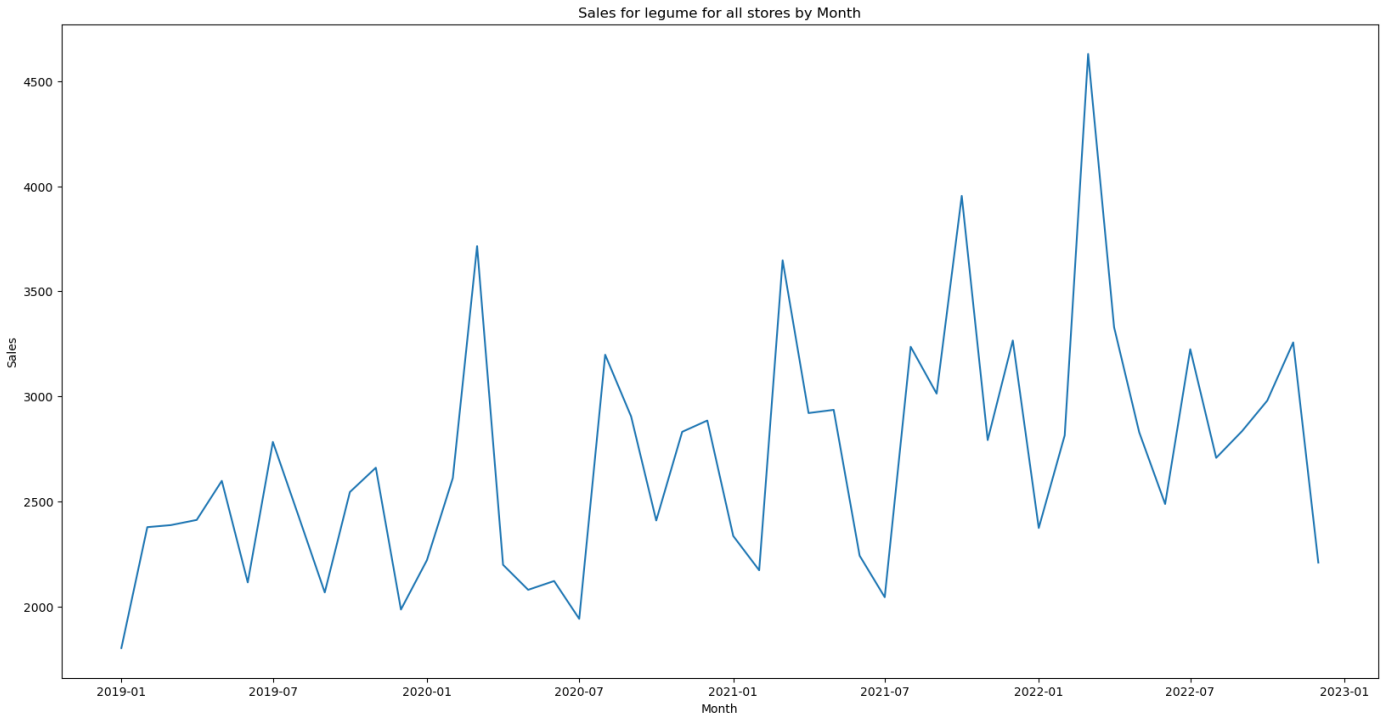
month_id	IPC Marfuri alimentare (%)	IPC Marfuri nealimentare (%)	\
2019-01-01	101.30	100.63	
2019-02-01	101.27	100.57	
2019-03-01	100.48	100.54	
2019-04-01	100.68	100.53	
2019-05-01	100.79	100.20	
2019-06-01	99.53	99.76	
2019-07-01	99.34	99.98	
2019-08-01	99.71	100.22	
2019-09-01	99.91	100.13	
2019-10-01	100.70	100.32	
2019-11-01	100.43	100.12	
2019-12-01	100.84	100.28	
2020-01-01	100.99	100.02	
2020-02-01	100.63	99.94	
2020-03-01	101.46	99.91	

2020-04-01	101.27	99.67
2020-05-01	100.34	99.82
2020-06-01	99.62	100.28
2020-07-01	99.55	100.19
2020-08-01	99.59	100.08
2020-09-01	99.45	99.99
2020-10-01	100.11	100.31
2020-11-01	99.92	100.29
2020-12-01	100.29	100.51
2021-01-01	100.63	102.24
2021-02-01	100.46	100.47
2021-03-01	100.37	100.46
2021-04-01	100.45	100.47
2021-05-01	101.10	100.28
2021-06-01	100.25	100.29
2021-07-01	99.71	102.02
2021-08-01	99.95	100.34
2021-09-01	100.95	100.73
2021-10-01	101.06	102.78
2021-11-01	100.73	99.47
2021-12-01	100.84	100.74
2022-01-01	101.15	101.73
2022-02-01	101.96	99.70
2022-03-01	102.54	101.86
2022-04-01	102.56	105.45
2022-05-01	101.73	101.00
2022-06-01	100.62	100.92
2022-07-01	100.92	100.87
2022-08-01	101.82	99.81
2022-09-01	101.72	101.28
2022-10-01	102.29	100.80
2022-11-01	101.54	101.03
2022-12-01	101.26	99.68

month_id	IPC Servicii (%)
2019-01-01	100.57
2019-02-01	100.55
2019-03-01	100.40
2019-04-01	100.72
2019-05-01	100.55
2019-06-01	100.17
2019-07-01	100.10
2019-08-01	100.25
2019-09-01	100.27
2019-10-01	100.25
2019-11-01	100.15
2019-12-01	100.12
2020-01-01	100.43
2020-02-01	100.39
2020-03-01	100.35
2020-04-01	100.00
2020-05-01	100.11
2020-06-01	100.34
2020-07-01	100.31
2020-08-01	100.21
2020-09-01	100.20
2020-10-01	100.22
2020-11-01	100.07
2020-12-01	100.04
2021-01-01	100.25
2021-02-01	100.20
2021-03-01	100.24
2021-04-01	100.40
2021-05-01	100.27
2021-06-01	100.23

```
2021-07-01            100.39
2021-08-01            100.43
2021-09-01            100.96
2021-10-01            100.42
2021-11-01            100.20
2021-12-01            100.42
2022-01-01            101.37
2022-02-01            100.60
2022-03-01            100.67
2022-04-01            100.94
2022-05-01            100.61
2022-06-01            100.54
2022-07-01            100.88
2022-08-01            100.37
2022-09-01            100.72
2022-10-01            100.70
2022-11-01            101.31
2022-12-01            100.67
```

In [71]:
```python
# Line graph for apa_plata for all stores
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index.get_level_values(0), filter_df_sales['sales'])
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales for ' + product + ' for all stores by Month')
plt.show()
```

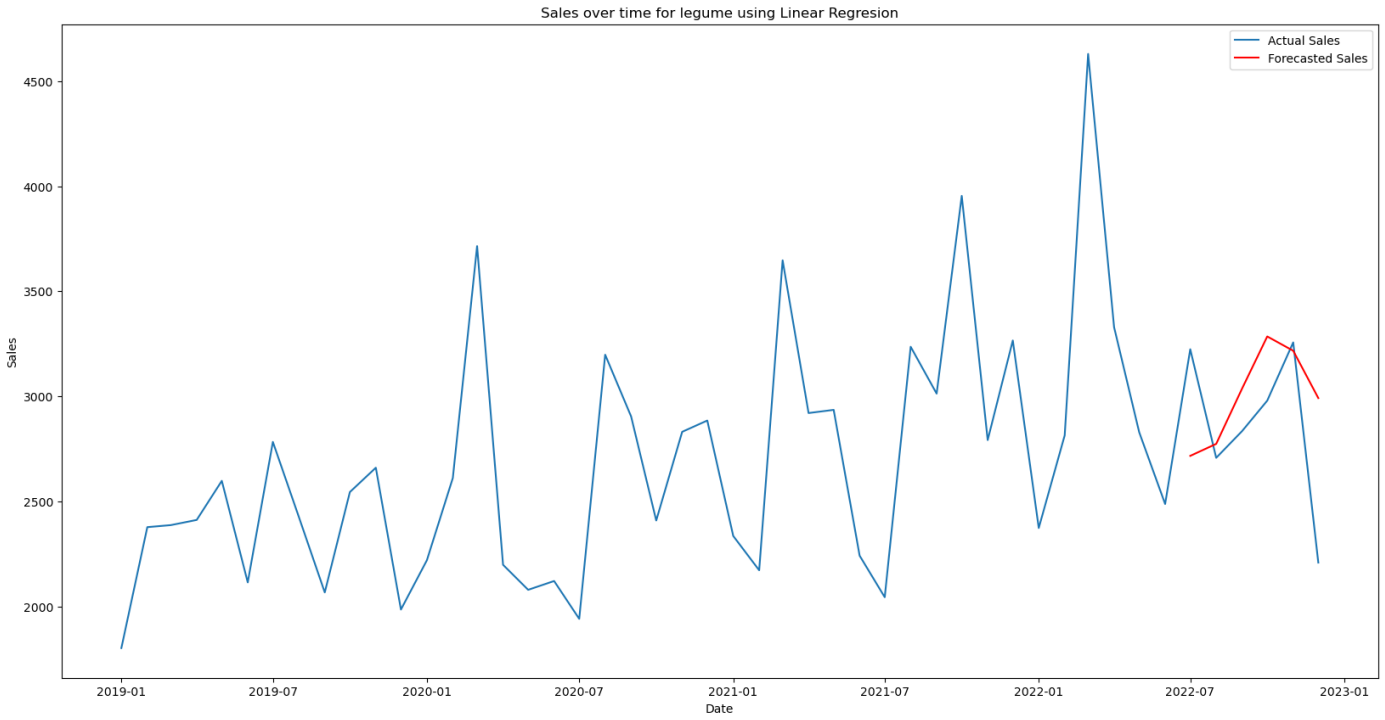


LR - legume

In [72]:
```python
# Get the last 6 months of data
last_6_months = filter_df_sales.tail(6)

# Split the data into train and test sets
X_train = filter_df_sales.drop('sales', axis=1)
y_train = filter_df_sales['sales']
X_test = last_6_months.drop('sales', axis=1)
y_test = last_6_months['sales']

# Create a linear regression model
model = LinearRegression()
```

```
# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the error
lr_mae = mean_absolute_error(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)

print("Mean Absolute Error: ", lr_mae)
print("Mean Squared Error: ", lr_mse)
print("Root Mean Squared Error:", lr_rmse)

all_sales = filter_df_sales['sales']

# Get the index of the last 6 months of data
last_6_months_index = last_6_months.index

# Plot the entire period of sales data
plt.figure(figsize=(20,10))
plt.plot(all_sales)

# Make predictions on the last 6 months of data
y_pred = model.predict(X_test)

# Plot the forecasted sales on the same chart
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Linear Regresion')
plt.legend(['Actual Sales', 'Forecasted Sales'])

plt.show()
```

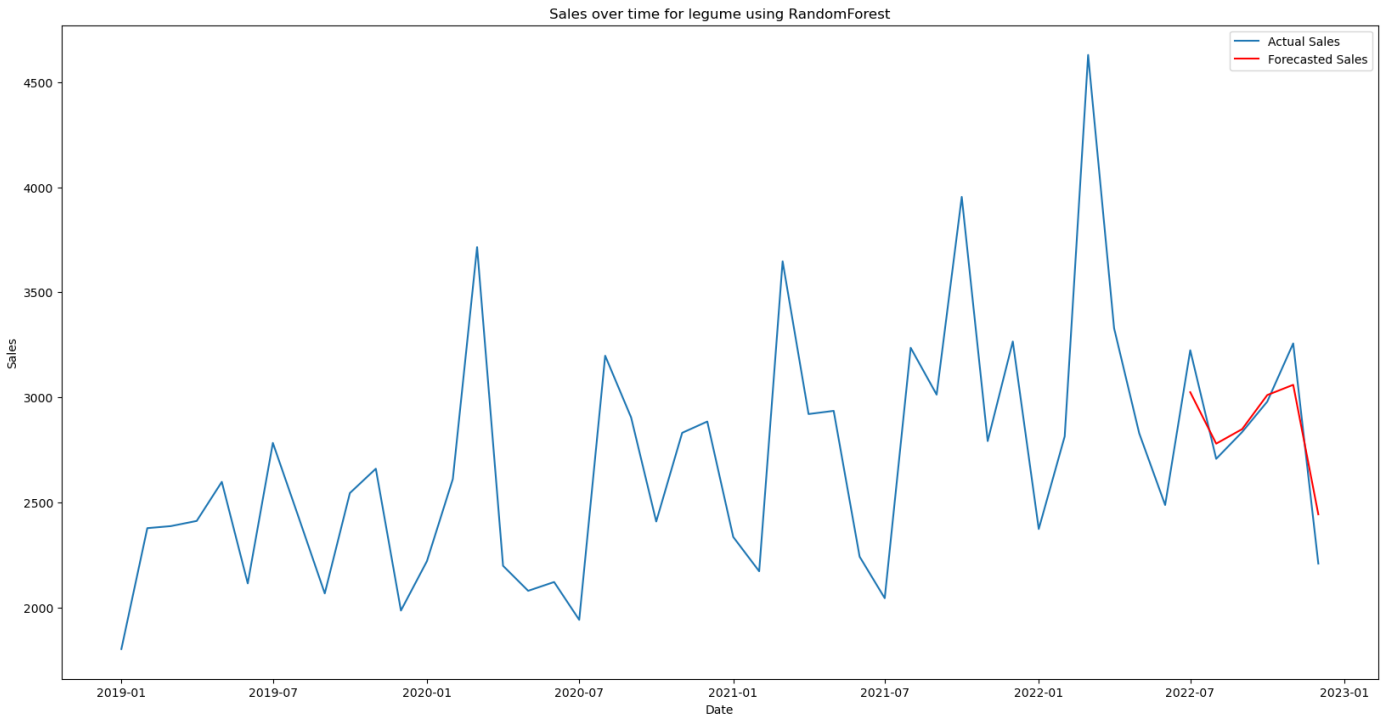
```
Mean Absolute Error:  317.3343167520664
Mean Squared Error:  168366.06832385538
Root Mean Squared Error: 410.3243452731697
```



RF - legume

In [73]:
```
last_6_months = filter_df_sales.tail(6)

X_train = filter_df_sales.drop(columns=["sales"])
y_train = filter_df_sales["sales"]
X_test = last_6_months.drop(columns=["sales"])
y_test = last_6_months["sales"]

# Create a random forest model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Use the model to make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate the model's accuracy
accuracy = rf.score(X_test, y_test)
print("Accuracy: ", accuracy)

# Calculate error
rf_mae = mean_absolute_error(y_test, y_pred)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rf_mse)

print("Mean Absolute Error: ", rf_mae)
print("Mean Squared Error: ", rf_mse)
print("Root Mean Squared Error:", rf_rmse)

# Line graph for the forcast
plt.figure(figsize=(20,10))

plt.plot(all_sales)
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using RandomForest')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
Accuracy:  0.8135762724080686
Mean Absolute Error:  124.7484166666668
Mean Squared Error:  23345.264738801623
Root Mean Squared Error: 152.7915728657887
```



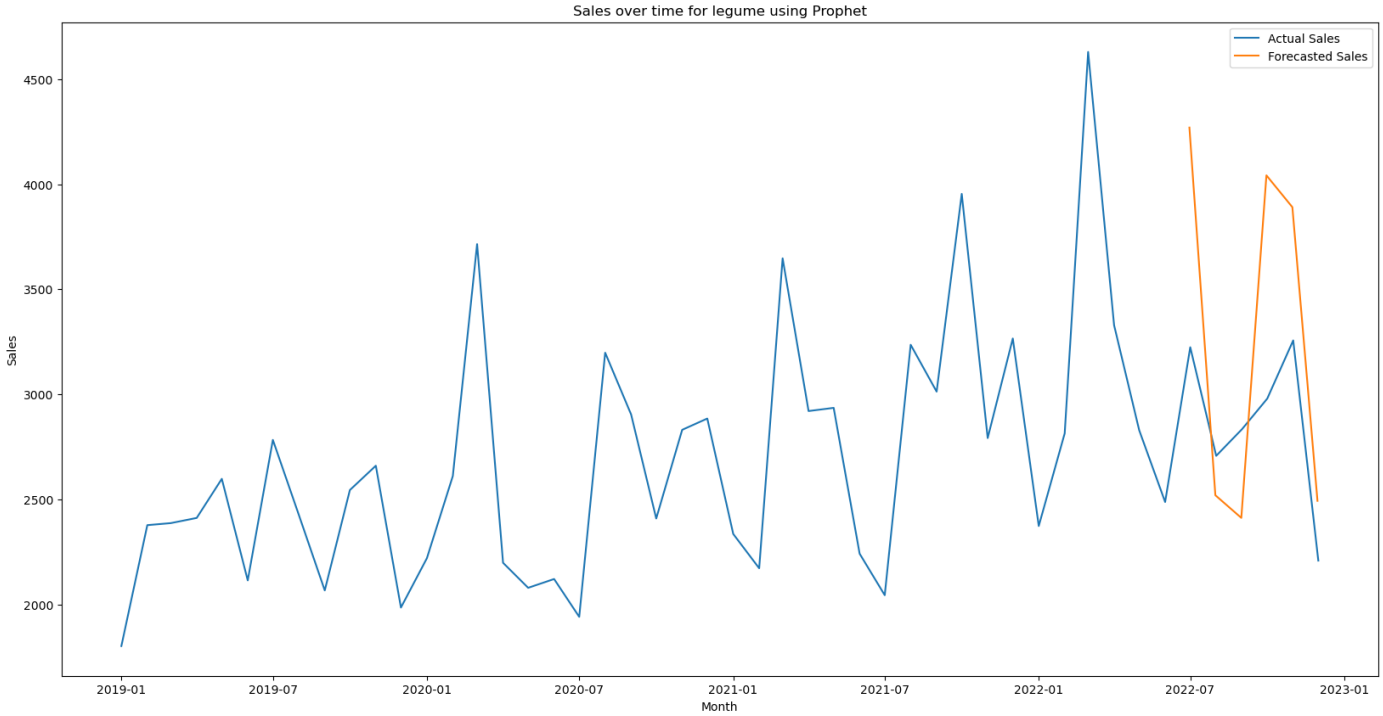
PR - legume

In [74]:

```
# create a copy of the dataframe with a column named 'ds' with the index values
filter_df_sales_prophet = filter_df_sales.reset_index().rename(columns={'month_id': 'ds'
filter_df_sales_prophet = filter_df_sales_prophet.rename(columns={'sales':'y'})

train_data = filter_df_sales_prophet[:-6]
test_data = filter_df_sales_prophet[-6:]

model = Prophet()

# Fit the model on the data
model.fit(train_data)

# Create a dataframe to hold the forecast
future_data = model.make_future_dataframe(periods=6, freq='m')

# Make predictions
forecast = model.predict(future_data)
forecast = forecast.tail(6)


# Calculate error
pr_mae = mean_absolute_error(test_data['y'], forecast[-6:]['yhat'])
pr_mse = mean_squared_error(test_data['y'], forecast[-6:]['yhat'])
pr_rmse = np.sqrt(pr_mse)
print("Mean Absolute Error:", pr_mae)
print("Mean Squared Error:", pr_mse)
print("Root Mean Squared Error:", pr_rmse)

# Plot the forecast
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index, filter_df_sales['sales'], label='Actual Sales')

# Plot the predicted values for the last six months
plt.plot(forecast.ds, forecast.yhat, label='Predicted Sales')

plt.xlabel('Month')
plt.ylabel('Sales')
```

```
plt.title('Sales over time for ' +product+ ' using Prophet')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
21:35:16 - cmdstanpy - INFO - Chain [1] start processing
21:35:16 - cmdstanpy - INFO - Chain [1] done processing
Mean Absolute Error: 605.9134602691287
Mean Squared Error: 486107.7457599249
Root Mean Squared Error: 697.2142753558084
```



In [75]:
```
# Add the errors from all methods to the DataFrame
new_row_df =  pd.DataFrame({'product': [product], 'LR-MAE': [lr_mae], 'LR-MSE': [lr_mse]
errors_df = pd.concat([errors_df, new_row_df], ignore_index=True)
print(errors_df)
```

```
        product      LR-MAE         LR-MSE     LR-RMSE      RF-MAE  \
0     apa_plata  887.015829  988186.848307  994.075877  377.426267
1  apa_minerala   44.324535    2453.366899   49.531474   10.533100
2          bere  283.109792   95844.002658  309.586826   65.444150
3         cafea  287.818261   94385.213013  307.221765  134.035500
4       carnati  264.936590   99377.947534  315.242680  155.934000
5      frigider   13.054626     245.441803   15.666582    8.523900
6         lapte  490.769164  269944.557868  519.561890  193.309633
7        legume  317.334317  168366.068324  410.324345  124.748417

           RF-MSE     RF-RMSE       PR-MAE        PR-MSE      PR-RMSE
0  200759.770117  448.062239  1107.910649  1.661801e+06  1289.108561
1     153.080170   12.372557  1703.258881  1.172504e+07  3424.185038
2    6221.823280   78.878535   437.297885  2.623325e+05   512.184061
3   22921.110989  151.397196   554.567175  3.976641e+05   630.606141
4   33523.190407  183.093393   370.439785  1.702760e+05   412.645094
5     105.584959   10.275454    34.075515  1.834104e+03    42.826436
6   48061.717358  219.229828   968.181840  1.101085e+06  1049.325744
7   23345.264739  152.791573   605.913460  4.861077e+05   697.214275
```

Masina de spalat DataFrame

In [76]:
```
# Filter the initial DataFrame
product = 'masina_de_spalat'
filter_df = df.loc[df['denumire_produs'] == product]
filter_df = filter_df.drop(columns=['colli', 'store_county'])
```

```
#print(apa_plata_df)

# Group by month_id and aggregate the other columns
filter_df_sales = filter_df.groupby("month_id").agg({'sales': 'sum','Inflatie anuala': '
filter_df_sales = filter_df_sales.sort_values(by='month_id', ascending=True)

# Print the apa_plata_sales dataframe
print(filter_df_sales)
```

month_id	sales	Inflatie anuala	unnemployment	Inflatie(de la o luna la alta)	net_average_salary
2019-01-01	14.84	3.8	1.42	0.83	2936.0
2019-03-01	14.84	3.8	3.31	0.49	3075.0
2019-04-01	29.68	3.8	3.19	0.61	3115.0
2019-05-01	51.94	3.8	3.00	0.46	3101.0
2019-06-01	44.52	3.8	2.95	-0.23	3142.0
2019-07-01	7.42	3.8	2.95	-0.20	3119.0
2019-08-01	14.84	3.8	3.01	0.06	3044.0
2019-11-01	7.42	3.8	2.98	0.23	3179.0
2019-12-01	7.42	3.8	2.98	0.42	3340.0
2020-01-01	7.42	2.6	2.97	0.41	3189.0
2020-05-01	37.10	2.6	2.90	0.05	3179.0
2020-07-01	29.68	2.6	3.00	0.00	3372.0
2020-08-01	7.42	2.6	3.02	-0.05	3275.0
2020-09-01	14.84	2.6	3.30	-0.14	3321.0
2020-12-01	7.42	2.6	3.32	0.34	3620.0
2021-01-01	7.42	5.1	3.38		
2021-03-01	7.42	5.1	3.35		
2021-04-01	14.84	5.1	3.33		
2021-05-01	32.36	5.1	3.16		
2021-06-01	23.61	5.1	3.06		
2021-07-01	8.75	5.1	3.00		
2021-08-01	36.30	5.1	2.96		
2021-09-01	37.79	5.1	2.93		
2021-10-01	37.79	5.1	2.85		
2021-11-01	46.54	5.1	2.76		
2021-12-01	61.40	5.1	2.72		
2022-01-01	17.50	13.8	2.69		
2022-02-01	61.71	13.8	2.68		
2022-03-01	112.80	13.8	2.67		
2022-04-01	99.65	13.8	2.64		
2022-05-01	30.08	13.8	2.57		
2022-06-01	36.24	13.8	2.55		
2022-07-01	51.63	13.8	2.55		
2022-08-01	36.24	13.8	2.56		
2022-09-01	48.20	13.8	2.56		
2022-10-01	93.32	13.8	2.88		
2022-12-01	116.28	13.8	3.04		

month_id		
2021-01-01	1.33	3395.0
2021-03-01	0.38	3547.0
2021-04-01	0.45	3561.0
2021-05-01	0.53	3492.0
2021-06-01	0.27	3541.0
2021-07-01	0.97	3545.0
2021-08-01	0.24	3487.0
2021-09-01	0.84	3517.0
2021-10-01	1.78	3544.0
2021-11-01	0.00	3645.0
2021-12-01	0.71	3879.0
2022-01-01	1.48	3698.0
2022-02-01	0.58	3721.0
2022-03-01	1.88	3937.0
2022-04-01	3.74	3967.0
2022-05-01	1.18	3928.0
2022-06-01	0.76	3977.0
2022-07-01	0.89	3975.0
2022-08-01	0.56	3933.0
2022-09-01	1.33	4003.0
2022-10-01	1.28	4008.0
2022-12-01	0.37	4141.0

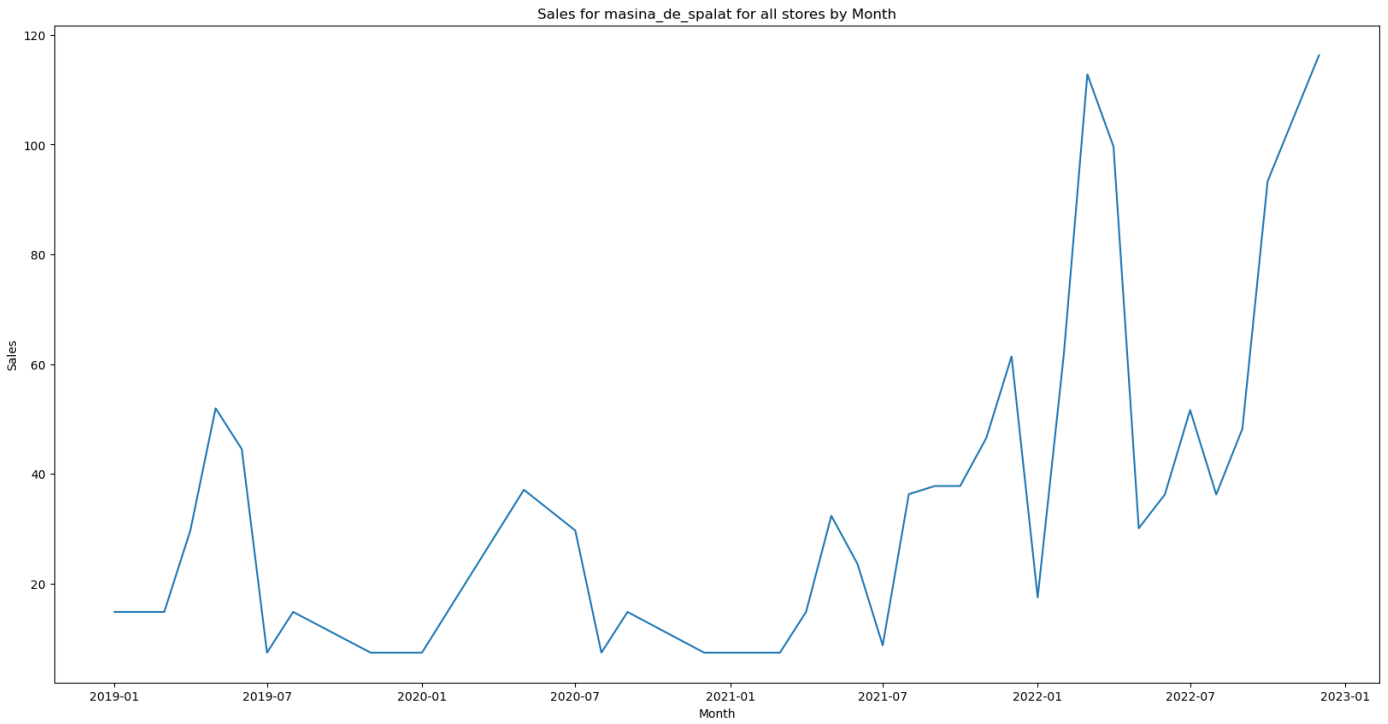
month_id	IPC Marfuri alimentare (%)	IPC Marfuri nealimentare (%)	\
2019-01-01	101.30	100.63	
2019-03-01	100.48	100.54	
2019-04-01	100.68	100.53	
2019-05-01	100.79	100.20	
2019-06-01	99.53	99.76	
2019-07-01	99.34	99.98	
2019-08-01	99.71	100.22	
2019-11-01	100.43	100.12	
2019-12-01	100.84	100.28	
2020-01-01	100.99	100.02	
2020-05-01	100.34	99.82	
2020-07-01	99.55	100.19	
2020-08-01	99.59	100.08	
2020-09-01	99.45	99.99	
2020-12-01	100.29	100.51	
2021-01-01	100.63	102.24	
2021-03-01	100.37	100.46	
2021-04-01	100.45	100.47	
2021-05-01	101.10	100.28	
2021-06-01	100.25	100.29	
2021-07-01	99.71	102.02	
2021-08-01	99.95	100.34	
2021-09-01	100.95	100.73	
2021-10-01	101.06	102.78	
2021-11-01	100.73	99.47	
2021-12-01	100.84	100.74	
2022-01-01	101.15	101.73	
2022-02-01	101.96	99.70	
2022-03-01	102.54	101.86	
2022-04-01	102.56	105.45	
2022-05-01	101.73	101.00	
2022-06-01	100.62	100.92	
2022-07-01	100.92	100.87	
2022-08-01	101.82	99.81	
2022-09-01	101.72	101.28	
2022-10-01	102.29	100.80	
2022-12-01	101.26	99.68	

month_id	IPC Servicii (%)
2019-01-01	100.57

```
2019-03-01            100.40
2019-04-01            100.72
2019-05-01            100.55
2019-06-01            100.17
2019-07-01            100.10
2019-08-01            100.25
2019-11-01            100.15
2019-12-01            100.12
2020-01-01            100.43
2020-05-01            100.11
2020-07-01            100.31
2020-08-01            100.21
2020-09-01            100.20
2020-12-01            100.04
2021-01-01            100.25
2021-03-01            100.24
2021-04-01            100.40
2021-05-01            100.27
2021-06-01            100.23
2021-07-01            100.39
2021-08-01            100.43
2021-09-01            100.96
2021-10-01            100.42
2021-11-01            100.20
2021-12-01            100.42
2022-01-01            101.37
2022-02-01            100.60
2022-03-01            100.67
2022-04-01            100.94
2022-05-01            100.61
2022-06-01            100.54
2022-07-01            100.88
2022-08-01            100.37
2022-09-01            100.72
2022-10-01            100.70
2022-12-01            100.67
```

In [77]:

```
# Line graph for apa_plata for all stores
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index.get_level_values(0), filter_df_sales['sales'])
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales for ' + product + ' for all stores by Month')
plt.show()
```



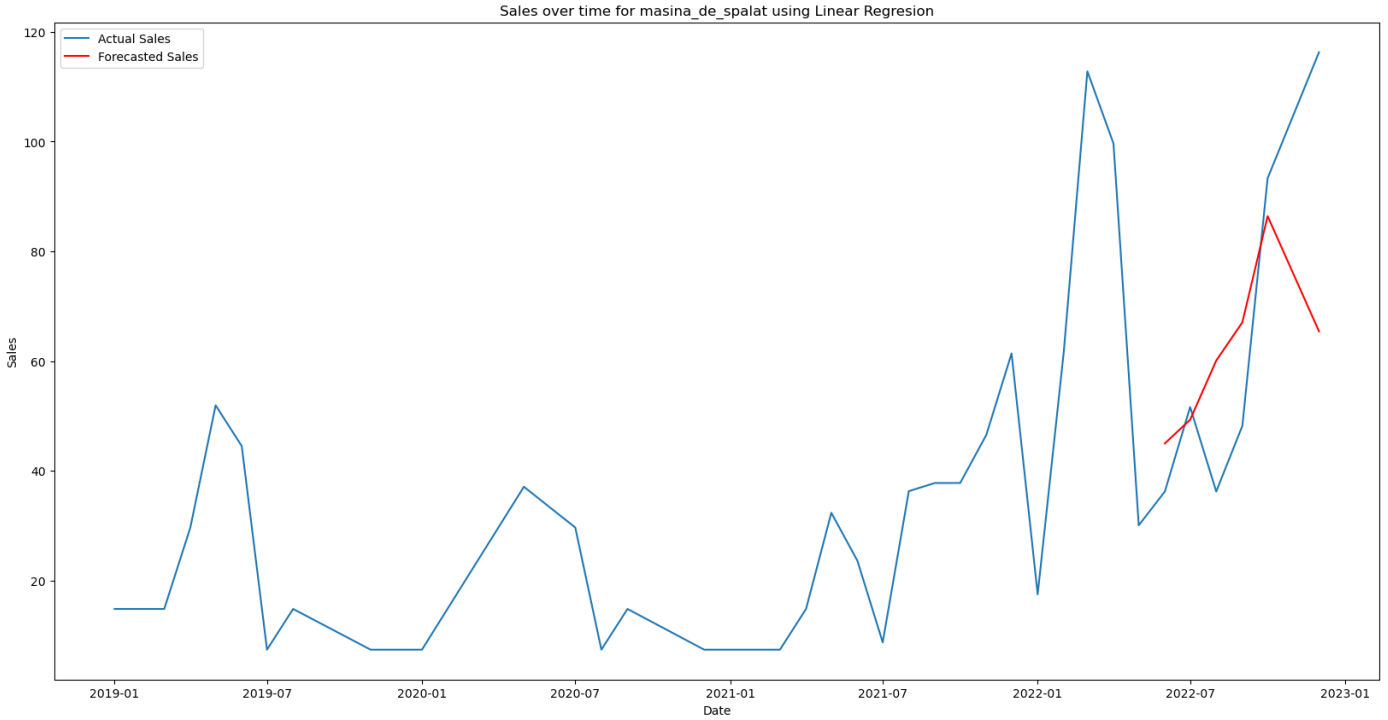
LR - masina_de_spalat

In [78]:

```
# Get the last 6 months of data
last_6_months = filter_df_sales.tail(6)

# Split the data into train and test sets
X_train = filter_df_sales.drop('sales', axis=1)
y_train = filter_df_sales['sales']
X_test = last_6_months.drop('sales', axis=1)
y_test = last_6_months['sales']

# Create a linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the error
lr_mae = mean_absolute_error(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)

print("Mean Absolute Error: ", lr_mae)
print("Mean Squared Error: ", lr_mse)
print("Root Mean Squared Error:", lr_rmse)

all_sales = filter_df_sales['sales']

# Get the index of the last 6 months of data
last_6_months_index = last_6_months.index

# Plot the entire period of sales data
plt.figure(figsize=(20,10))
plt.plot(all_sales)

# Make predictions on the last 6 months of data
```

```
y_pred = model.predict(X_test)

# Plot the forecasted sales on the same chart
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Linear Regresion')
plt.legend(['Actual Sales', 'Forecasted Sales'])

plt.show()
```

```
Mean Absolute Error:  18.592357252542083
Mean Squared Error:  606.8673255496291
Root Mean Squared Error: 24.63467729745265
```



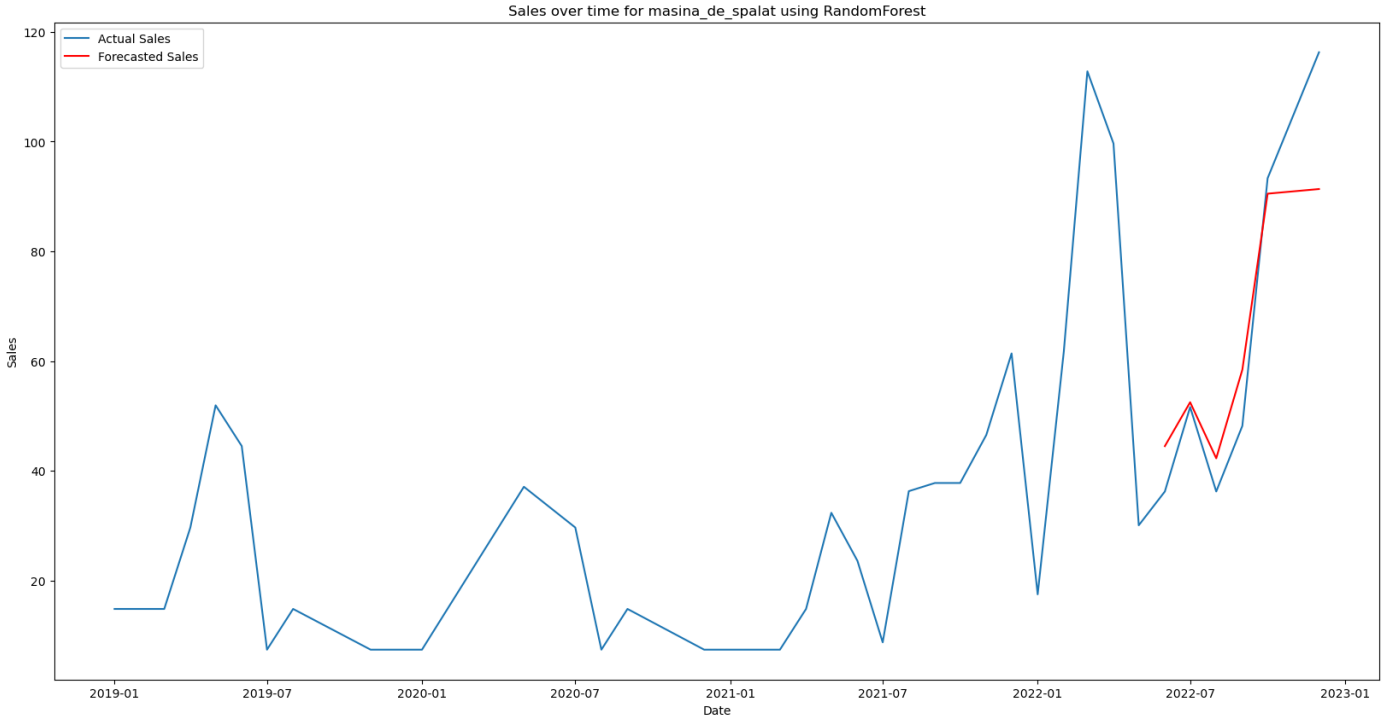
RF - masina_de_spalat

In [79]:

```
last_6_months = filter_df_sales.tail(6)

X_train = filter_df_sales.drop(columns=["sales"])
y_train = filter_df_sales["sales"]
X_test = last_6_months.drop(columns=["sales"])
y_test = last_6_months["sales"]

# Create a random forest model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Use the model to make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate the model's accuracy
accuracy = rf.score(X_test, y_test)
print("Accuracy: ", accuracy)

# Calculate error
rf_mae = mean_absolute_error(y_test, y_pred)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rf_mse)
```

```
print("Mean Absolute Error: ", rf_mae)
print("Mean Squared Error: ", rf_mse)
print("Root Mean Squared Error:", rf_rmse)

# Line graph for the forcast
plt.figure(figsize=(20,10))

plt.plot(all_sales)
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using RandomForest')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
Accuracy:  0.8482943039854413
Mean Absolute Error:  8.862466666666664
Mean Squared Error:  139.97441801333323
Root Mean Squared Error: 11.831078480566902
```



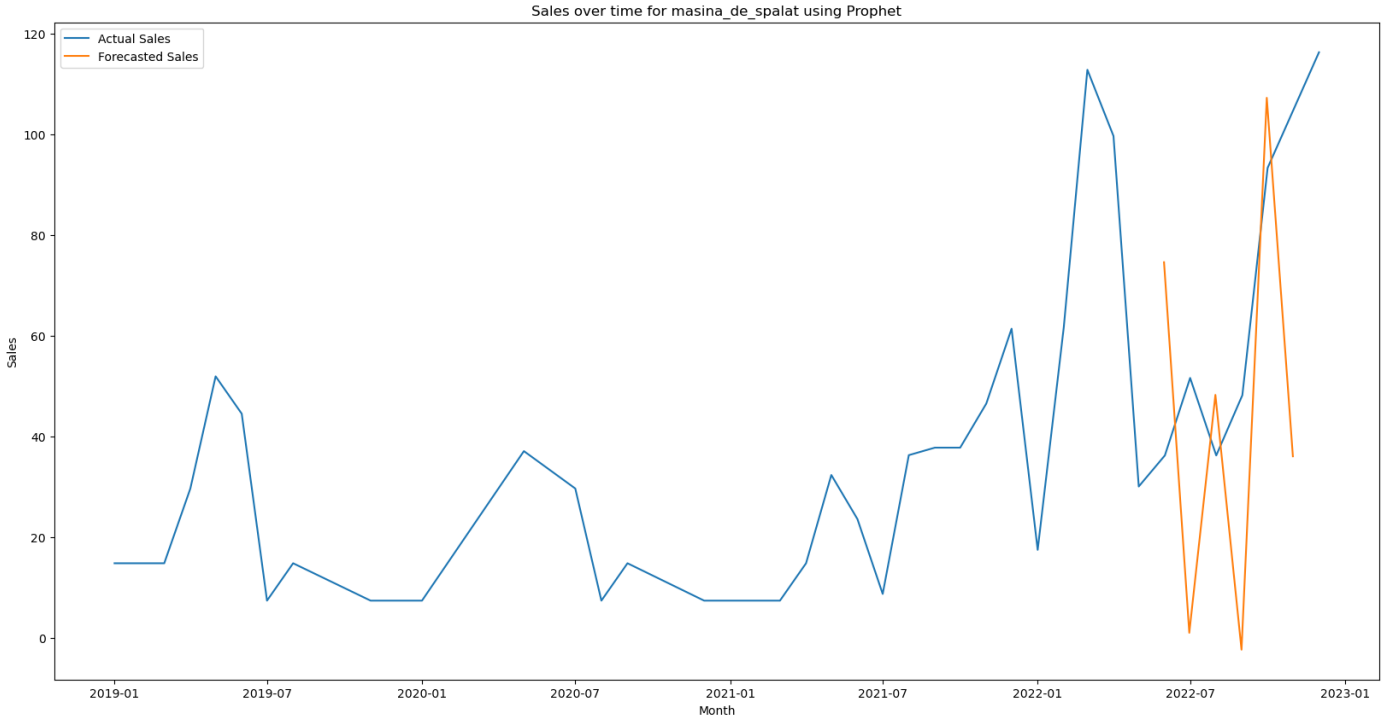
PR - masina_de_spalat

In [80]:

```
# create a copy of the dataframe with a column named 'ds' with the index values
filter_df_sales_prophet = filter_df_sales.reset_index().rename(columns={'month_id': 'ds'
filter_df_sales_prophet = filter_df_sales_prophet.rename(columns={'sales':'y'})

train_data = filter_df_sales_prophet[:-6]
test_data = filter_df_sales_prophet[-6:]

model = Prophet()

# Fit the model on the data
model.fit(train_data)

# Create a dataframe to hold the forecast
future_data = model.make_future_dataframe(periods=6, freq='m')

# Make predictions
forecast = model.predict(future_data)
forecast = forecast.tail(6)
```

```
# Calculate error
pr_mae = mean_absolute_error(test_data['y'], forecast[-6:]['yhat'])
pr_mse = mean_squared_error(test_data['y'], forecast[-6:]['yhat'])
pr_rmse = np.sqrt(pr_mse)
print("Mean Absolute Error:", pr_mae)
print("Mean Squared Error:", pr_mse)
print("Root Mean Squared Error:", pr_rmse)

# Plot the forecast
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index, filter_df_sales['sales'], label='Actual Sales')

# Plot the predicted values for the last six months
plt.plot(forecast.ds, forecast.yhat, label='Predicted Sales')

plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Prophet')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
21:35:18 - cmdstanpy - INFO - Chain [1] start processing
21:35:18 - cmdstanpy - INFO - Chain [1] done processing
Mean Absolute Error: 40.934713284676455
Mean Squared Error: 2225.6679616706197
Root Mean Squared Error: 47.176985508514846
```



In [81]:

```
# Add the errors from all methods to the DataFrame
new_row_df =  pd.DataFrame({'product': [product], 'LR-MAE': [lr_mae], 'LR-MSE': [lr_mse]
errors_df = pd.concat([errors_df, new_row_df], ignore_index=True)
print(errors_df)
```

	product	LR-MAE	LR-MSE	LR-RMSE	RF-MAE	\
0	apa_plata	887.015829	988186.848307	994.075877	377.426267	
1	apa_minerala	44.324535	2453.366899	49.531474	10.533100	
2	bere	283.109792	95844.002658	309.586826	65.444150	
3	cafea	287.818261	94385.213013	307.221765	134.035500	
4	carnati	264.936590	99377.947534	315.242680	155.934000	
5	frigider	13.054626	245.441803	15.666582	8.523900	
6	lapte	490.769164	269944.557868	519.561890	193.309633	
7	legume	317.334317	168366.068324	410.324345	124.748417	

```
8  masina_de_spalat   18.592357     606.867326   24.634677    8.862467
```

	RF-MSE	RF-RMSE	PR-MAE	PR-MSE	PR-RMSE
0	200759.770117	448.062239	1107.910649	1.661801e+06	1289.108561
1	153.080170	12.372557	1703.258881	1.172504e+07	3424.185038
2	6221.823280	78.878535	437.297885	2.623325e+05	512.184061
3	22921.110989	151.397196	554.567175	3.976641e+05	630.606141
4	33523.190407	183.093393	370.439785	1.702760e+05	412.645094
5	105.584959	10.275454	34.075515	1.834104e+03	42.826436
6	48061.717358	219.229828	968.181840	1.101085e+06	1049.325744
7	23345.264739	152.791573	605.913460	4.861077e+05	697.214275
8	139.974418	11.831078	40.934713	2.225668e+03	47.176986

Orez DataFrame

In [82]:
```
# Filter the initial DataFrame
product = 'orez'
filter_df = df.loc[df['denumire_produs'] == product]
filter_df = filter_df.drop(columns=['colli', 'store_county'])
#print(apa_plata_df)

# Group by month_id and aggregate the other columns
filter_df_sales = filter_df.groupby("month_id").agg({'sales': 'sum','Inflatie anuala': '
filter_df_sales = filter_df_sales.sort_values(by='month_id', ascending=True)

# Print the apa_plata_sales dataframe
print(filter_df_sales)
```

month_id	sales	Inflatie anuala	unnemployment
2019-01-01	1969.43	3.8	1.42
2019-02-01	2372.65	3.8	3.32
2019-03-01	2056.41	3.8	3.31
2019-04-01	2457.07	3.8	3.19
2019-05-01	2150.26	3.8	3.00
2019-06-01	2515.28	3.8	2.95
2019-07-01	2364.15	3.8	2.95
2019-08-01	1988.91	3.8	3.01
2019-09-01	2213.87	3.8	3.03
2019-10-01	2194.08	3.8	3.00
2019-11-01	2723.12	3.8	2.98
2019-12-01	2853.49	3.8	2.98
2020-01-01	1903.98	2.6	2.97
2020-02-01	3808.90	2.6	2.98
2020-03-01	4881.36	2.6	2.95
2020-04-01	1548.81	2.6	2.88
2020-05-01	2079.69	2.6	2.90
2020-06-01	2244.95	2.6	2.87
2020-07-01	2671.49	2.6	3.00
2020-08-01	2224.49	2.6	3.02
2020-09-01	2737.83	2.6	3.30
2020-10-01	3003.50	2.6	3.26
2020-11-01	2461.79	2.6	3.27
2020-12-01	2876.05	2.6	3.32
2021-01-01	2821.56	5.1	3.38
2021-02-01	2464.96	5.1	3.34
2021-03-01	3215.38	5.1	3.35
2021-04-01	2644.93	5.1	3.33
2021-05-01	2674.71	5.1	3.16
2021-06-01	2767.84	5.1	3.06
2021-07-01	2741.05	5.1	3.00
2021-08-01	3769.09	5.1	2.96
2021-09-01	3788.12	5.1	2.93
2021-10-01	3468.08	5.1	2.85

2021-11-01	4455.87	5.1	2.76
2021-12-01	3670.16	5.1	2.72
2022-01-01	3289.81	13.8	2.69
2022-02-01	3217.05	13.8	2.68
2022-03-01	5079.54	13.8	2.67
2022-04-01	3464.63	13.8	2.64
2022-05-01	3216.84	13.8	2.57
2022-06-01	3467.01	13.8	2.55
2022-07-01	3568.29	13.8	2.55
2022-08-01	2949.64	13.8	2.56
2022-09-01	3232.49	13.8	2.56
2022-10-01	3059.78	13.8	2.88
2022-11-01	3707.90	13.8	2.96
2022-12-01	3398.35	13.8	3.04

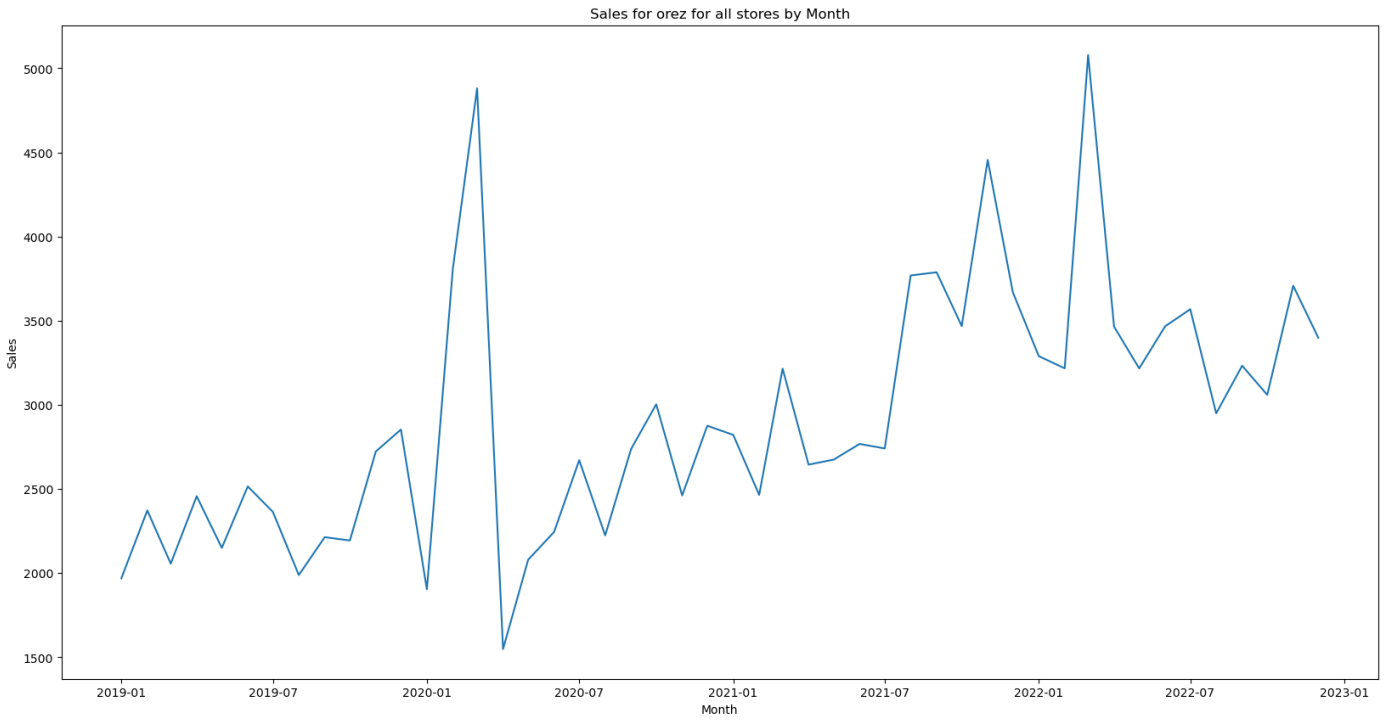
month_id	Inflatie(de la o luna la alta)	net_average_salary
2019-01-01	0.83	2936.0
2019-02-01	0.79	2933.0
2019-03-01	0.49	3075.0
2019-04-01	0.61	3115.0
2019-05-01	0.46	3101.0
2019-06-01	-0.23	3142.0
2019-07-01	-0.20	3119.0
2019-08-01	0.06	3044.0
2019-09-01	0.09	3082.0
2019-10-01	0.43	3116.0
2019-11-01	0.23	3179.0
2019-12-01	0.42	3340.0
2020-01-01	0.41	3189.0
2020-02-01	0.25	3202.0
2020-03-01	0.50	3294.0
2020-04-01	0.26	3182.0
2020-05-01	0.05	3179.0
2020-06-01	0.08	3298.0
2020-07-01	0.00	3372.0
2020-08-01	-0.05	3275.0
2020-09-01	-0.14	3321.0
2020-10-01	0.22	3343.0
2020-11-01	0.13	3411.0
2020-12-01	0.34	3620.0
2021-01-01	1.33	3395.0
2021-02-01	0.41	3365.0
2021-03-01	0.38	3547.0
2021-04-01	0.45	3561.0
2021-05-01	0.53	3492.0
2021-06-01	0.27	3541.0
2021-07-01	0.97	3545.0
2021-08-01	0.24	3487.0
2021-09-01	0.84	3517.0
2021-10-01	1.78	3544.0
2021-11-01	0.00	3645.0
2021-12-01	0.71	3879.0
2022-01-01	1.48	3698.0
2022-02-01	0.58	3721.0
2022-03-01	1.88	3937.0
2022-04-01	3.74	3967.0
2022-05-01	1.18	3928.0
2022-06-01	0.76	3977.0
2022-07-01	0.89	3975.0
2022-08-01	0.56	3933.0
2022-09-01	1.33	4003.0
2022-10-01	1.28	4008.0
2022-11-01	1.25	4141.0
2022-12-01	0.37	4141.0

month_id	IPC Marfuri alimentare (%)	IPC Marfuri nealimentare (%)	IPC Servicii (%)
2019-01-01	101.30	100.63	100.57
2019-02-01	101.27	100.57	100.55
2019-03-01	100.48	100.54	100.40
2019-04-01	100.68	100.53	100.72
2019-05-01	100.79	100.20	100.55
2019-06-01	99.53	99.76	100.17
2019-07-01	99.34	99.98	100.10
2019-08-01	99.71	100.22	100.25
2019-09-01	99.91	100.13	100.27
2019-10-01	100.70	100.32	100.25
2019-11-01	100.43	100.12	100.15
2019-12-01	100.84	100.28	100.12
2020-01-01	100.99	100.02	100.43
2020-02-01	100.63	99.94	100.41
2020-03-01	101.46	99.91	100.40
2020-04-01	101.27	99.67	100.39
2020-05-01	100.34	99.82	100.39
2020-06-01	99.62	100.28	100.39
2020-07-01	99.55	100.19	100.39
2020-08-01	99.59	100.08	100.39
2020-09-01	99.45	99.99	100.39
2020-10-01	100.11	100.31	100.39
2020-11-01	99.92	100.29	100.39
2020-12-01	100.29	100.51	100.39
2021-01-01	100.63	102.24	100.39
2021-02-01	100.46	100.47	100.39
2021-03-01	100.37	100.46	100.39
2021-04-01	100.45	100.47	100.39
2021-05-01	101.10	100.28	100.39
2021-06-01	100.25	100.29	100.39
2021-07-01	99.71	102.02	100.39
2021-08-01	99.95	100.34	100.39
2021-09-01	100.95	100.73	100.39
2021-10-01	101.06	102.78	100.39
2021-11-01	100.73	99.47	100.39
2021-12-01	100.84	100.74	100.39
2022-01-01	101.15	101.73	100.39
2022-02-01	101.96	99.70	100.39
2022-03-01	102.54	101.86	100.39
2022-04-01	102.56	105.45	100.39
2022-05-01	101.73	101.00	100.39
2022-06-01	100.62	100.92	100.39
2022-07-01	100.92	100.87	100.39
2022-08-01	101.82	99.81	100.39
2022-09-01	101.72	101.28	100.39
2022-10-01	102.29	100.80	100.39
2022-11-01	101.54	101.03	100.39
2022-12-01	101.26	99.68	100.39

```
2020-02-01            100.39
2020-03-01            100.35
2020-04-01            100.00
2020-05-01            100.11
2020-06-01            100.34
2020-07-01            100.31
2020-08-01            100.21
2020-09-01            100.20
2020-10-01            100.22
2020-11-01            100.07
2020-12-01            100.04
2021-01-01            100.25
2021-02-01            100.20
2021-03-01            100.24
2021-04-01            100.40
2021-05-01            100.27
2021-06-01            100.23
2021-07-01            100.39
2021-08-01            100.43
2021-09-01            100.96
2021-10-01            100.42
2021-11-01            100.20
2021-12-01            100.42
2022-01-01            101.37
2022-02-01            100.60
2022-03-01            100.67
2022-04-01            100.94
2022-05-01            100.61
2022-06-01            100.54
2022-07-01            100.88
2022-08-01            100.37
2022-09-01            100.72
2022-10-01            100.70
2022-11-01            101.31
2022-12-01            100.67
```

In [83]:

```
# Line graph for apa_plata for all stores
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index.get_level_values(0), filter_df_sales['sales'])
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales for ' + product + ' for all stores by Month')
plt.show()
```

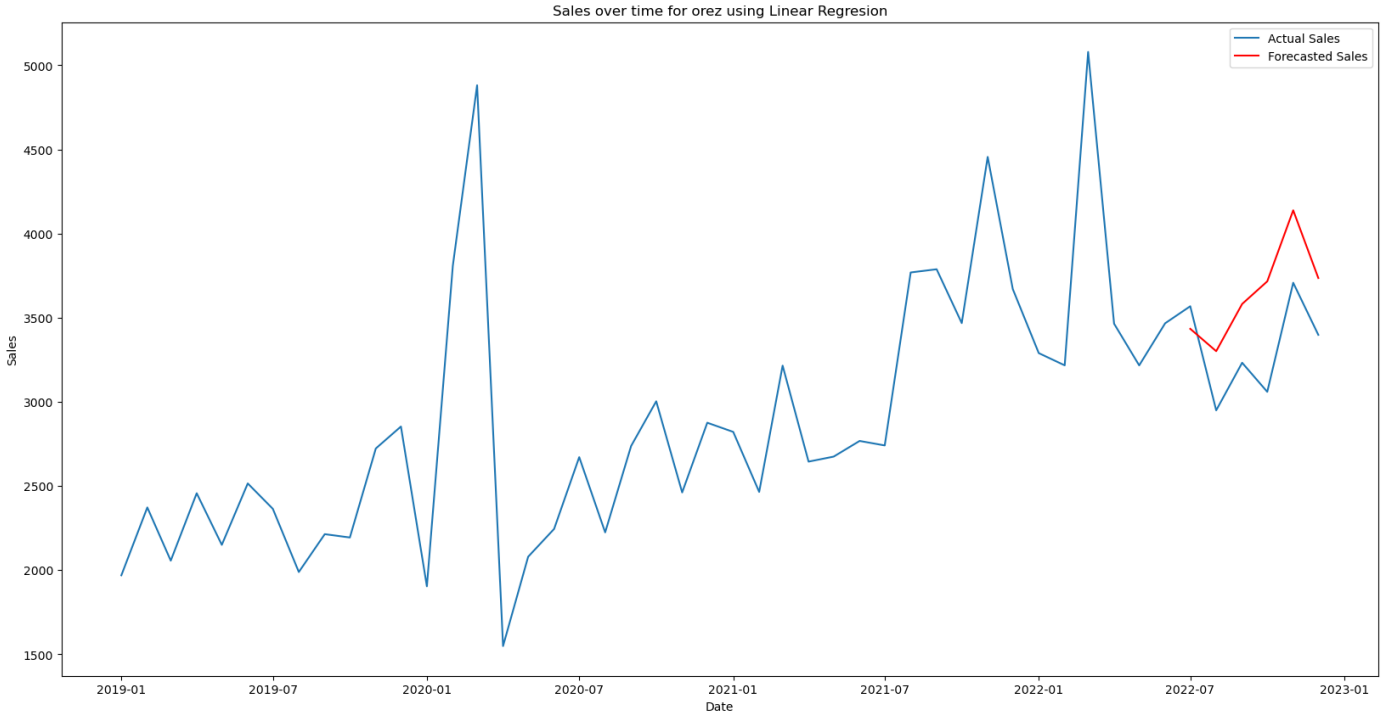


LR - orez

In [84]:
```
# Get the last 6 months of data
last_6_months = filter_df_sales.tail(6)

# Split the data into train and test sets
X_train = filter_df_sales.drop('sales', axis=1)
y_train = filter_df_sales['sales']
X_test = last_6_months.drop('sales', axis=1)
y_test = last_6_months['sales']

# Create a linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the error
lr_mae = mean_absolute_error(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)

print("Mean Absolute Error: ", lr_mae)
print("Mean Squared Error: ", lr_mse)
print("Root Mean Squared Error:", lr_rmse)

all_sales = filter_df_sales['sales']

# Get the index of the last 6 months of data
last_6_months_index = last_6_months.index

# Plot the entire period of sales data
plt.figure(figsize=(20,10))
plt.plot(all_sales)

# Make predictions on the last 6 months of data
```

```
y_pred = model.predict(X_test)

# Plot the forecasted sales on the same chart
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Linear Regresion')
plt.legend(['Actual Sales', 'Forecasted Sales'])

plt.show()
```

```
Mean Absolute Error:  376.60136133500333
Mean Squared Error:  165639.5949179056
Root Mean Squared Error: 406.9884456810852
```



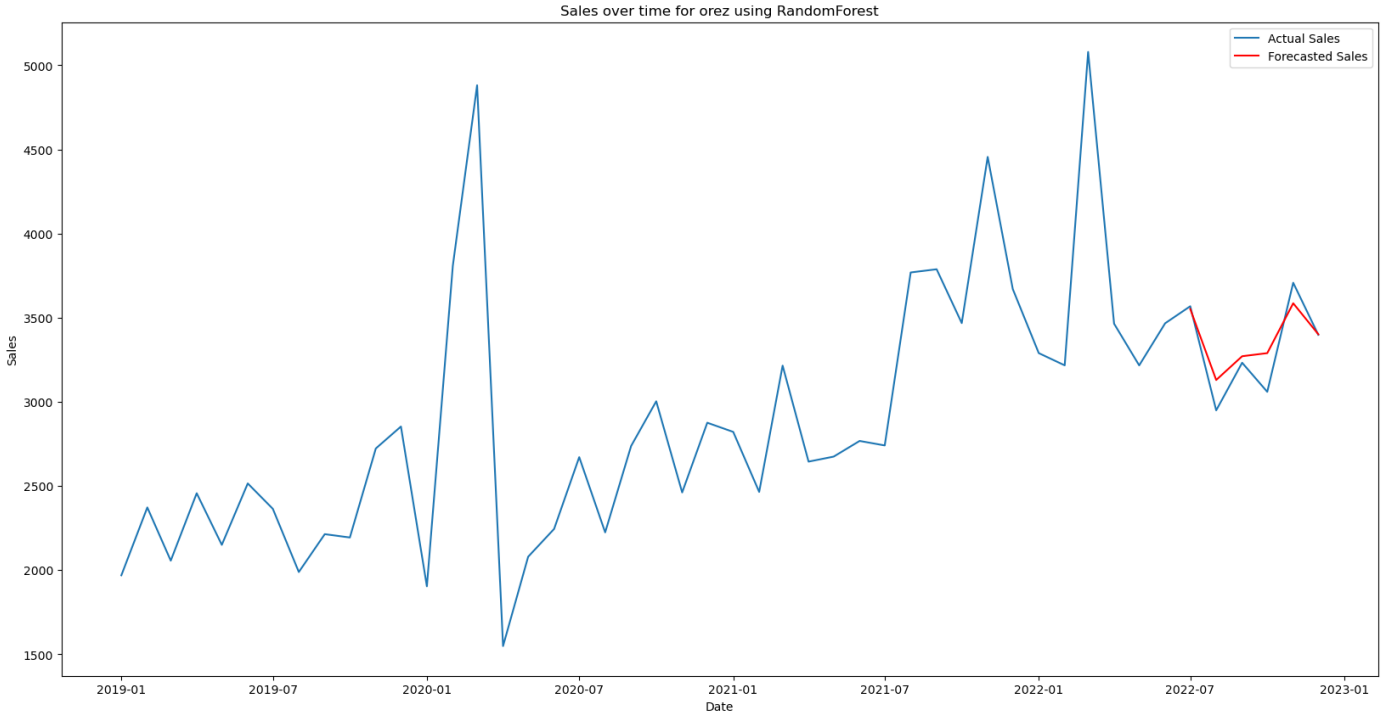
RF - orez

In [85]:

```
last_6_months = filter_df_sales.tail(6)

X_train = filter_df_sales.drop(columns=["sales"])
y_train = filter_df_sales["sales"]
X_test = last_6_months.drop(columns=["sales"])
y_test = last_6_months["sales"]

# Create a random forest model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Use the model to make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate the model's accuracy
accuracy = rf.score(X_test, y_test)
print("Accuracy: ", accuracy)

# Calculate error
rf_mae = mean_absolute_error(y_test, y_pred)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rf_mse)
```

```
print("Mean Absolute Error: ", rf_mae)
print("Mean Squared Error: ", rf_mse)
print("Root Mean Squared Error:", rf_rmse)

# Line graph for the forcast
plt.figure(figsize=(20,10))

plt.plot(all_sales)
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using RandomForest')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
Accuracy:  0.7631470038958155
Mean Absolute Error:  98.47421666666735
Mean Squared Error:  17005.647771088712
Root Mean Squared Error: 130.40570451896923
```



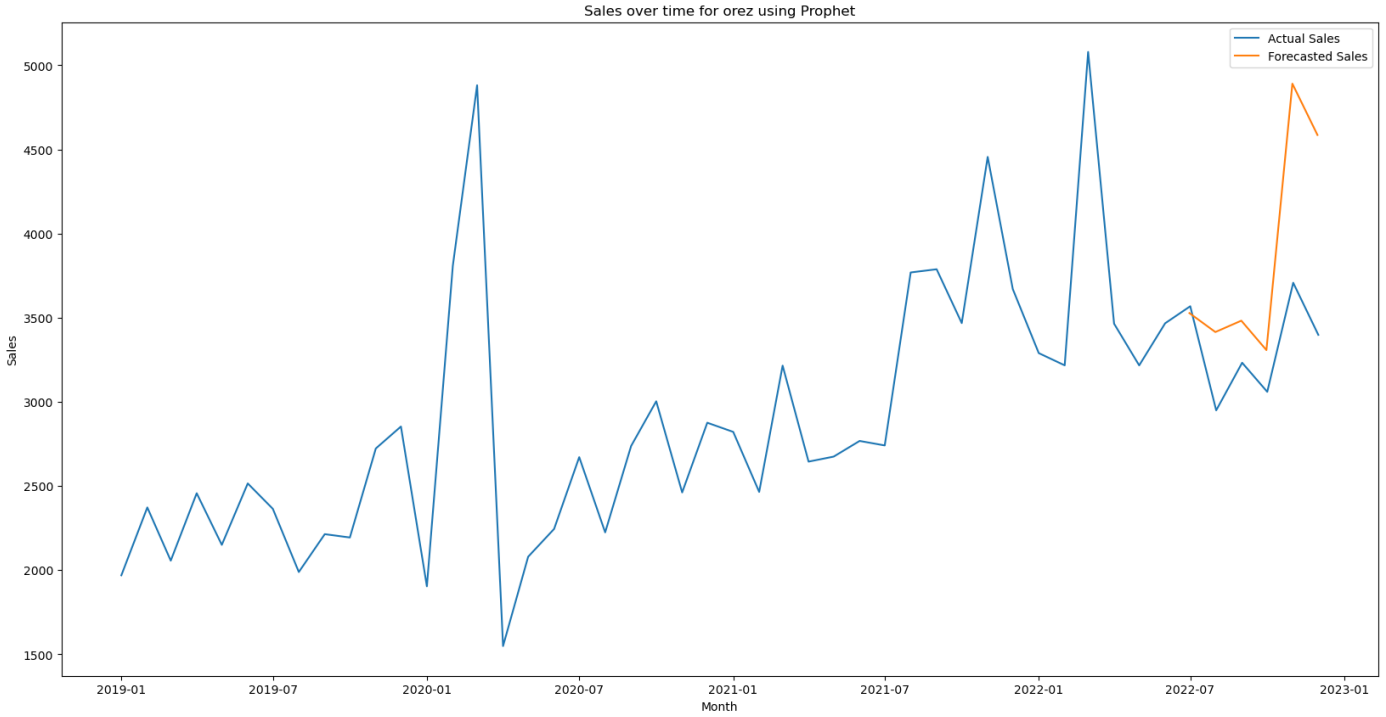
PR - orez

In [86]:

```
# create a copy of the dataframe with a column named 'ds' with the index values
filter_df_sales_prophet = filter_df_sales.reset_index().rename(columns={'month_id': 'ds'
filter_df_sales_prophet = filter_df_sales_prophet.rename(columns={'sales':'y'})

train_data = filter_df_sales_prophet[:-6]
test_data = filter_df_sales_prophet[-6:]

model = Prophet()

# Fit the model on the data
model.fit(train_data)

# Create a dataframe to hold the forecast
future_data = model.make_future_dataframe(periods=6, freq='m')

# Make predictions
forecast = model.predict(future_data)
forecast = forecast.tail(6)
```

```
# Calculate error
pr_mae = mean_absolute_error(test_data['y'], forecast[-6:]['yhat'])
pr_mse = mean_squared_error(test_data['y'], forecast[-6:]['yhat'])
pr_rmse = np.sqrt(pr_mse)
print("Mean Absolute Error:", pr_mae)
print("Mean Squared Error:", pr_mse)
print("Root Mean Squared Error:", pr_rmse)

# Plot the forecast
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index, filter_df_sales['sales'], label='Actual Sales')

# Plot the predicted values for the last six months
plt.plot(forecast.ds, forecast.yhat, label='Predicted Sales')

plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Prophet')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
21:35:19 - cmdstanpy - INFO - Chain [1] start processing
21:35:20 - cmdstanpy - INFO - Chain [1] done processing
Mean Absolute Error: 562.4378904145551
Mean Squared Error: 525392.3273153774
Root Mean Squared Error: 724.8395183179359
```



In [87]:

```
# Add the errors from all methods to the DataFrame
new_row_df =  pd.DataFrame({'product': [product], 'LR-MAE': [lr_mae], 'LR-MSE': [lr_mse]
errors_df = pd.concat([errors_df, new_row_df], ignore_index=True)
print(errors_df)
```

	product	LR-MAE	LR-MSE	LR-RMSE	RF-MAE	\
0	apa_plata	887.015829	988186.848307	994.075877	377.426267	
1	apa_minerala	44.324535	2453.366899	49.531474	10.533100	
2	bere	283.109792	95844.002658	309.586826	65.444150	
3	cafea	287.818261	94385.213013	307.221765	134.035500	
4	carnati	264.936590	99377.947534	315.242680	155.934000	
5	frigider	13.054626	245.441803	15.666582	8.523900	
6	lapte	490.769164	269944.557868	519.561890	193.309633	
7	legume	317.334317	168366.068324	410.324345	124.748417	

```
8  masina_de_spalat    18.592357     606.867326   24.634677    8.862467
9              orez   376.601361  165639.594918  406.988446   98.474217

          RF-MSE     RF-RMSE       PR-MAE        PR-MSE      PR-RMSE
0  200759.770117  448.062239  1107.910649  1.661801e+06  1289.108561
1     153.080170   12.372557  1703.258881  1.172504e+07  3424.185038
2    6221.823280   78.878535   437.297885  2.623325e+05   512.184061
3   22921.110989  151.397196   554.567175  3.976641e+05   630.606141
4   33523.190407  183.093393   370.439785  1.702760e+05   412.645094
5     105.584959   10.275454    34.075515  1.834104e+03    42.826436
6   48061.717358  219.229828   968.181840  1.101085e+06  1049.325744
7   23345.264739  152.791573   605.913460  4.861077e+05   697.214275
8     139.974418   11.831078    40.934713  2.225668e+03    47.176986
9   17005.647771  130.405705   562.437890  5.253923e+05   724.839518
```

Oua DataFrame

In [88]:

```
# Filter the initial DataFrame
product = 'oua'
filter_df = df.loc[df['denumire_produs'] == product]
filter_df = filter_df.drop(columns=['colli', 'store_county'])
#print(apa_plata_df)

# Group by month_id and aggregate the other columns
filter_df_sales = filter_df.groupby("month_id").agg({'sales': 'sum','Inflatie anuala': '
filter_df_sales = filter_df_sales.sort_values(by='month_id', ascending=True)

# Print the apa_plata_sales dataframe
print(filter_df_sales)
```

```
              sales  Inflatie anuala  unnemployment  \
month_id
2019-01-01  3191.59              3.8           1.42
2019-02-01  2520.09              3.8           3.32
2019-03-01  3217.99              3.8           3.31
2019-04-01  3353.62              3.8           3.19
2019-05-01  2487.07              3.8           3.00
2019-06-01  2998.63              3.8           2.95
2019-07-01  2988.50              3.8           2.95
2019-08-01  3252.98              3.8           3.01
2019-09-01  2987.78              3.8           3.03
2019-10-01  3111.00              3.8           3.00
2019-11-01  3281.98              3.8           2.98
2019-12-01  3944.47              3.8           2.98
2020-01-01  3740.51              2.6           2.97
2020-02-01  3377.26              2.6           2.98
2020-03-01  5038.33              2.6           2.95
2020-04-01  3182.10              2.6           2.88
2020-05-01  3029.75              2.6           2.90
2020-06-01  4205.38              2.6           2.87
2020-07-01  3644.65              2.6           3.00
2020-08-01  3832.07              2.6           3.02
2020-09-01  4437.19              2.6           3.30
2020-10-01  3469.83              2.6           3.26
2020-11-01  4266.28              2.6           3.27
2020-12-01  4684.68              2.6           3.32
2021-01-01  4033.08              5.1           3.38
2021-02-01  3343.81              5.1           3.34
2021-03-01  3415.69              5.1           3.35
2021-04-01  5001.10              5.1           3.33
2021-05-01  4048.60              5.1           3.16
2021-06-01  3723.52              5.1           3.06
2021-07-01  3256.70              5.1           3.00
2021-08-01  4306.94              5.1           2.96
```

2021-09-01	4347.46	5.1	2.93
2021-10-01	4561.33	5.1	2.85
2021-11-01	4123.69	5.1	2.76
2021-12-01	4850.90	5.1	2.72
2022-01-01	4181.89	13.8	2.69
2022-02-01	4699.15	13.8	2.68
2022-03-01	4470.28	13.8	2.67
2022-04-01	4355.17	13.8	2.64
2022-05-01	3331.72	13.8	2.57
2022-06-01	3789.82	13.8	2.55
2022-07-01	2910.64	13.8	2.55
2022-08-01	2996.69	13.8	2.56
2022-09-01	4738.11	13.8	2.56
2022-10-01	4312.22	13.8	2.88
2022-11-01	3596.37	13.8	2.96
2022-12-01	5114.52	13.8	3.04

month_id	Inflatie(de la o luna la alta)	net_average_salary
2019-01-01	0.83	2936.0
2019-02-01	0.79	2933.0
2019-03-01	0.49	3075.0
2019-04-01	0.61	3115.0
2019-05-01	0.46	3101.0
2019-06-01	-0.23	3142.0
2019-07-01	-0.20	3119.0
2019-08-01	0.06	3044.0
2019-09-01	0.09	3082.0
2019-10-01	0.43	3116.0
2019-11-01	0.23	3179.0
2019-12-01	0.42	3340.0
2020-01-01	0.41	3189.0
2020-02-01	0.25	3202.0
2020-03-01	0.50	3294.0
2020-04-01	0.26	3182.0
2020-05-01	0.05	3179.0
2020-06-01	0.08	3298.0
2020-07-01	0.00	3372.0
2020-08-01	-0.05	3275.0
2020-09-01	-0.14	3321.0
2020-10-01	0.22	3343.0
2020-11-01	0.13	3411.0
2020-12-01	0.34	3620.0
2021-01-01	1.33	3395.0
2021-02-01	0.41	3365.0
2021-03-01	0.38	3547.0
2021-04-01	0.45	3561.0
2021-05-01	0.53	3492.0
2021-06-01	0.27	3541.0
2021-07-01	0.97	3545.0
2021-08-01	0.24	3487.0
2021-09-01	0.84	3517.0
2021-10-01	1.78	3544.0
2021-11-01	0.00	3645.0
2021-12-01	0.71	3879.0
2022-01-01	1.48	3698.0
2022-02-01	0.58	3721.0
2022-03-01	1.88	3937.0
2022-04-01	3.74	3967.0
2022-05-01	1.18	3928.0
2022-06-01	0.76	3977.0
2022-07-01	0.89	3975.0
2022-08-01	0.56	3933.0
2022-09-01	1.33	4003.0
2022-10-01	1.28	4008.0
2022-11-01	1.25	4141.0

2022-12-01	0.37	4141.0	

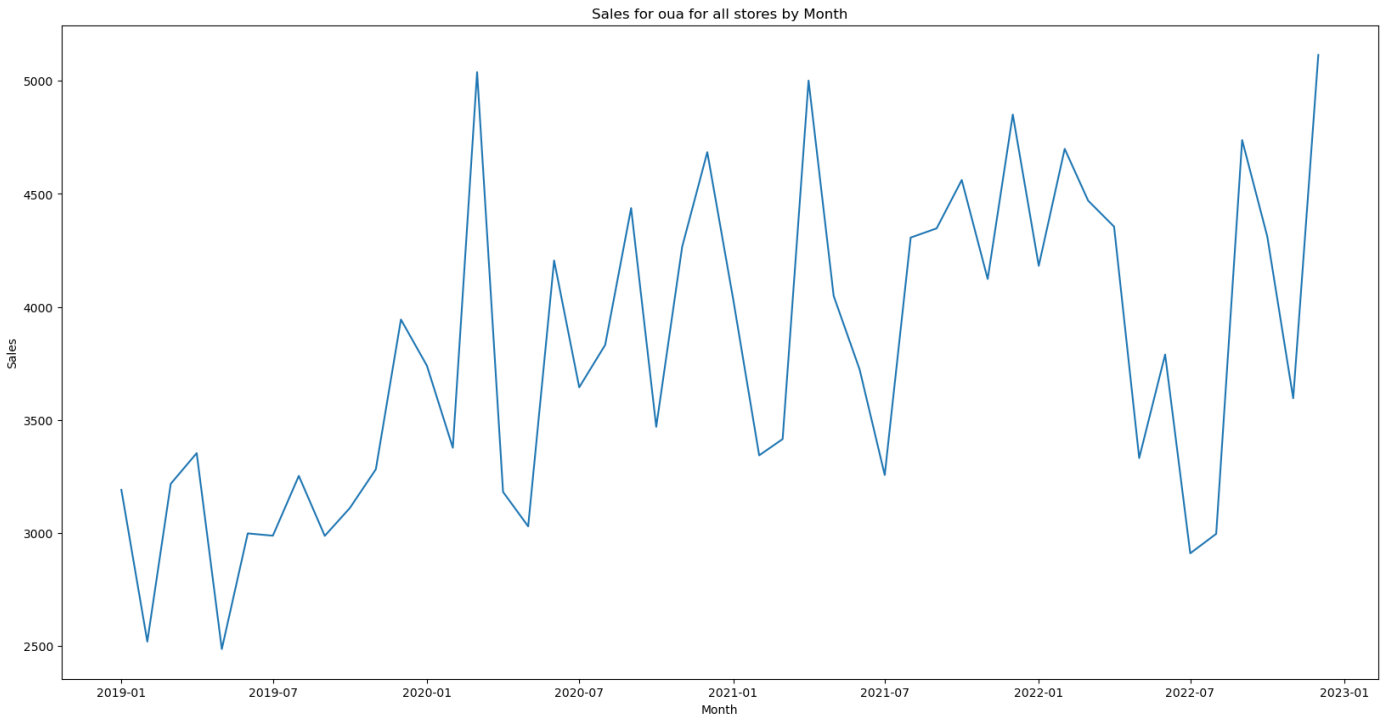
month_id	IPC Marfuri alimentare (%)	IPC Marfuri nealimentare (%)	\
2019-01-01	101.30	100.63	
2019-02-01	101.27	100.57	
2019-03-01	100.48	100.54	
2019-04-01	100.68	100.53	
2019-05-01	100.79	100.20	
2019-06-01	99.53	99.76	
2019-07-01	99.34	99.98	
2019-08-01	99.71	100.22	
2019-09-01	99.91	100.13	
2019-10-01	100.70	100.32	
2019-11-01	100.43	100.12	
2019-12-01	100.84	100.28	
2020-01-01	100.99	100.02	
2020-02-01	100.63	99.94	
2020-03-01	101.46	99.91	
2020-04-01	101.27	99.67	
2020-05-01	100.34	99.82	
2020-06-01	99.62	100.28	
2020-07-01	99.55	100.19	
2020-08-01	99.59	100.08	
2020-09-01	99.45	99.99	
2020-10-01	100.11	100.31	
2020-11-01	99.92	100.29	
2020-12-01	100.29	100.51	
2021-01-01	100.63	102.24	
2021-02-01	100.46	100.47	
2021-03-01	100.37	100.46	
2021-04-01	100.45	100.47	
2021-05-01	101.10	100.28	
2021-06-01	100.25	100.29	
2021-07-01	99.71	102.02	
2021-08-01	99.95	100.34	
2021-09-01	100.95	100.73	
2021-10-01	101.06	102.78	
2021-11-01	100.73	99.47	
2021-12-01	100.84	100.74	
2022-01-01	101.15	101.73	
2022-02-01	101.96	99.70	
2022-03-01	102.54	101.86	
2022-04-01	102.56	105.45	
2022-05-01	101.73	101.00	
2022-06-01	100.62	100.92	
2022-07-01	100.92	100.87	
2022-08-01	101.82	99.81	
2022-09-01	101.72	101.28	
2022-10-01	102.29	100.80	
2022-11-01	101.54	101.03	
2022-12-01	101.26	99.68	

month_id	IPC Servicii (%)
2019-01-01	100.57
2019-02-01	100.55
2019-03-01	100.40
2019-04-01	100.72
2019-05-01	100.55
2019-06-01	100.17
2019-07-01	100.10
2019-08-01	100.25
2019-09-01	100.27
2019-10-01	100.25
2019-11-01	100.15

```
2019-12-01            100.12
2020-01-01            100.43
2020-02-01            100.39
2020-03-01            100.35
2020-04-01            100.00
2020-05-01            100.11
2020-06-01            100.34
2020-07-01            100.31
2020-08-01            100.21
2020-09-01            100.20
2020-10-01            100.22
2020-11-01            100.07
2020-12-01            100.04
2021-01-01            100.25
2021-02-01            100.20
2021-03-01            100.24
2021-04-01            100.40
2021-05-01            100.27
2021-06-01            100.23
2021-07-01            100.39
2021-08-01            100.43
2021-09-01            100.96
2021-10-01            100.42
2021-11-01            100.20
2021-12-01            100.42
2022-01-01            101.37
2022-02-01            100.60
2022-03-01            100.67
2022-04-01            100.94
2022-05-01            100.61
2022-06-01            100.54
2022-07-01            100.88
2022-08-01            100.37
2022-09-01            100.72
2022-10-01            100.70
2022-11-01            101.31
2022-12-01            100.67
```

In [89]:

```
# Line graph for apa_plata for all stores
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index.get_level_values(0), filter_df_sales['sales'])
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales for ' + product + ' for all stores by Month')
plt.show()
```

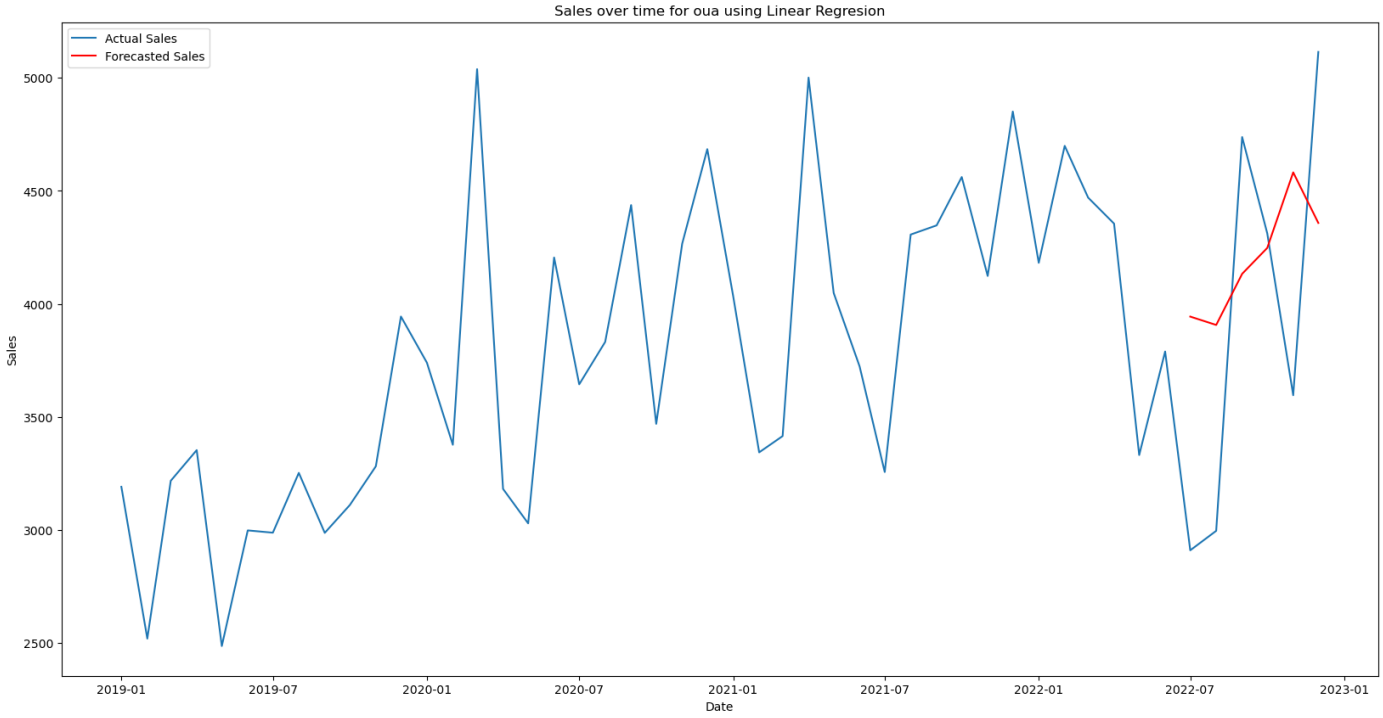


LR - oua

In [90]:
```
# Get the last 6 months of data
last_6_months = filter_df_sales.tail(6)

# Split the data into train and test sets
X_train = filter_df_sales.drop('sales', axis=1)
y_train = filter_df_sales['sales']
X_test = last_6_months.drop('sales', axis=1)
y_test = last_6_months['sales']

# Create a linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the error
lr_mae = mean_absolute_error(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)

print("Mean Absolute Error: ", lr_mae)
print("Mean Squared Error: ", lr_mse)
print("Root Mean Squared Error:", lr_rmse)

all_sales = filter_df_sales['sales']

# Get the index of the last 6 months of data
last_6_months_index = last_6_months.index

# Plot the entire period of sales data
plt.figure(figsize=(20,10))
plt.plot(all_sales)

# Make predictions on the last 6 months of data
```

```
y_pred = model.predict(X_test)

# Plot the forecasted sales on the same chart
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Linear Regresion')
plt.legend(['Actual Sales', 'Forecasted Sales'])

plt.show()
```

```
Mean Absolute Error:  725.8358588237083
Mean Squared Error:  635033.7038031333
Root Mean Squared Error: 796.8900198917875
```



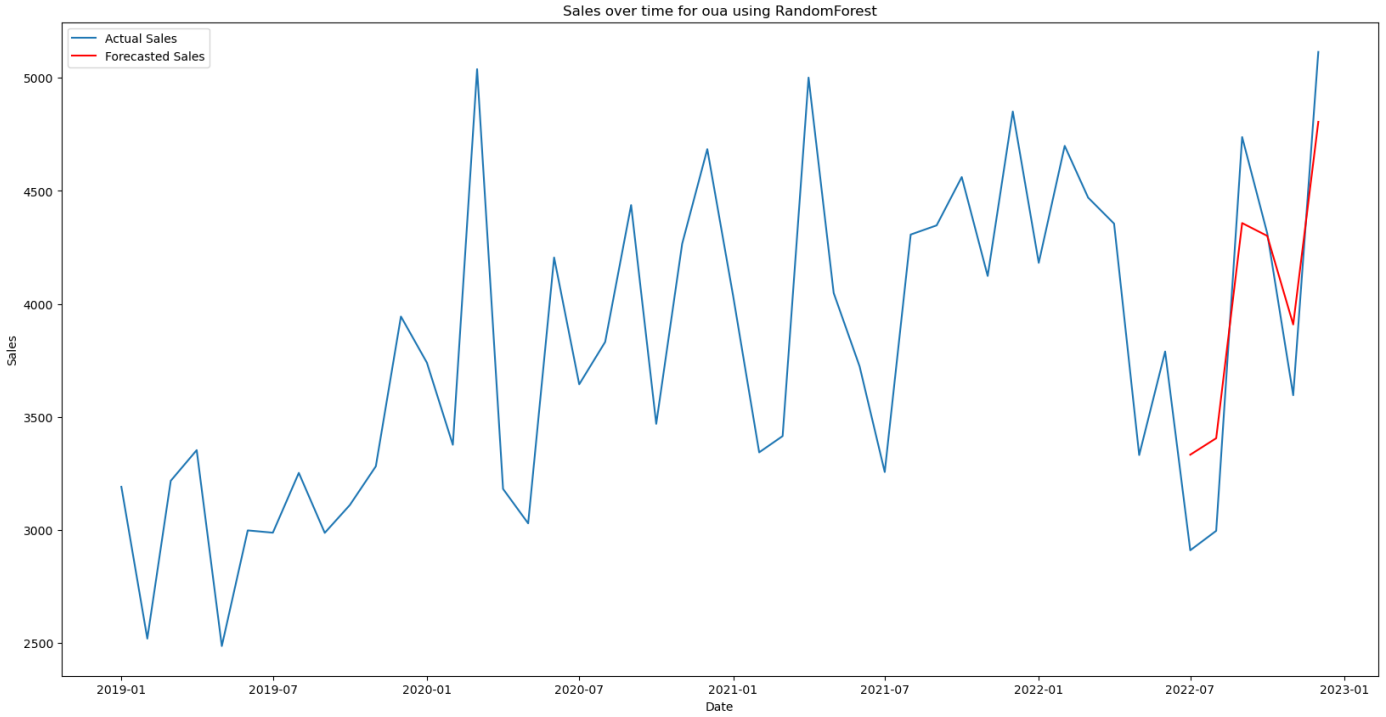
RF - oua

In [91]:

```
last_6_months = filter_df_sales.tail(6)

X_train = filter_df_sales.drop(columns=["sales"])
y_train = filter_df_sales["sales"]
X_test = last_6_months.drop(columns=["sales"])
y_test = last_6_months["sales"]

# Create a random forest model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Use the model to make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate the model's accuracy
accuracy = rf.score(X_test, y_test)
print("Accuracy: ", accuracy)

# Calculate error
rf_mae = mean_absolute_error(y_test, y_pred)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rf_mse)
```

```
print("Mean Absolute Error: ", rf_mae)
print("Mean Squared Error: ", rf_mse)
print("Root Mean Squared Error:", rf_rmse)

# Line graph for the forcast
plt.figure(figsize=(20,10))

plt.plot(all_sales)
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using RandomForest')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
Accuracy:  0.8378857224580227
Mean Absolute Error:  307.59591666666915
Mean Squared Error:  114084.8432725829
Root Mean Squared Error: 337.7644789976929
```



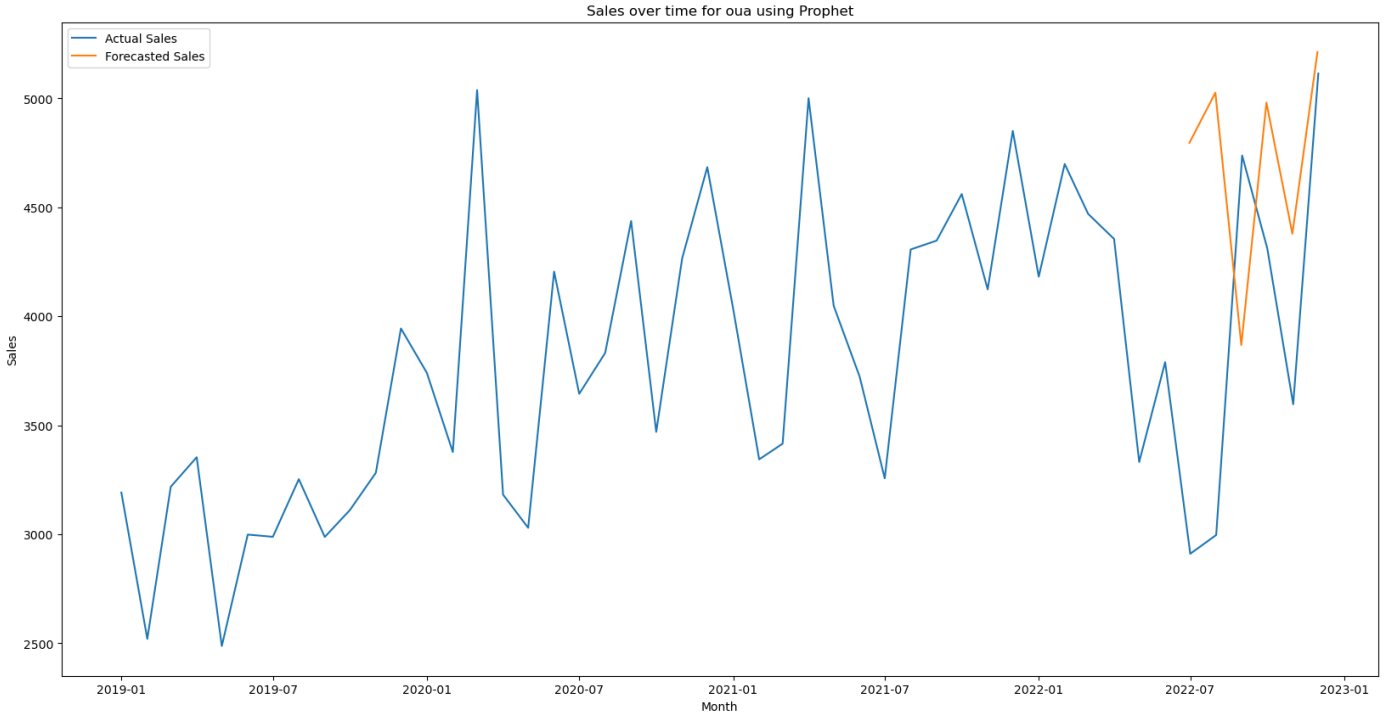
PR - oua

In [92]:

```
# create a copy of the dataframe with a column named 'ds' with the index values
filter_df_sales_prophet = filter_df_sales.reset_index().rename(columns={'month_id': 'ds'
filter_df_sales_prophet = filter_df_sales_prophet.rename(columns={'sales':'y'})

train_data = filter_df_sales_prophet[:-6]
test_data = filter_df_sales_prophet[-6:]

model = Prophet()

# Fit the model on the data
model.fit(train_data)

# Create a dataframe to hold the forecast
future_data = model.make_future_dataframe(periods=6, freq='m')

# Make predictions
forecast = model.predict(future_data)
forecast = forecast.tail(6)
```

```
# Calculate error
pr_mae = mean_absolute_error(test_data['y'], forecast[-6:]['yhat'])
pr_mse = mean_squared_error(test_data['y'], forecast[-6:]['yhat'])
pr_rmse = np.sqrt(pr_mse)
print("Mean Absolute Error:", pr_mae)
print("Mean Squared Error:", pr_mse)
print("Root Mean Squared Error:", pr_rmse)

# Plot the forecast
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index, filter_df_sales['sales'], label='Actual Sales')

# Plot the predicted values for the last six months
plt.plot(forecast.ds, forecast.yhat, label='Predicted Sales')

plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Prophet')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
21:35:21 - cmdstanpy - INFO - Chain [1] start processing
21:35:21 - cmdstanpy - INFO - Chain [1] done processing
Mean Absolute Error: 1055.7345501111292
Mean Squared Error: 1583207.5016601842
Root Mean Squared Error: 1258.2557377815467
```



In [93]:

```
# Add the errors from all methods to the DataFrame
new_row_df =  pd.DataFrame({'product': [product], 'LR-MAE': [lr_mae], 'LR-MSE': [lr_mse]
errors_df = pd.concat([errors_df, new_row_df], ignore_index=True)
print(errors_df)
```

	product	LR-MAE	LR-MSE	LR-RMSE	RF-MAE	\
0	apa_plata	887.015829	988186.848307	994.075877	377.426267	
1	apa_minerala	44.324535	2453.366899	49.531474	10.533100	
2	bere	283.109792	95844.002658	309.586826	65.444150	
3	cafea	287.818261	94385.213013	307.221765	134.035500	
4	carnati	264.936590	99377.947534	315.242680	155.934000	
5	frigider	13.054626	245.441803	15.666582	8.523900	
6	lapte	490.769164	269944.557868	519.561890	193.309633	
7	legume	317.334317	168366.068324	410.324345	124.748417	

8	masina_de_spalat	18.592357	606.867326	24.634677	8.862467
9	orez	376.601361	165639.594918	406.988446	98.474217
10	oua	725.835859	635033.703803	796.890020	307.595917

	RF-MSE	RF-RMSE	PR-MAE	PR-MSE	PR-RMSE
0	200759.770117	448.062239	1107.910649	1.661801e+06	1289.108561
1	153.080170	12.372557	1703.258881	1.172504e+07	3424.185038
2	6221.823280	78.878535	437.297885	2.623325e+05	512.184061
3	22921.110989	151.397196	554.567175	3.976641e+05	630.606141
4	33523.190407	183.093393	370.439785	1.702760e+05	412.645094
5	105.584959	10.275454	34.075515	1.834104e+03	42.826436
6	48061.717358	219.229828	968.181840	1.101085e+06	1049.325744
7	23345.264739	152.791573	605.913460	4.861077e+05	697.214275
8	139.974418	11.831078	40.934713	2.225668e+03	47.176986
9	17005.647771	130.405705	562.437890	5.253923e+05	724.839518
10	114084.843273	337.764479	1055.734550	1.583208e+06	1258.255738

Paste DataFrame

In [94]:
```
# Filter the initial DataFrame
product = 'paste'
filter_df = df.loc[df['denumire_produs'] == product]
filter_df = filter_df.drop(columns=['colli', 'store_county'])
#print(apa_plata_df)

# Group by month_id and aggregate the other columns
filter_df_sales = filter_df.groupby("month_id").agg({'sales': 'sum','Inflatie anuala': '
filter_df_sales = filter_df_sales.sort_values(by='month_id', ascending=True)

# Print the apa_plata_sales dataframe
print(filter_df_sales)
```

month_id	sales	Inflatie anuala	unnemployment
2019-01-01	198.11	3.8	1.42
2019-02-01	130.38	3.8	3.32
2019-03-01	292.67	3.8	3.31
2019-04-01	500.69	3.8	3.19
2019-05-01	208.61	3.8	3.00
2019-06-01	122.38	3.8	2.95
2019-07-01	166.37	3.8	2.95
2019-08-01	236.58	3.8	3.01
2019-09-01	433.63	3.8	3.03
2019-10-01	243.70	3.8	3.00
2019-11-01	477.22	3.8	2.98
2019-12-01	398.83	3.8	2.98
2020-01-01	275.14	2.6	2.97
2020-02-01	262.92	2.6	2.98
2020-03-01	650.92	2.6	2.95
2020-04-01	794.37	2.6	2.88
2020-05-01	175.48	2.6	2.90
2020-06-01	285.18	2.6	2.87
2020-07-01	236.44	2.6	3.00
2020-08-01	240.01	2.6	3.02
2020-09-01	126.89	2.6	3.30
2020-10-01	378.03	2.6	3.26
2020-11-01	191.56	2.6	3.27
2020-12-01	332.53	2.6	3.32
2021-01-01	349.54	5.1	3.38
2021-02-01	198.83	5.1	3.34
2021-03-01	309.39	5.1	3.35
2021-04-01	1392.26	5.1	3.33
2021-05-01	201.91	5.1	3.16
2021-06-01	376.05	5.1	3.06

2021-07-01	426.19	5.1	3.00
2021-08-01	706.13	5.1	2.96
2021-09-01	406.95	5.1	2.93
2021-10-01	332.31	5.1	2.85
2021-11-01	348.51	5.1	2.76
2021-12-01	352.17	5.1	2.72
2022-01-01	354.88	13.8	2.69
2022-02-01	467.11	13.8	2.68
2022-03-01	814.56	13.8	2.67
2022-04-01	1514.45	13.8	2.64
2022-05-01	484.15	13.8	2.57
2022-06-01	474.69	13.8	2.55
2022-07-01	432.37	13.8	2.55
2022-08-01	594.01	13.8	2.56
2022-09-01	584.69	13.8	2.56
2022-10-01	413.15	13.8	2.88
2022-11-01	951.23	13.8	2.96
2022-12-01	740.53	13.8	3.04

month_id	Inflatie(de la o luna la alta)	net_average_salary	\
2019-01-01	0.83	2936.0	
2019-02-01	0.79	2933.0	
2019-03-01	0.49	3075.0	
2019-04-01	0.61	3115.0	
2019-05-01	0.46	3101.0	
2019-06-01	-0.23	3142.0	
2019-07-01	-0.20	3119.0	
2019-08-01	0.06	3044.0	
2019-09-01	0.09	3082.0	
2019-10-01	0.43	3116.0	
2019-11-01	0.23	3179.0	
2019-12-01	0.42	3340.0	
2020-01-01	0.41	3189.0	
2020-02-01	0.25	3202.0	
2020-03-01	0.50	3294.0	
2020-04-01	0.26	3182.0	
2020-05-01	0.05	3179.0	
2020-06-01	0.08	3298.0	
2020-07-01	0.00	3372.0	
2020-08-01	-0.05	3275.0	
2020-09-01	-0.14	3321.0	
2020-10-01	0.22	3343.0	
2020-11-01	0.13	3411.0	
2020-12-01	0.34	3620.0	
2021-01-01	1.33	3395.0	
2021-02-01	0.41	3365.0	
2021-03-01	0.38	3547.0	
2021-04-01	0.45	3561.0	
2021-05-01	0.53	3492.0	
2021-06-01	0.27	3541.0	
2021-07-01	0.97	3545.0	
2021-08-01	0.24	3487.0	
2021-09-01	0.84	3517.0	
2021-10-01	1.78	3544.0	
2021-11-01	0.00	3645.0	
2021-12-01	0.71	3879.0	
2022-01-01	1.48	3698.0	
2022-02-01	0.58	3721.0	
2022-03-01	1.88	3937.0	
2022-04-01	3.74	3967.0	
2022-05-01	1.18	3928.0	
2022-06-01	0.76	3977.0	
2022-07-01	0.89	3975.0	
2022-08-01	0.56	3933.0	
2022-09-01	1.33	4003.0	

2022-10-01	1.28	4008.0
2022-11-01	1.25	4141.0
2022-12-01	0.37	4141.0

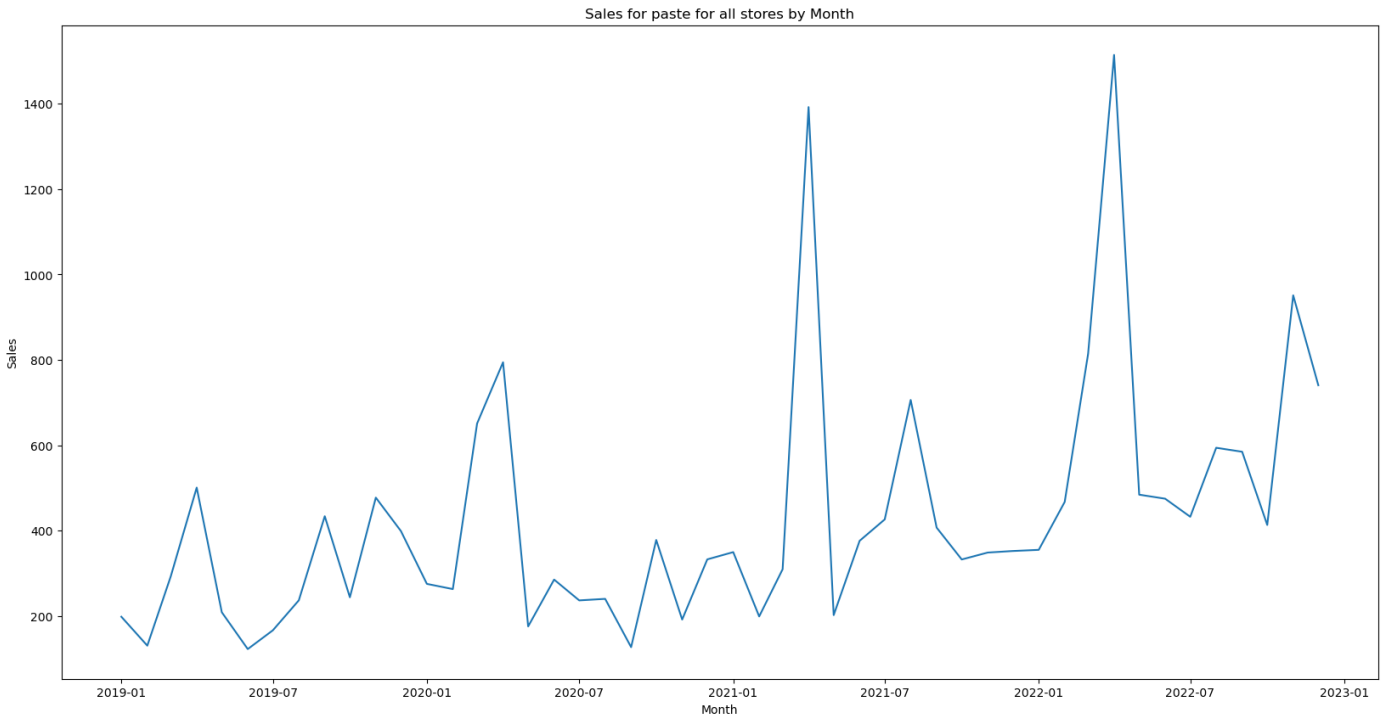
month_id	IPC Marfuri alimentare (%)	IPC Marfuri nealimentare (%)
2019-01-01	101.30	100.63
2019-02-01	101.27	100.57
2019-03-01	100.48	100.54
2019-04-01	100.68	100.53
2019-05-01	100.79	100.20
2019-06-01	99.53	99.76
2019-07-01	99.34	99.98
2019-08-01	99.71	100.22
2019-09-01	99.91	100.13
2019-10-01	100.70	100.32
2019-11-01	100.43	100.12
2019-12-01	100.84	100.28
2020-01-01	100.99	100.02
2020-02-01	100.63	99.94
2020-03-01	101.46	99.91
2020-04-01	101.27	99.67
2020-05-01	100.34	99.82
2020-06-01	99.62	100.28
2020-07-01	99.55	100.19
2020-08-01	99.59	100.08
2020-09-01	99.45	99.99
2020-10-01	100.11	100.31
2020-11-01	99.92	100.29
2020-12-01	100.29	100.51
2021-01-01	100.63	102.24
2021-02-01	100.46	100.47
2021-03-01	100.37	100.46
2021-04-01	100.45	100.47
2021-05-01	101.10	100.28
2021-06-01	100.25	100.29
2021-07-01	99.71	102.02
2021-08-01	99.95	100.34
2021-09-01	100.95	100.73
2021-10-01	101.06	102.78
2021-11-01	100.73	99.47
2021-12-01	100.84	100.74
2022-01-01	101.15	101.73
2022-02-01	101.96	99.70
2022-03-01	102.54	101.86
2022-04-01	102.56	105.45
2022-05-01	101.73	101.00
2022-06-01	100.62	100.92
2022-07-01	100.92	100.87
2022-08-01	101.82	99.81
2022-09-01	101.72	101.28
2022-10-01	102.29	100.80
2022-11-01	101.54	101.03
2022-12-01	101.26	99.68

month_id	IPC Servicii (%)
2019-01-01	100.57
2019-02-01	100.55
2019-03-01	100.40
2019-04-01	100.72
2019-05-01	100.55
2019-06-01	100.17
2019-07-01	100.10
2019-08-01	100.25
2019-09-01	100.27

```
2019-10-01            100.25
2019-11-01            100.15
2019-12-01            100.12
2020-01-01            100.43
2020-02-01            100.39
2020-03-01            100.35
2020-04-01            100.00
2020-05-01            100.11
2020-06-01            100.34
2020-07-01            100.31
2020-08-01            100.21
2020-09-01            100.20
2020-10-01            100.22
2020-11-01            100.07
2020-12-01            100.04
2021-01-01            100.25
2021-02-01            100.20
2021-03-01            100.24
2021-04-01            100.40
2021-05-01            100.27
2021-06-01            100.23
2021-07-01            100.39
2021-08-01            100.43
2021-09-01            100.96
2021-10-01            100.42
2021-11-01            100.20
2021-12-01            100.42
2022-01-01            101.37
2022-02-01            100.60
2022-03-01            100.67
2022-04-01            100.94
2022-05-01            100.61
2022-06-01            100.54
2022-07-01            100.88
2022-08-01            100.37
2022-09-01            100.72
2022-10-01            100.70
2022-11-01            101.31
2022-12-01            100.67
```

In [95]:

```
# Line graph for apa_plata for all stores
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index.get_level_values(0), filter_df_sales['sales'])
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales for ' + product + ' for all stores by Month')
plt.show()
```

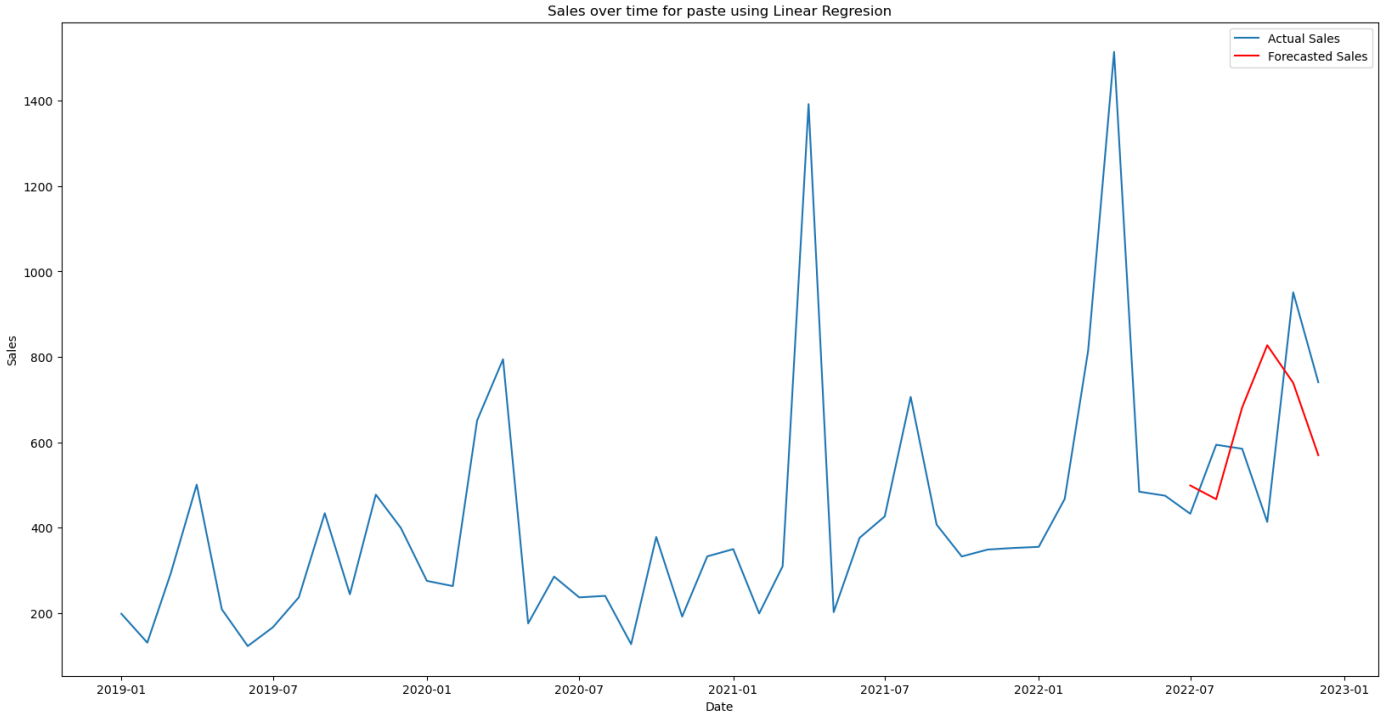


LR - paste

In [96]:
```
# Get the last 6 months of data
last_6_months = filter_df_sales.tail(6)

# Split the data into train and test sets
X_train = filter_df_sales.drop('sales', axis=1)
y_train = filter_df_sales['sales']
X_test = last_6_months.drop('sales', axis=1)
y_test = last_6_months['sales']

# Create a linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the error
lr_mae = mean_absolute_error(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)

print("Mean Absolute Error: ", lr_mae)
print("Mean Squared Error: ", lr_mse)
print("Root Mean Squared Error:", lr_rmse)

all_sales = filter_df_sales['sales']

# Get the index of the last 6 months of data
last_6_months_index = last_6_months.index

# Plot the entire period of sales data
plt.figure(figsize=(20,10))
plt.plot(all_sales)

# Make predictions on the last 6 months of data
```

```
y_pred = model.predict(X_test)

# Plot the forecasted sales on the same chart
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Linear Regresion')
plt.legend(['Actual Sales', 'Forecasted Sales'])

plt.show()
```

```
Mean Absolute Error:  181.28857145203355
Mean Squared Error:  45954.816230733144
Root Mean Squared Error: 214.37074481079068
```



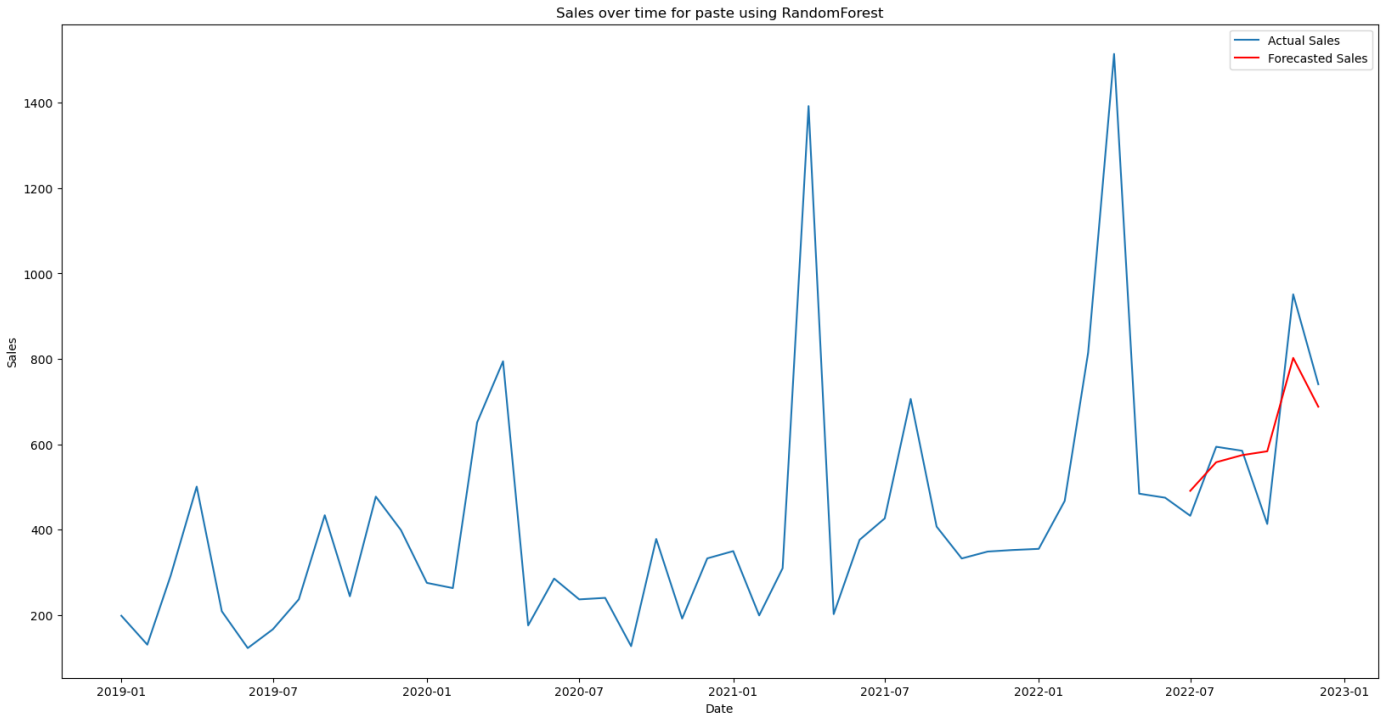
RF - paste

In [97]:

```
last_6_months = filter_df_sales.tail(6)

X_train = filter_df_sales.drop(columns=["sales"])
y_train = filter_df_sales["sales"]
X_test = last_6_months.drop(columns=["sales"])
y_test = last_6_months["sales"]

# Create a random forest model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Use the model to make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate the model's accuracy
accuracy = rf.score(X_test, y_test)
print("Accuracy: ", accuracy)

# Calculate error
rf_mae = mean_absolute_error(y_test, y_pred)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rf_mse)
```

```
print("Mean Absolute Error: ", rf_mae)
print("Mean Squared Error: ", rf_mse)
print("Root Mean Squared Error:", rf_rmse)

# Line graph for the forcast
plt.figure(figsize=(20,10))

plt.plot(all_sales)
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using RandomForest')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
Accuracy:  0.71202839040474
Mean Absolute Error:  79.45993333333334
Mean Squared Error:  9798.345173520027
Root Mean Squared Error: 98.98659087735079
```



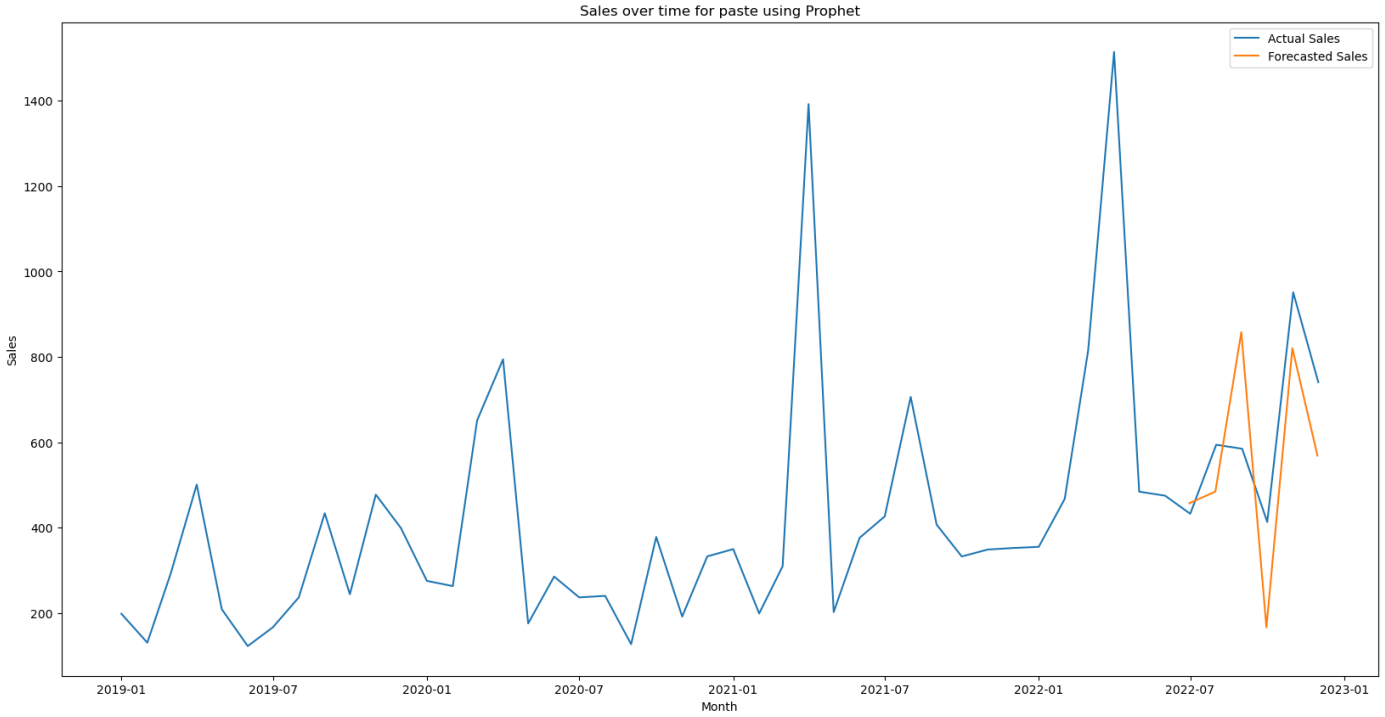
PR - paste

In [98]:

```
# create a copy of the dataframe with a column named 'ds' with the index values
filter_df_sales_prophet = filter_df_sales.reset_index().rename(columns={'month_id': 'ds'
filter_df_sales_prophet = filter_df_sales_prophet.rename(columns={'sales':'y'})

train_data = filter_df_sales_prophet[:-6]
test_data = filter_df_sales_prophet[-6:]

model = Prophet()

# Fit the model on the data
model.fit(train_data)

# Create a dataframe to hold the forecast
future_data = model.make_future_dataframe(periods=6, freq='m')

# Make predictions
forecast = model.predict(future_data)
forecast = forecast.tail(6)
```

```
# Calculate error
pr_mae = mean_absolute_error(test_data['y'], forecast[-6:]['yhat'])
pr_mse = mean_squared_error(test_data['y'], forecast[-6:]['yhat'])
pr_rmse = np.sqrt(pr_mse)
print("Mean Absolute Error:", pr_mae)
print("Mean Squared Error:", pr_mse)
print("Root Mean Squared Error:", pr_rmse)

# Plot the forecast
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index, filter_df_sales['sales'], label='Actual Sales')

# Plot the predicted values for the last six months
plt.plot(forecast.ds, forecast.yhat, label='Predicted Sales')

plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Prophet')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
21:35:23 - cmdstanpy - INFO - Chain [1] start processing
21:35:23 - cmdstanpy - INFO - Chain [1] done processing
Mean Absolute Error: 159.51180569449429
Mean Squared Error: 32467.133732709073
Root Mean Squared Error: 180.18638609148326
```



In [99]:

```
# Add the errors from all methods to the DataFrame
new_row_df =  pd.DataFrame({'product': [product], 'LR-MAE': [lr_mae], 'LR-MSE': [lr_mse]
errors_df = pd.concat([errors_df, new_row_df], ignore_index=True)
print(errors_df)
```

```
           product      LR-MAE         LR-MSE     LR-RMSE      RF-MAE  \
0       apa_plata  887.015829  988186.848307  994.075877  377.426267
1    apa_minerala   44.324535    2453.366899   49.531474   10.533100
2            bere  283.109792   95844.002658  309.586826   65.444150
3           cafea  287.818261   94385.213013  307.221765  134.035500
4         carnati  264.936590   99377.947534  315.242680  155.934000
5        frigider   13.054626     245.441803   15.666582    8.523900
6           lapte  490.769164  269944.557868  519.561890  193.309633
7          legume  317.334317  168366.068324  410.324345  124.748417
```

```
8   masina_de_spalat   18.592357     606.867326   24.634677    8.862467
9               orez  376.601361  165639.594918  406.988446   98.474217
10               oua  725.835859  635033.703803  796.890020  307.595917
11             paste  181.288571   45954.816231  214.370745   79.459933

           RF-MSE     RF-RMSE        PR-MAE        PR-MSE      PR-RMSE
0   200759.770117  448.062239  1107.910649  1.661801e+06  1289.108561
1      153.080170   12.372557  1703.258881  1.172504e+07  3424.185038
2     6221.823280   78.878535   437.297885  2.623325e+05   512.184061
3    22921.110989  151.397196   554.567175  3.976641e+05   630.606141
4    33523.190407  183.093393   370.439785  1.702760e+05   412.645094
5      105.584959   10.275454    34.075515  1.834104e+03    42.826436
6    48061.717358  219.229828   968.181840  1.101085e+06  1049.325744
7    23345.264739  152.791573   605.913460  4.861077e+05   697.214275
8      139.974418   11.831078    40.934713  2.225668e+03    47.176986
9    17005.647771  130.405705   562.437890  5.253923e+05   724.839518
10  114084.843273  337.764479  1055.734550  1.583208e+06  1258.255738
11    9798.345174   98.986591   159.511806  3.246713e+04   180.186386
```

Peste DataFrame

In [100…

```
# Filter the initial DataFrame
product = 'peste'
filter_df = df.loc[df['denumire_produs'] == product]
filter_df = filter_df.drop(columns=['colli', 'store_county'])
#print(apa_plata_df)

# Group by month_id and aggregate the other columns
filter_df_sales = filter_df.groupby("month_id").agg({'sales': 'sum','Inflatie anuala': '
filter_df_sales = filter_df_sales.sort_values(by='month_id', ascending=True)

# Print the apa_plata_sales dataframe
print(filter_df_sales)
```

```
              sales  Inflatie anuala  unnemployment  \
month_id
2019-01-01   513.06              3.8           1.42
2019-02-01   497.41              3.8           3.32
2019-03-01   574.26              3.8           3.31
2019-04-01   780.17              3.8           3.19
2019-05-01   472.98              3.8           3.00
2019-06-01   559.79              3.8           2.95
2019-07-01   316.60              3.8           2.95
2019-08-01   718.95              3.8           3.01
2019-09-01   659.42              3.8           3.03
2019-10-01   708.27              3.8           3.00
2019-11-01  1055.53              3.8           2.98
2019-12-01   551.47              3.8           2.98
2020-01-01   636.55              2.6           2.97
2020-02-01   740.23              2.6           2.98
2020-03-01   772.83              2.6           2.95
2020-04-01   835.73              2.6           2.88
2020-05-01   822.16              2.6           2.90
2020-06-01   904.78              2.6           2.87
2020-07-01  1116.61              2.6           3.00
2020-08-01   879.58              2.6           3.02
2020-09-01   839.32              2.6           3.30
2020-10-01  1145.84              2.6           3.26
2020-11-01  1019.95              2.6           3.27
2020-12-01   866.41              2.6           3.32
2021-01-01   884.23              5.1           3.38
2021-02-01  1053.90              5.1           3.34
2021-03-01   812.46              5.1           3.35
2021-04-01  1045.18              5.1           3.33
```

2021-05-01	857.60	5.1	3.16
2021-06-01	886.35	5.1	3.06
2021-07-01	420.08	5.1	3.00
2021-08-01	934.87	5.1	2.96
2021-09-01	1193.78	5.1	2.93
2021-10-01	1300.40	5.1	2.85
2021-11-01	1703.46	5.1	2.76
2021-12-01	1384.59	5.1	2.72
2022-01-01	1101.81	13.8	2.69
2022-02-01	1366.47	13.8	2.68
2022-03-01	1546.75	13.8	2.67
2022-04-01	1327.34	13.8	2.64
2022-05-01	1476.41	13.8	2.57
2022-06-01	750.05	13.8	2.55
2022-07-01	1165.06	13.8	2.55
2022-08-01	827.77	13.8	2.56
2022-09-01	1324.00	13.8	2.56
2022-10-01	927.48	13.8	2.88
2022-11-01	1108.39	13.8	2.96
2022-12-01	1311.05	13.8	3.04

month_id	Inflatie(de la o luna la alta)	net_average_salary
2019-01-01	0.83	2936.0
2019-02-01	0.79	2933.0
2019-03-01	0.49	3075.0
2019-04-01	0.61	3115.0
2019-05-01	0.46	3101.0
2019-06-01	-0.23	3142.0
2019-07-01	-0.20	3119.0
2019-08-01	0.06	3044.0
2019-09-01	0.09	3082.0
2019-10-01	0.43	3116.0
2019-11-01	0.23	3179.0
2019-12-01	0.42	3340.0
2020-01-01	0.41	3189.0
2020-02-01	0.25	3202.0
2020-03-01	0.50	3294.0
2020-04-01	0.26	3182.0
2020-05-01	0.05	3179.0
2020-06-01	0.08	3298.0
2020-07-01	0.00	3372.0
2020-08-01	-0.05	3275.0
2020-09-01	-0.14	3321.0
2020-10-01	0.22	3343.0
2020-11-01	0.13	3411.0
2020-12-01	0.34	3620.0
2021-01-01	1.33	3395.0
2021-02-01	0.41	3365.0
2021-03-01	0.38	3547.0
2021-04-01	0.45	3561.0
2021-05-01	0.53	3492.0
2021-06-01	0.27	3541.0
2021-07-01	0.97	3545.0
2021-08-01	0.24	3487.0
2021-09-01	0.84	3517.0
2021-10-01	1.78	3544.0
2021-11-01	0.00	3645.0
2021-12-01	0.71	3879.0
2022-01-01	1.48	3698.0
2022-02-01	0.58	3721.0
2022-03-01	1.88	3937.0
2022-04-01	3.74	3967.0
2022-05-01	1.18	3928.0
2022-06-01	0.76	3977.0
2022-07-01	0.89	3975.0

month_id		
2022-08-01	0.56	3933.0
2022-09-01	1.33	4003.0
2022-10-01	1.28	4008.0
2022-11-01	1.25	4141.0
2022-12-01	0.37	4141.0

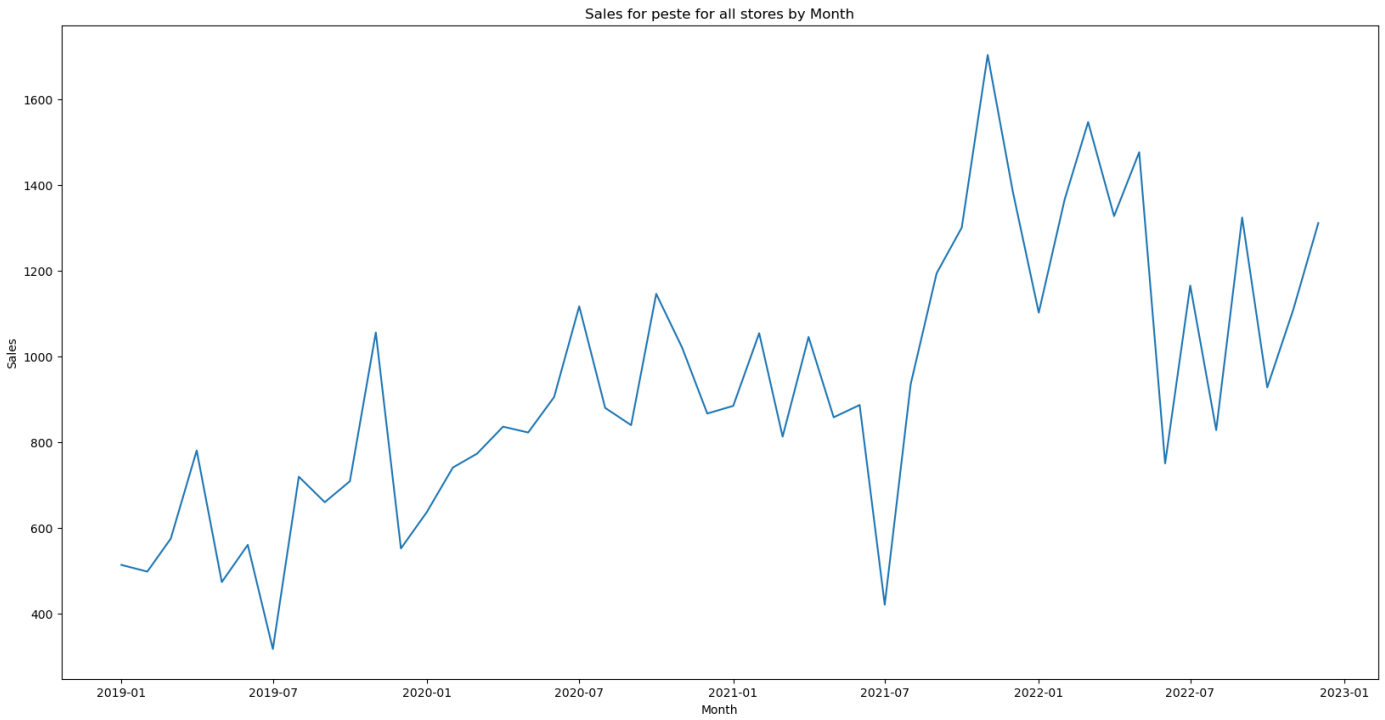
month_id	IPC Marfuri alimentare (%)	IPC Marfuri nealimentare (%)	\
2019-01-01	101.30	100.63	
2019-02-01	101.27	100.57	
2019-03-01	100.48	100.54	
2019-04-01	100.68	100.53	
2019-05-01	100.79	100.20	
2019-06-01	99.53	99.76	
2019-07-01	99.34	99.98	
2019-08-01	99.71	100.22	
2019-09-01	99.91	100.13	
2019-10-01	100.70	100.32	
2019-11-01	100.43	100.12	
2019-12-01	100.84	100.28	
2020-01-01	100.99	100.02	
2020-02-01	100.63	99.94	
2020-03-01	101.46	99.91	
2020-04-01	101.27	99.67	
2020-05-01	100.34	99.82	
2020-06-01	99.62	100.28	
2020-07-01	99.55	100.19	
2020-08-01	99.59	100.08	
2020-09-01	99.45	99.99	
2020-10-01	100.11	100.31	
2020-11-01	99.92	100.29	
2020-12-01	100.29	100.51	
2021-01-01	100.63	102.24	
2021-02-01	100.46	100.47	
2021-03-01	100.37	100.46	
2021-04-01	100.45	100.47	
2021-05-01	101.10	100.28	
2021-06-01	100.25	100.29	
2021-07-01	99.71	102.02	
2021-08-01	99.95	100.34	
2021-09-01	100.95	100.73	
2021-10-01	101.06	102.78	
2021-11-01	100.73	99.47	
2021-12-01	100.84	100.74	
2022-01-01	101.15	101.73	
2022-02-01	101.96	99.70	
2022-03-01	102.54	101.86	
2022-04-01	102.56	105.45	
2022-05-01	101.73	101.00	
2022-06-01	100.62	100.92	
2022-07-01	100.92	100.87	
2022-08-01	101.82	99.81	
2022-09-01	101.72	101.28	
2022-10-01	102.29	100.80	
2022-11-01	101.54	101.03	
2022-12-01	101.26	99.68	

month_id	IPC Servicii (%)
2019-01-01	100.57
2019-02-01	100.55
2019-03-01	100.40
2019-04-01	100.72
2019-05-01	100.55
2019-06-01	100.17
2019-07-01	100.10

```
2019-08-01            100.25
2019-09-01            100.27
2019-10-01            100.25
2019-11-01            100.15
2019-12-01            100.12
2020-01-01            100.43
2020-02-01            100.39
2020-03-01            100.35
2020-04-01            100.00
2020-05-01            100.11
2020-06-01            100.34
2020-07-01            100.31
2020-08-01            100.21
2020-09-01            100.20
2020-10-01            100.22
2020-11-01            100.07
2020-12-01            100.04
2021-01-01            100.25
2021-02-01            100.20
2021-03-01            100.24
2021-04-01            100.40
2021-05-01            100.27
2021-06-01            100.23
2021-07-01            100.39
2021-08-01            100.43
2021-09-01            100.96
2021-10-01            100.42
2021-11-01            100.20
2021-12-01            100.42
2022-01-01            101.37
2022-02-01            100.60
2022-03-01            100.67
2022-04-01            100.94
2022-05-01            100.61
2022-06-01            100.54
2022-07-01            100.88
2022-08-01            100.37
2022-09-01            100.72
2022-10-01            100.70
2022-11-01            101.31
2022-12-01            100.67
```

In [101…

```
# Line graph for apa_plata for all stores
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index.get_level_values(0), filter_df_sales['sales'])
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales for ' + product + ' for all stores by Month')
plt.show()
```



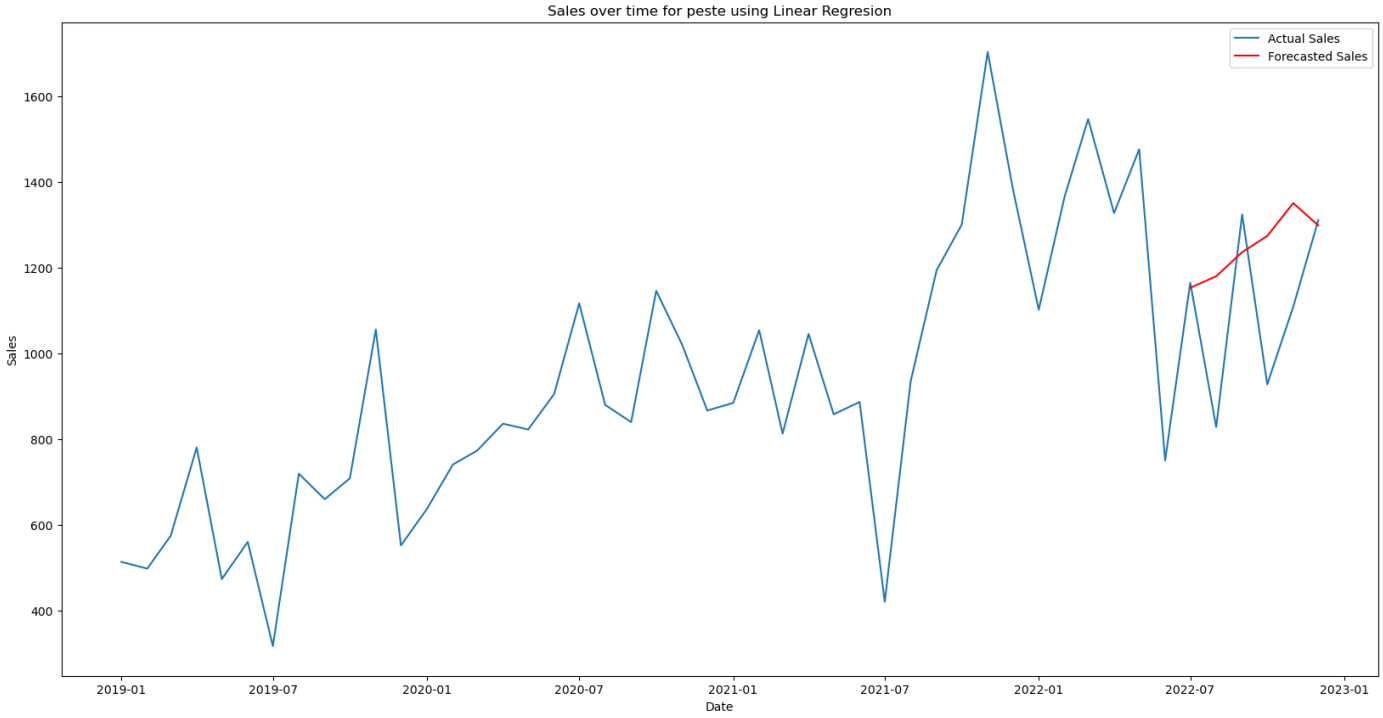
LR - peste

In [102…

```
# Get the last 6 months of data
last_6_months = filter_df_sales.tail(6)

# Split the data into train and test sets
X_train = filter_df_sales.drop('sales', axis=1)
y_train = filter_df_sales['sales']
X_test = last_6_months.drop('sales', axis=1)
y_test = last_6_months['sales']

# Create a linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the error
lr_mae = mean_absolute_error(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)

print("Mean Absolute Error: ", lr_mae)
print("Mean Squared Error: ", lr_mse)
print("Root Mean Squared Error:", lr_rmse)

all_sales = filter_df_sales['sales']

# Get the index of the last 6 months of data
last_6_months_index = last_6_months.index

# Plot the entire period of sales data
plt.figure(figsize=(20,10))
plt.plot(all_sales)

# Make predictions on the last 6 months of data
```

```
y_pred = model.predict(X_test)

# Plot the forecasted sales on the same chart
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Linear Regresion')
plt.legend(['Actual Sales', 'Forecasted Sales'])

plt.show()
```

```
Mean Absolute Error:  175.54861912148024
Mean Squared Error:  51772.45432961764
Root Mean Squared Error: 227.53561112409997
```



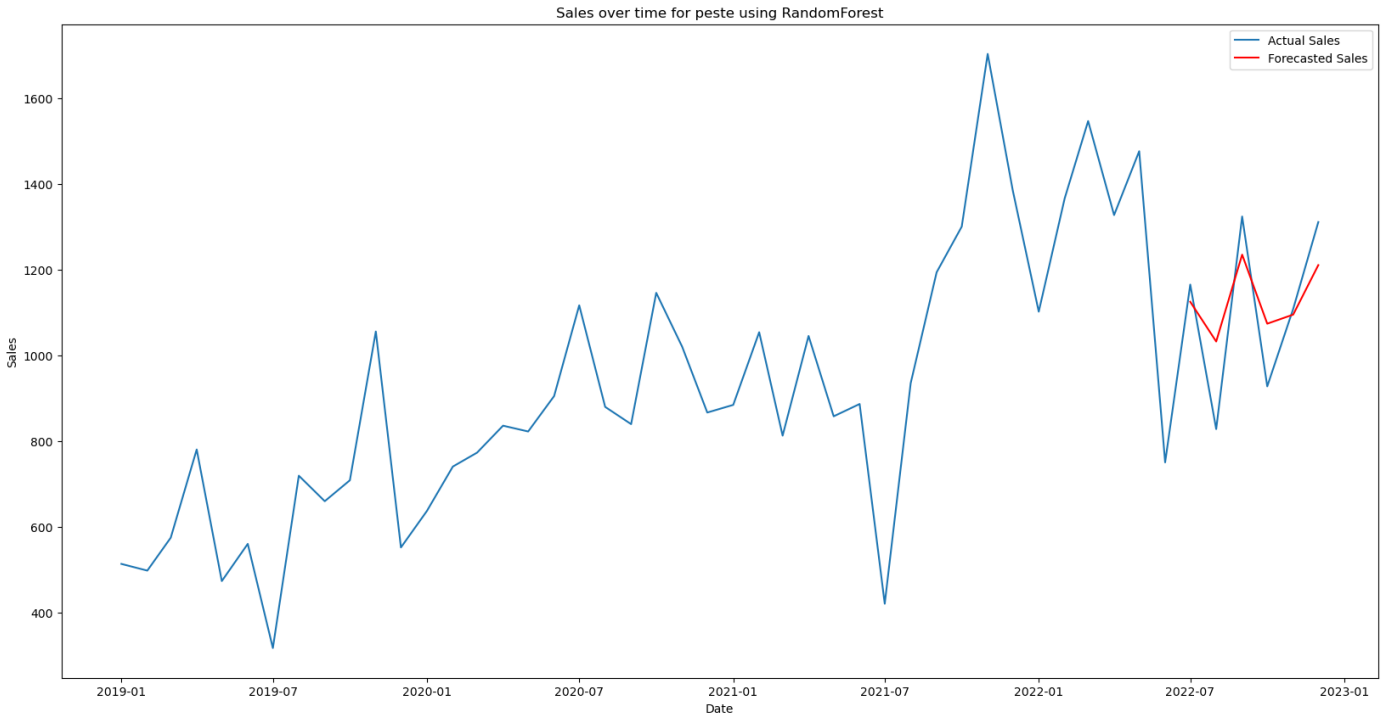
RF - peste

In [103…

```
last_6_months = filter_df_sales.tail(6)

X_train = filter_df_sales.drop(columns=["sales"])
y_train = filter_df_sales["sales"]
X_test = last_6_months.drop(columns=["sales"])
y_test = last_6_months["sales"]

# Create a random forest model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Use the model to make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate the model's accuracy
accuracy = rf.score(X_test, y_test)
print("Accuracy: ", accuracy)

# Calculate error
rf_mae = mean_absolute_error(y_test, y_pred)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rf_mse)
```

```
print("Mean Absolute Error: ", rf_mae)
print("Mean Squared Error: ", rf_mse)
print("Root Mean Squared Error:", rf_rmse)

# Line graph for the forcast
plt.figure(figsize=(20,10))

plt.plot(all_sales)
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using RandomForest')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
Accuracy:  0.5894687074989444
Mean Absolute Error:  98.96516666666662
Mean Squared Error:  13836.018311576656
Root Mean Squared Error: 117.62660545801981
```



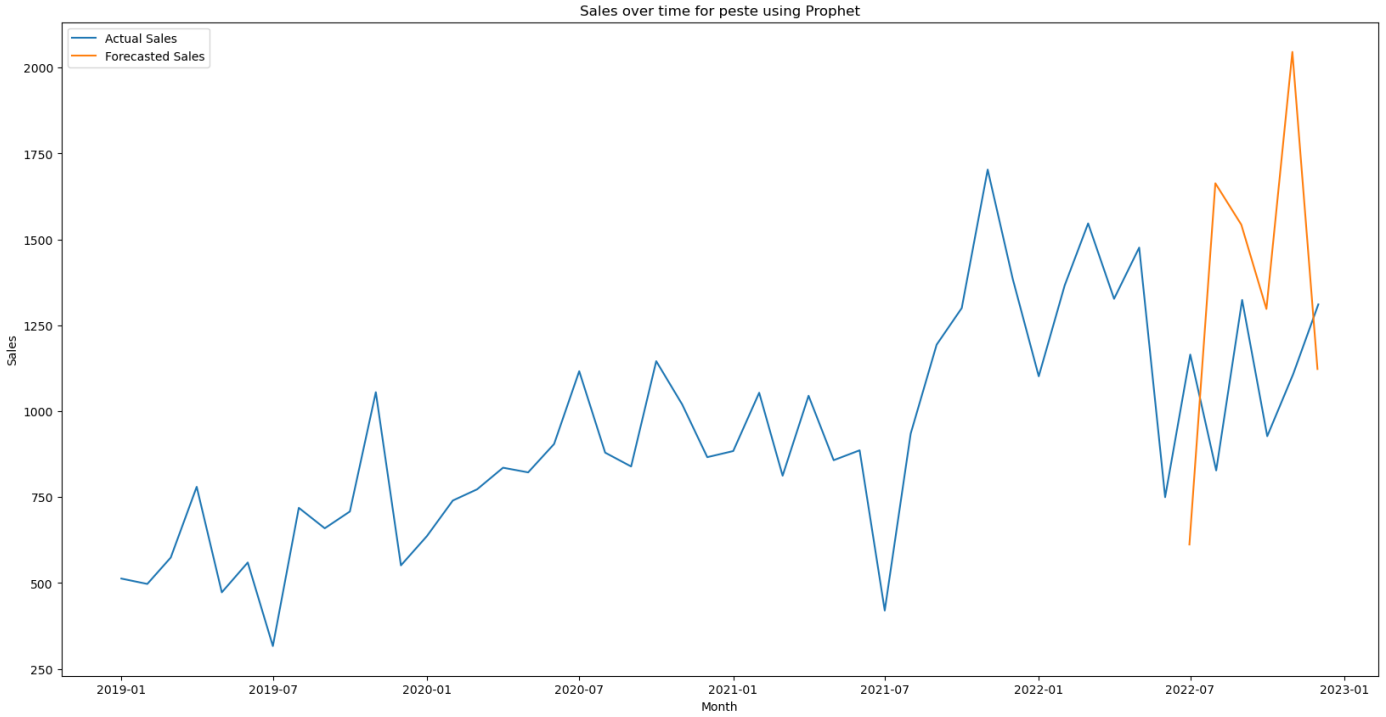
PR - peste

In [104…

```
# create a copy of the dataframe with a column named 'ds' with the index values
filter_df_sales_prophet = filter_df_sales.reset_index().rename(columns={'month_id': 'ds'
filter_df_sales_prophet = filter_df_sales_prophet.rename(columns={'sales':'y'})

train_data = filter_df_sales_prophet[:-6]
test_data = filter_df_sales_prophet[-6:]

model = Prophet()

# Fit the model on the data
model.fit(train_data)

# Create a dataframe to hold the forecast
future_data = model.make_future_dataframe(periods=6, freq='m')

# Make predictions
forecast = model.predict(future_data)
forecast = forecast.tail(6)
```

```
# Calculate error
pr_mae = mean_absolute_error(test_data['y'], forecast[-6:]['yhat'])
pr_mse = mean_squared_error(test_data['y'], forecast[-6:]['yhat'])
pr_rmse = np.sqrt(pr_mse)
print("Mean Absolute Error:", pr_mae)
print("Mean Squared Error:", pr_mse)
print("Root Mean Squared Error:", pr_rmse)

# Plot the forecast
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index, filter_df_sales['sales'], label='Actual Sales')

# Plot the predicted values for the last six months
plt.plot(forecast.ds, forecast.yhat, label='Predicted Sales')

plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Prophet')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
21:35:24 - cmdstanpy - INFO - Chain [1] start processing
21:35:24 - cmdstanpy - INFO - Chain [1] done processing
Mean Absolute Error: 517.2363324155657
Mean Squared Error: 350568.17088977277
Root Mean Squared Error: 592.0879756334972
```



In [105…

```
# Add the errors from all methods to the DataFrame
new_row_df =  pd.DataFrame({'product': [product], 'LR-MAE': [lr_mae], 'LR-MSE': [lr_mse]
errors_df = pd.concat([errors_df, new_row_df], ignore_index=True)
print(errors_df)
```

	product	LR-MAE	LR-MSE	LR-RMSE	RF-MAE	\
0	apa_plata	887.015829	988186.848307	994.075877	377.426267	
1	apa_minerala	44.324535	2453.366899	49.531474	10.533100	
2	bere	283.109792	95844.002658	309.586826	65.444150	
3	cafea	287.818261	94385.213013	307.221765	134.035500	
4	carnati	264.936590	99377.947534	315.242680	155.934000	
5	frigider	13.054626	245.441803	15.666582	8.523900	
6	lapte	490.769164	269944.557868	519.561890	193.309633	
7	legume	317.334317	168366.068324	410.324345	124.748417	

8	masina_de_spalat	18.592357	606.867326	24.634677	8.862467
9	orez	376.601361	165639.594918	406.988446	98.474217
10	oua	725.835859	635033.703803	796.890020	307.595917
11	paste	181.288571	45954.816231	214.370745	79.459933
12	peste	175.548619	51772.454330	227.535611	98.965167

	RF-MSE	RF-RMSE	PR-MAE	PR-MSE	PR-RMSE
0	200759.770117	448.062239	1107.910649	1.661801e+06	1289.108561
1	153.080170	12.372557	1703.258881	1.172504e+07	3424.185038
2	6221.823280	78.878535	437.297885	2.623325e+05	512.184061
3	22921.110989	151.397196	554.567175	3.976641e+05	630.606141
4	33523.190407	183.093393	370.439785	1.702760e+05	412.645094
5	105.584959	10.275454	34.075515	1.834104e+03	42.826436
6	48061.717358	219.229828	968.181840	1.101085e+06	1049.325744
7	23345.264739	152.791573	605.913460	4.861077e+05	697.214275
8	139.974418	11.831078	40.934713	2.225668e+03	47.176986
9	17005.647771	130.405705	562.437890	5.253923e+05	724.839518
10	114084.843273	337.764479	1055.734550	1.583208e+06	1258.255738
11	9798.345174	98.986591	159.511806	3.246713e+04	180.186386
12	13836.018312	117.626605	517.236332	3.505682e+05	592.087976

Pizza DataFrame

In [106…

```
# Filter the initial DataFrame
product = 'pizza'
filter_df = df.loc[df['denumire_produs'] == product]
filter_df = filter_df.drop(columns=['colli', 'store_county'])
#print(apa_plata_df)

# Group by month_id and aggregate the other columns
filter_df_sales = filter_df.groupby("month_id").agg({'sales': 'sum','Inflatie anuala': '
filter_df_sales = filter_df_sales.sort_values(by='month_id', ascending=True)

# Print the apa_plata_sales dataframe
print(filter_df_sales)
```

month_id	sales	Inflatie anuala	unnemployment	\
2019-01-01	2474.98	3.8	1.42	
2019-02-01	1952.60	3.8	3.32	
2019-03-01	2359.51	3.8	3.31	
2019-04-01	2227.08	3.8	3.19	
2019-05-01	2730.53	3.8	3.00	
2019-06-01	2258.80	3.8	2.95	
2019-07-01	2337.19	3.8	2.95	
2019-08-01	2115.40	3.8	3.01	
2019-09-01	2679.56	3.8	3.03	
2019-10-01	1633.75	3.8	3.00	
2019-11-01	2265.34	3.8	2.98	
2019-12-01	1667.71	3.8	2.98	
2020-01-01	1937.38	2.6	2.97	
2020-02-01	1710.87	2.6	2.98	
2020-03-01	3328.27	2.6	2.95	
2020-04-01	1743.00	2.6	2.88	
2020-05-01	2559.57	2.6	2.90	
2020-06-01	2353.21	2.6	2.87	
2020-07-01	2616.40	2.6	3.00	
2020-08-01	2129.87	2.6	3.02	
2020-09-01	2590.09	2.6	3.30	
2020-10-01	1734.65	2.6	3.26	
2020-11-01	1962.38	2.6	3.27	
2020-12-01	1956.31	2.6	3.32	
2021-01-01	1579.70	5.1	3.38	
2021-02-01	2027.92	5.1	3.34	

2021-03-01	2812.76	5.1	3.35
2021-04-01	2315.60	5.1	3.33
2021-05-01	2543.28	5.1	3.16
2021-06-01	3001.72	5.1	3.06
2021-07-01	2058.34	5.1	3.00
2021-08-01	2009.36	5.1	2.96
2021-09-01	2975.83	5.1	2.93
2021-10-01	2435.71	5.1	2.85
2021-11-01	2779.48	5.1	2.76
2021-12-01	2339.94	5.1	2.72
2022-01-01	2623.29	13.8	2.69
2022-02-01	2901.40	13.8	2.68
2022-03-01	2995.58	13.8	2.67
2022-04-01	3605.17	13.8	2.64
2022-05-01	2833.75	13.8	2.57
2022-06-01	3076.14	13.8	2.55
2022-07-01	2602.61	13.8	2.55
2022-08-01	3726.59	13.8	2.56
2022-09-01	3373.73	13.8	2.56
2022-10-01	3290.23	13.8	2.88
2022-11-01	3063.59	13.8	2.96
2022-12-01	2520.87	13.8	3.04

month_id	Inflatie(de la o luna la alta)	net_average_salary
2019-01-01	0.83	2936.0
2019-02-01	0.79	2933.0
2019-03-01	0.49	3075.0
2019-04-01	0.61	3115.0
2019-05-01	0.46	3101.0
2019-06-01	-0.23	3142.0
2019-07-01	-0.20	3119.0
2019-08-01	0.06	3044.0
2019-09-01	0.09	3082.0
2019-10-01	0.43	3116.0
2019-11-01	0.23	3179.0
2019-12-01	0.42	3340.0
2020-01-01	0.41	3189.0
2020-02-01	0.25	3202.0
2020-03-01	0.50	3294.0
2020-04-01	0.26	3182.0
2020-05-01	0.05	3179.0
2020-06-01	0.08	3298.0
2020-07-01	0.00	3372.0
2020-08-01	-0.05	3275.0
2020-09-01	-0.14	3321.0
2020-10-01	0.22	3343.0
2020-11-01	0.13	3411.0
2020-12-01	0.34	3620.0
2021-01-01	1.33	3395.0
2021-02-01	0.41	3365.0
2021-03-01	0.38	3547.0
2021-04-01	0.45	3561.0
2021-05-01	0.53	3492.0
2021-06-01	0.27	3541.0
2021-07-01	0.97	3545.0
2021-08-01	0.24	3487.0
2021-09-01	0.84	3517.0
2021-10-01	1.78	3544.0
2021-11-01	0.00	3645.0
2021-12-01	0.71	3879.0
2022-01-01	1.48	3698.0
2022-02-01	0.58	3721.0
2022-03-01	1.88	3937.0
2022-04-01	3.74	3967.0
2022-05-01	1.18	3928.0

month_id		
2022-06-01	0.76	3977.0
2022-07-01	0.89	3975.0
2022-08-01	0.56	3933.0
2022-09-01	1.33	4003.0
2022-10-01	1.28	4008.0
2022-11-01	1.25	4141.0
2022-12-01	0.37	4141.0

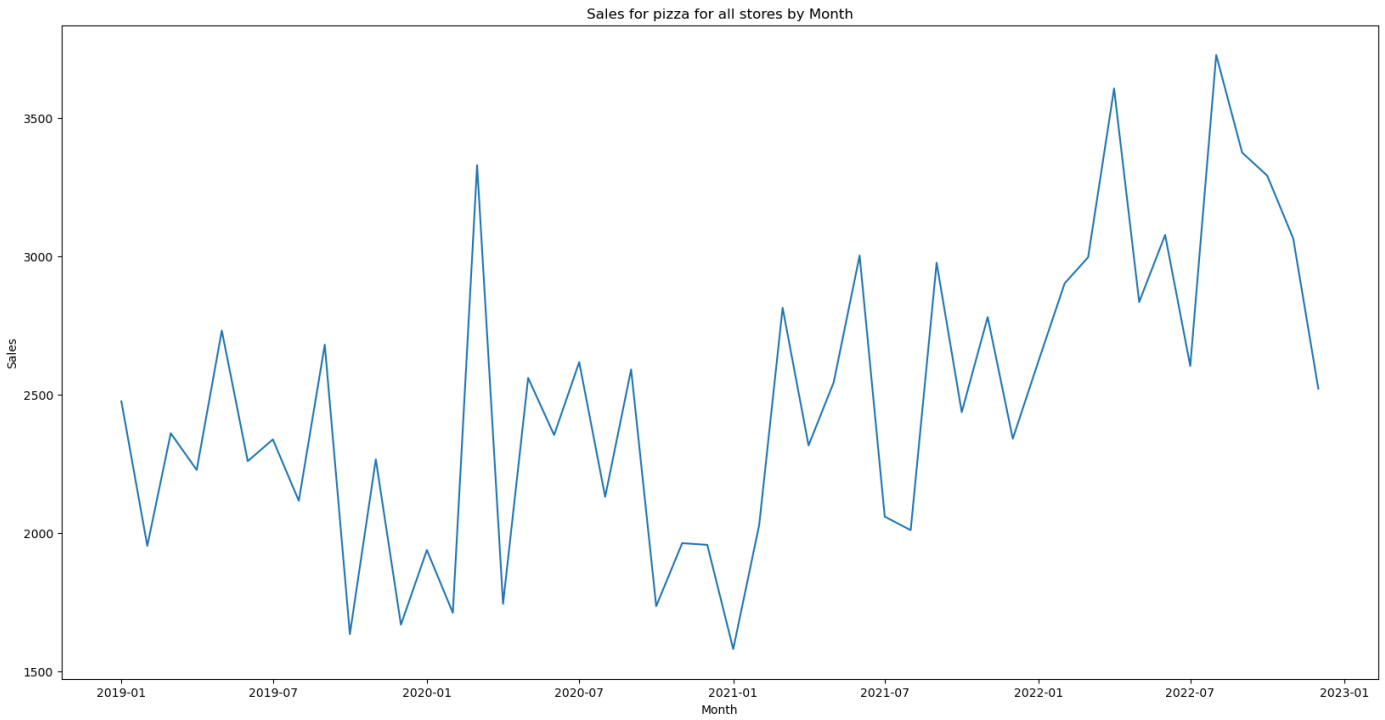
month_id	IPC Marfuri alimentare (%)	IPC Marfuri nealimentare (%)
2019-01-01	101.30	100.63
2019-02-01	101.27	100.57
2019-03-01	100.48	100.54
2019-04-01	100.68	100.53
2019-05-01	100.79	100.20
2019-06-01	99.53	99.76
2019-07-01	99.34	99.98
2019-08-01	99.71	100.22
2019-09-01	99.91	100.13
2019-10-01	100.70	100.32
2019-11-01	100.43	100.12
2019-12-01	100.84	100.28
2020-01-01	100.99	100.02
2020-02-01	100.63	99.94
2020-03-01	101.46	99.91
2020-04-01	101.27	99.67
2020-05-01	100.34	99.82
2020-06-01	99.62	100.28
2020-07-01	99.55	100.19
2020-08-01	99.59	100.08
2020-09-01	99.45	99.99
2020-10-01	100.11	100.31
2020-11-01	99.92	100.29
2020-12-01	100.29	100.51
2021-01-01	100.63	102.24
2021-02-01	100.46	100.47
2021-03-01	100.37	100.46
2021-04-01	100.45	100.47
2021-05-01	101.10	100.28
2021-06-01	100.25	100.29
2021-07-01	99.71	102.02
2021-08-01	99.95	100.34
2021-09-01	100.95	100.73
2021-10-01	101.06	102.78
2021-11-01	100.73	99.47
2021-12-01	100.84	100.74
2022-01-01	101.15	101.73
2022-02-01	101.96	99.70
2022-03-01	102.54	101.86
2022-04-01	102.56	105.45
2022-05-01	101.73	101.00
2022-06-01	100.62	100.92
2022-07-01	100.92	100.87
2022-08-01	101.82	99.81
2022-09-01	101.72	101.28
2022-10-01	102.29	100.80
2022-11-01	101.54	101.03
2022-12-01	101.26	99.68

month_id	IPC Servicii (%)
2019-01-01	100.57
2019-02-01	100.55
2019-03-01	100.40
2019-04-01	100.72
2019-05-01	100.55

```
2019-06-01            100.17
2019-07-01            100.10
2019-08-01            100.25
2019-09-01            100.27
2019-10-01            100.25
2019-11-01            100.15
2019-12-01            100.12
2020-01-01            100.43
2020-02-01            100.39
2020-03-01            100.35
2020-04-01            100.00
2020-05-01            100.11
2020-06-01            100.34
2020-07-01            100.31
2020-08-01            100.21
2020-09-01            100.20
2020-10-01            100.22
2020-11-01            100.07
2020-12-01            100.04
2021-01-01            100.25
2021-02-01            100.20
2021-03-01            100.24
2021-04-01            100.40
2021-05-01            100.27
2021-06-01            100.23
2021-07-01            100.39
2021-08-01            100.43
2021-09-01            100.96
2021-10-01            100.42
2021-11-01            100.20
2021-12-01            100.42
2022-01-01            101.37
2022-02-01            100.60
2022-03-01            100.67
2022-04-01            100.94
2022-05-01            100.61
2022-06-01            100.54
2022-07-01            100.88
2022-08-01            100.37
2022-09-01            100.72
2022-10-01            100.70
2022-11-01            101.31
2022-12-01            100.67
```

In [107…

```
# Line graph for apa_plata for all stores
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index.get_level_values(0), filter_df_sales['sales'])
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales for ' + product + ' for all stores by Month')
plt.show()
```



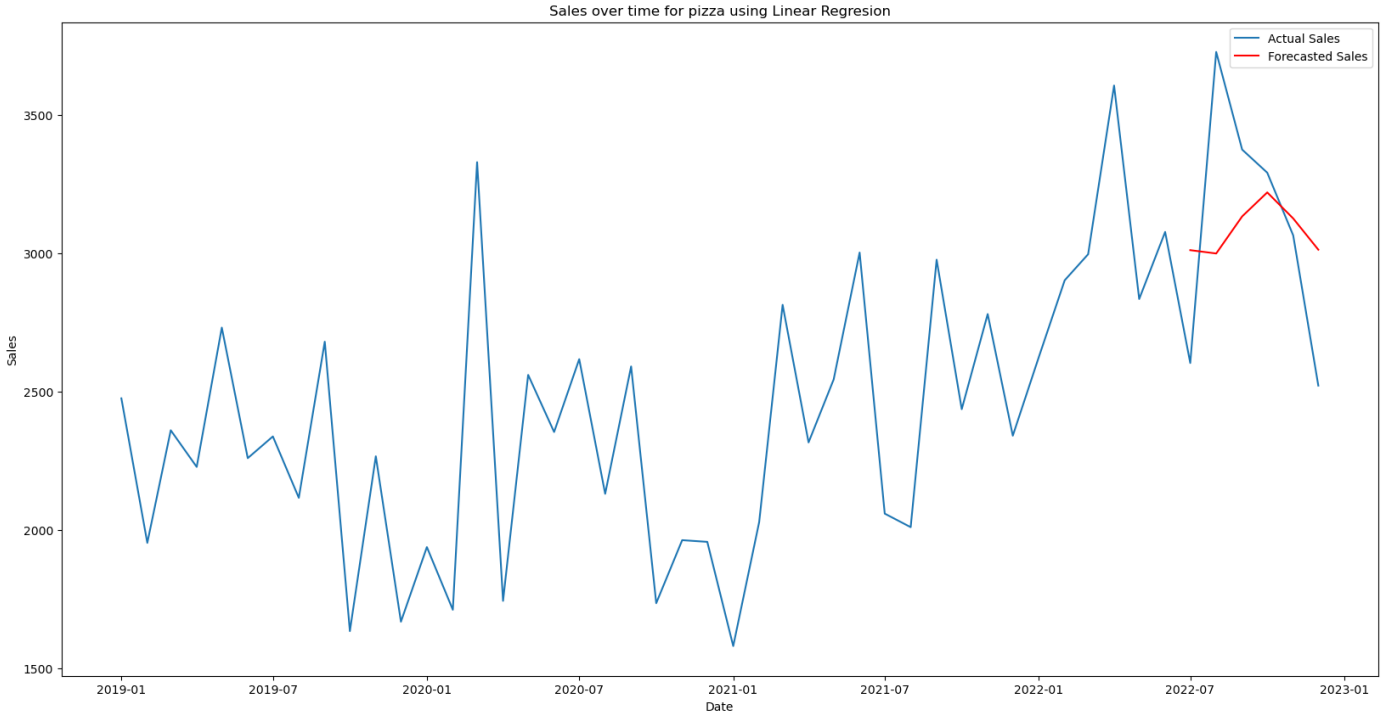
LR - pizza

In [108…

```
# Get the last 6 months of data
last_6_months = filter_df_sales.tail(6)

# Split the data into train and test sets
X_train = filter_df_sales.drop('sales', axis=1)
y_train = filter_df_sales['sales']
X_test = last_6_months.drop('sales', axis=1)
y_test = last_6_months['sales']

# Create a linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the error
lr_mae = mean_absolute_error(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)

print("Mean Absolute Error: ", lr_mae)
print("Mean Squared Error: ", lr_mse)
print("Root Mean Squared Error:", lr_rmse)

all_sales = filter_df_sales['sales']

# Get the index of the last 6 months of data
last_6_months_index = last_6_months.index

# Plot the entire period of sales data
plt.figure(figsize=(20,10))
plt.plot(all_sales)

# Make predictions on the last 6 months of data
```

```
y_pred = model.predict(X_test)

# Plot the forecasted sales on the same chart
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Linear Regresion')
plt.legend(['Actual Sales', 'Forecasted Sales'])

plt.show()
```

```
Mean Absolute Error:  333.5431530297144
Mean Squared Error:  167517.48433976737
Root Mean Squared Error: 409.28899855697
```



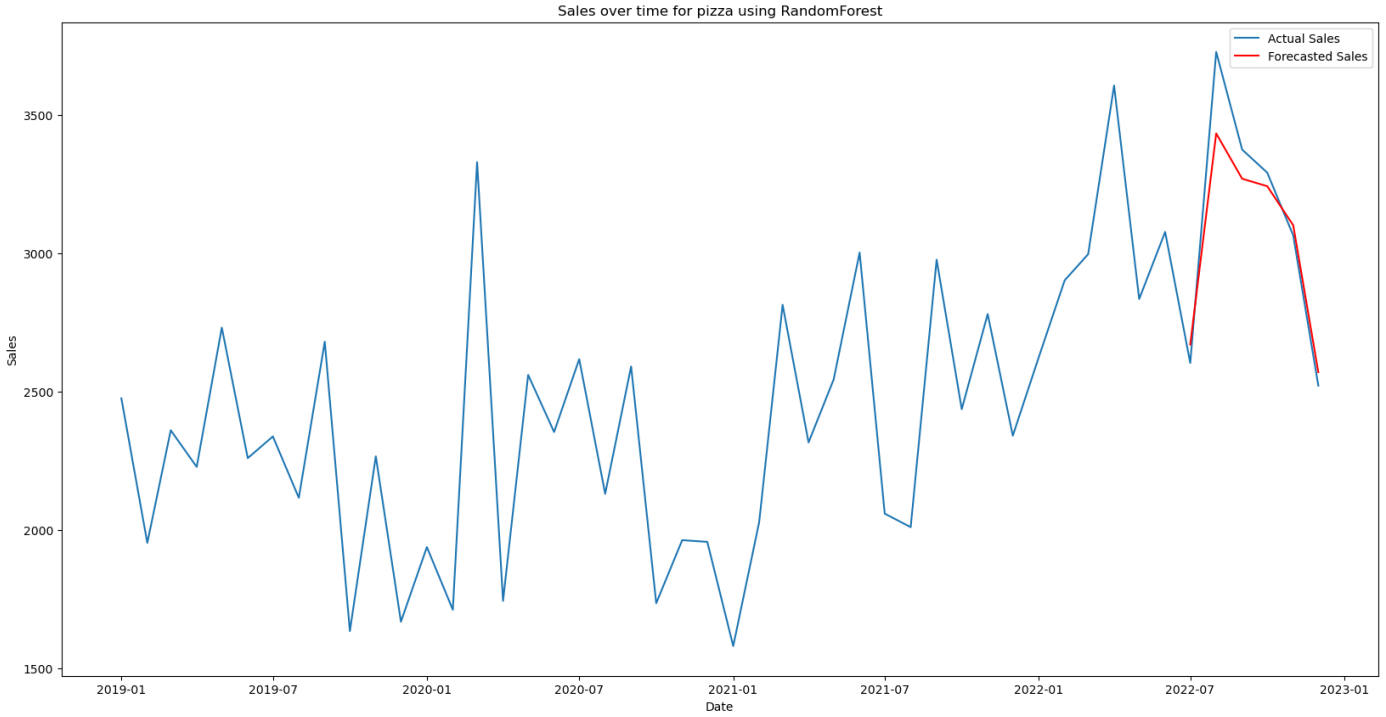
RF - pizza

In [109…

```
last_6_months = filter_df_sales.tail(6)

X_train = filter_df_sales.drop(columns=["sales"])
y_train = filter_df_sales["sales"]
X_test = last_6_months.drop(columns=["sales"])
y_test = last_6_months["sales"]

# Create a random forest model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Use the model to make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate the model's accuracy
accuracy = rf.score(X_test, y_test)
print("Accuracy: ", accuracy)

# Calculate error
rf_mae = mean_absolute_error(y_test, y_pred)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rf_mse)
```

```
print("Mean Absolute Error: ", rf_mae)
print("Mean Squared Error: ", rf_mse)
print("Root Mean Squared Error:", rf_rmse)

# Line graph for the forcast
plt.figure(figsize=(20,10))

plt.plot(all_sales)
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using RandomForest')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
Accuracy:  0.9003664497512486
Mean Absolute Error:  100.11986666666485
Mean Squared Error:  18062.917924752746
Root Mean Squared Error: 134.3983553647616
```



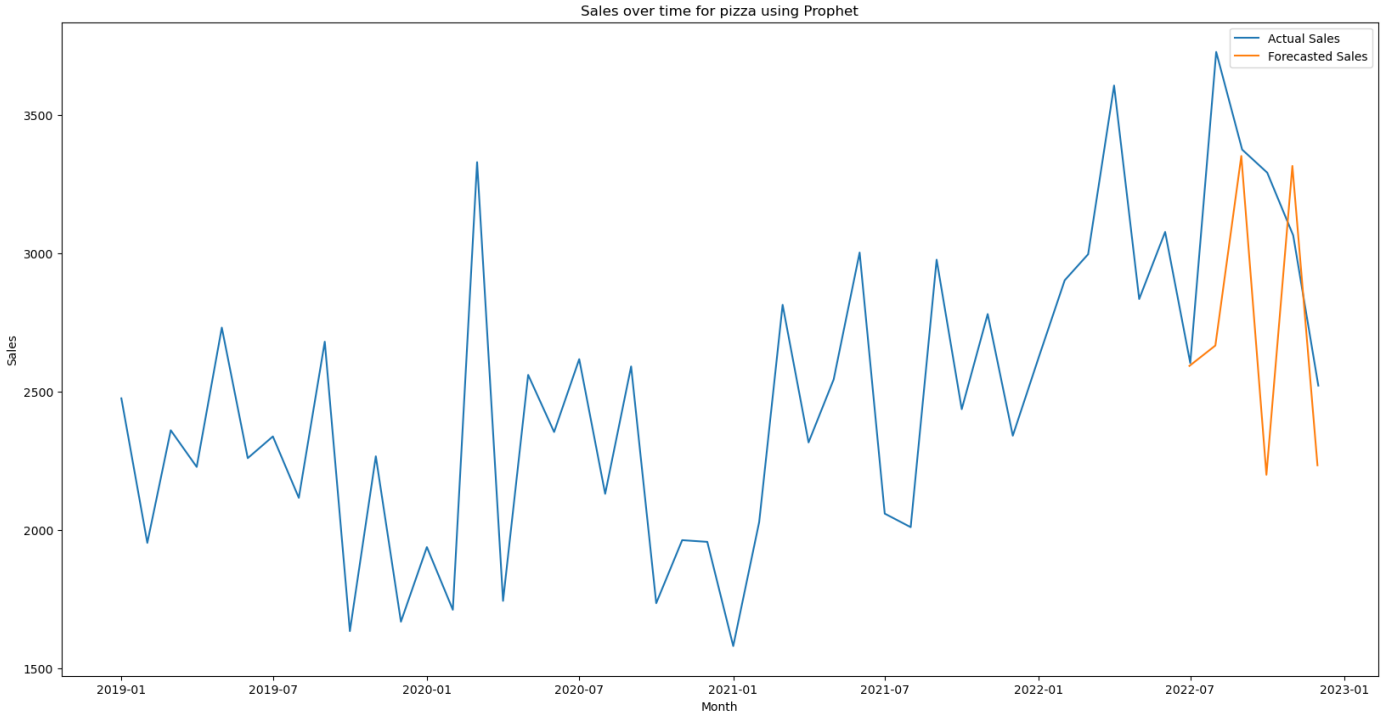
PR - pizza

In [110…

```
# create a copy of the dataframe with a column named 'ds' with the index values
filter_df_sales_prophet = filter_df_sales.reset_index().rename(columns={'month_id': 'ds'
filter_df_sales_prophet = filter_df_sales_prophet.rename(columns={'sales':'y'})

train_data = filter_df_sales_prophet[:-6]
test_data = filter_df_sales_prophet[-6:]

model = Prophet()

# Fit the model on the data
model.fit(train_data)

# Create a dataframe to hold the forecast
future_data = model.make_future_dataframe(periods=6, freq='m')

# Make predictions
forecast = model.predict(future_data)
forecast = forecast.tail(6)
```

```
# Calculate error
pr_mae = mean_absolute_error(test_data['y'], forecast[-6:]['yhat'])
pr_mse = mean_squared_error(test_data['y'], forecast[-6:]['yhat'])
pr_rmse = np.sqrt(pr_mse)
print("Mean Absolute Error:", pr_mae)
print("Mean Squared Error:", pr_mse)
print("Root Mean Squared Error:", pr_rmse)

# Plot the forecast
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index, filter_df_sales['sales'], label='Actual Sales')

# Plot the predicted values for the last six months
plt.plot(forecast.ds, forecast.yhat, label='Predicted Sales')

plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Prophet')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
21:35:26 - cmdstanpy - INFO - Chain [1] start processing
21:35:26 - cmdstanpy - INFO - Chain [1] done processing
Mean Absolute Error: 454.19137619152457
Mean Squared Error: 410590.9881513573
Root Mean Squared Error: 640.7737417773587
```



In [111…

```
# Add the errors from all methods to the DataFrame
new_row_df =  pd.DataFrame({'product': [product], 'LR-MAE': [lr_mae], 'LR-MSE': [lr_mse]
errors_df = pd.concat([errors_df, new_row_df], ignore_index=True)
print(errors_df)
```

	product	LR-MAE	LR-MSE	LR-RMSE	RF-MAE	\
0	apa_plata	887.015829	988186.848307	994.075877	377.426267	
1	apa_minerala	44.324535	2453.366899	49.531474	10.533100	
2	bere	283.109792	95844.002658	309.586826	65.444150	
3	cafea	287.818261	94385.213013	307.221765	134.035500	
4	carnati	264.936590	99377.947534	315.242680	155.934000	
5	frigider	13.054626	245.441803	15.666582	8.523900	
6	lapte	490.769164	269944.557868	519.561890	193.309633	
7	legume	317.334317	168366.068324	410.324345	124.748417	

8	masina_de_spalat	18.592357	606.867326	24.634677	8.862467
9	orez	376.601361	165639.594918	406.988446	98.474217
10	oua	725.835859	635033.703803	796.890020	307.595917
11	paste	181.288571	45954.816231	214.370745	79.459933
12	peste	175.548619	51772.454330	227.535611	98.965167
13	pizza	333.543153	167517.484340	409.288999	100.119867

	RF-MSE	RF-RMSE	PR-MAE	PR-MSE	PR-RMSE
0	200759.770117	448.062239	1107.910649	1.661801e+06	1289.108561
1	153.080170	12.372557	1703.258881	1.172504e+07	3424.185038
2	6221.823280	78.878535	437.297885	2.623325e+05	512.184061
3	22921.110989	151.397196	554.567175	3.976641e+05	630.606141
4	33523.190407	183.093393	370.439785	1.702760e+05	412.645094
5	105.584959	10.275454	34.075515	1.834104e+03	42.826436
6	48061.717358	219.229828	968.181840	1.101085e+06	1049.325744
7	23345.264739	152.791573	605.913460	4.861077e+05	697.214275
8	139.974418	11.831078	40.934713	2.225668e+03	47.176986
9	17005.647771	130.405705	562.437890	5.253923e+05	724.839518
10	114084.843273	337.764479	1055.734550	1.583208e+06	1258.255738
11	9798.345174	98.986591	159.511806	3.246713e+04	180.186386
12	13836.018312	117.626605	517.236332	3.505682e+05	592.087976
13	18062.917925	134.398355	454.191376	4.105910e+05	640.773742

Racoritoare DataFrame

In [112…

```
# Filter the initial DataFrame
product = 'racoritaore'
filter_df = df.loc[df['denumire_produs'] == product]
filter_df = filter_df.drop(columns=['colli', 'store_county'])
#print(apa_plata_df)

# Group by month_id and aggregate the other columns
filter_df_sales = filter_df.groupby("month_id").agg({'sales': 'sum','Inflatie anuala': '
filter_df_sales = filter_df_sales.sort_values(by='month_id', ascending=True)

# Print the apa_plata_sales dataframe
print(filter_df_sales)
```

month_id	sales	Inflatie anuala	unnemployment
2019-01-01	1108.46	3.8	1.42
2019-02-01	1000.24	3.8	3.32
2019-03-01	1155.07	3.8	3.31
2019-04-01	1219.38	3.8	3.19
2019-05-01	1558.59	3.8	3.00
2019-06-01	1357.45	3.8	2.95
2019-07-01	1413.91	3.8	2.95
2019-08-01	2157.45	3.8	3.01
2019-09-01	1456.56	3.8	3.03
2019-10-01	1665.64	3.8	3.00
2019-11-01	1464.34	3.8	2.98
2019-12-01	2081.36	3.8	2.98
2020-01-01	1467.42	2.6	2.97
2020-02-01	1668.44	2.6	2.98
2020-03-01	2501.68	2.6	2.95
2020-04-01	970.53	2.6	2.88
2020-05-01	1248.31	2.6	2.90
2020-06-01	1600.39	2.6	2.87
2020-07-01	1278.74	2.6	3.00
2020-08-01	2335.09	2.6	3.02
2020-09-01	2294.24	2.6	3.30
2020-10-01	2042.27	2.6	3.26
2020-11-01	1565.68	2.6	3.27
2020-12-01	1526.19	2.6	3.32

2021-01-01	1370.75	5.1	3.38
2021-02-01	1273.65	5.1	3.34
2021-03-01	1721.65	5.1	3.35
2021-04-01	1802.84	5.1	3.33
2021-05-01	2151.84	5.1	3.16
2021-06-01	1861.34	5.1	3.06
2021-07-01	1695.67	5.1	3.00
2021-08-01	2569.49	5.1	2.96
2021-09-01	2791.49	5.1	2.93
2021-10-01	2585.48	5.1	2.85
2021-11-01	2207.00	5.1	2.76
2021-12-01	2743.49	5.1	2.72
2022-01-01	1610.68	13.8	2.69
2022-02-01	2838.23	13.8	2.68
2022-03-01	2143.11	13.8	2.67
2022-04-01	2410.50	13.8	2.64
2022-05-01	2253.33	13.8	2.57
2022-06-01	2720.47	13.8	2.55
2022-07-01	2750.62	13.8	2.55
2022-08-01	2640.33	13.8	2.56
2022-09-01	3439.50	13.8	2.56
2022-10-01	2573.20	13.8	2.88
2022-11-01	2794.53	13.8	2.96
2022-12-01	2370.11	13.8	3.04

month_id	Inflatie(de la o luna la alta)	net_average_salary
2019-01-01	0.83	2936.0
2019-02-01	0.79	2933.0
2019-03-01	0.49	3075.0
2019-04-01	0.61	3115.0
2019-05-01	0.46	3101.0
2019-06-01	-0.23	3142.0
2019-07-01	-0.20	3119.0
2019-08-01	0.06	3044.0
2019-09-01	0.09	3082.0
2019-10-01	0.43	3116.0
2019-11-01	0.23	3179.0
2019-12-01	0.42	3340.0
2020-01-01	0.41	3189.0
2020-02-01	0.25	3202.0
2020-03-01	0.50	3294.0
2020-04-01	0.26	3182.0
2020-05-01	0.05	3179.0
2020-06-01	0.08	3298.0
2020-07-01	0.00	3372.0
2020-08-01	-0.05	3275.0
2020-09-01	-0.14	3321.0
2020-10-01	0.22	3343.0
2020-11-01	0.13	3411.0
2020-12-01	0.34	3620.0
2021-01-01	1.33	3395.0
2021-02-01	0.41	3365.0
2021-03-01	0.38	3547.0
2021-04-01	0.45	3561.0
2021-05-01	0.53	3492.0
2021-06-01	0.27	3541.0
2021-07-01	0.97	3545.0
2021-08-01	0.24	3487.0
2021-09-01	0.84	3517.0
2021-10-01	1.78	3544.0
2021-11-01	0.00	3645.0
2021-12-01	0.71	3879.0
2022-01-01	1.48	3698.0
2022-02-01	0.58	3721.0
2022-03-01	1.88	3937.0

month_id		
2022-04-01	3.74	3967.0
2022-05-01	1.18	3928.0
2022-06-01	0.76	3977.0
2022-07-01	0.89	3975.0
2022-08-01	0.56	3933.0
2022-09-01	1.33	4003.0
2022-10-01	1.28	4008.0
2022-11-01	1.25	4141.0
2022-12-01	0.37	4141.0

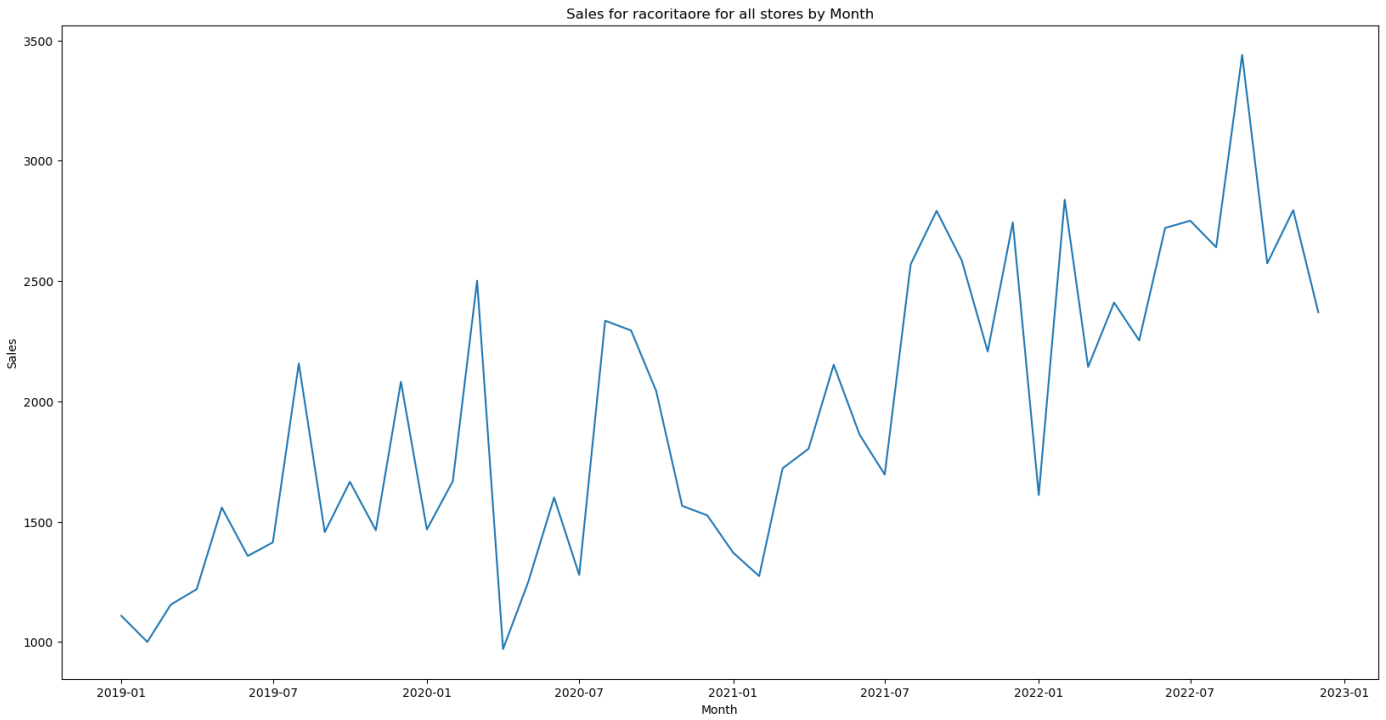
month_id	IPC Marfuri alimentare (%)	IPC Marfuri nealimentare (%)
2019-01-01	101.30	100.63
2019-02-01	101.27	100.57
2019-03-01	100.48	100.54
2019-04-01	100.68	100.53
2019-05-01	100.79	100.20
2019-06-01	99.53	99.76
2019-07-01	99.34	99.98
2019-08-01	99.71	100.22
2019-09-01	99.91	100.13
2019-10-01	100.70	100.32
2019-11-01	100.43	100.12
2019-12-01	100.84	100.28
2020-01-01	100.99	100.02
2020-02-01	100.63	99.94
2020-03-01	101.46	99.91
2020-04-01	101.27	99.67
2020-05-01	100.34	99.82
2020-06-01	99.62	100.28
2020-07-01	99.55	100.19
2020-08-01	99.59	100.08
2020-09-01	99.45	99.99
2020-10-01	100.11	100.31
2020-11-01	99.92	100.29
2020-12-01	100.29	100.51
2021-01-01	100.63	102.24
2021-02-01	100.46	100.47
2021-03-01	100.37	100.46
2021-04-01	100.45	100.47
2021-05-01	101.10	100.28
2021-06-01	100.25	100.29
2021-07-01	99.71	102.02
2021-08-01	99.95	100.34
2021-09-01	100.95	100.73
2021-10-01	101.06	102.78
2021-11-01	100.73	99.47
2021-12-01	100.84	100.74
2022-01-01	101.15	101.73
2022-02-01	101.96	99.70
2022-03-01	102.54	101.86
2022-04-01	102.56	105.45
2022-05-01	101.73	101.00
2022-06-01	100.62	100.92
2022-07-01	100.92	100.87
2022-08-01	101.82	99.81
2022-09-01	101.72	101.28
2022-10-01	102.29	100.80
2022-11-01	101.54	101.03
2022-12-01	101.26	99.68

month_id	IPC Servicii (%)
2019-01-01	100.57
2019-02-01	100.55
2019-03-01	100.40

```
2019-04-01            100.72
2019-05-01            100.55
2019-06-01            100.17
2019-07-01            100.10
2019-08-01            100.25
2019-09-01            100.27
2019-10-01            100.25
2019-11-01            100.15
2019-12-01            100.12
2020-01-01            100.43
2020-02-01            100.39
2020-03-01            100.35
2020-04-01            100.00
2020-05-01            100.11
2020-06-01            100.34
2020-07-01            100.31
2020-08-01            100.21
2020-09-01            100.20
2020-10-01            100.22
2020-11-01            100.07
2020-12-01            100.04
2021-01-01            100.25
2021-02-01            100.20
2021-03-01            100.24
2021-04-01            100.40
2021-05-01            100.27
2021-06-01            100.23
2021-07-01            100.39
2021-08-01            100.43
2021-09-01            100.96
2021-10-01            100.42
2021-11-01            100.20
2021-12-01            100.42
2022-01-01            101.37
2022-02-01            100.60
2022-03-01            100.67
2022-04-01            100.94
2022-05-01            100.61
2022-06-01            100.54
2022-07-01            100.88
2022-08-01            100.37
2022-09-01            100.72
2022-10-01            100.70
2022-11-01            101.31
2022-12-01            100.67
```

In [113…

```
# Line graph for apa_plata for all stores
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index.get_level_values(0), filter_df_sales['sales'])
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales for ' + product + ' for all stores by Month')
plt.show()
```



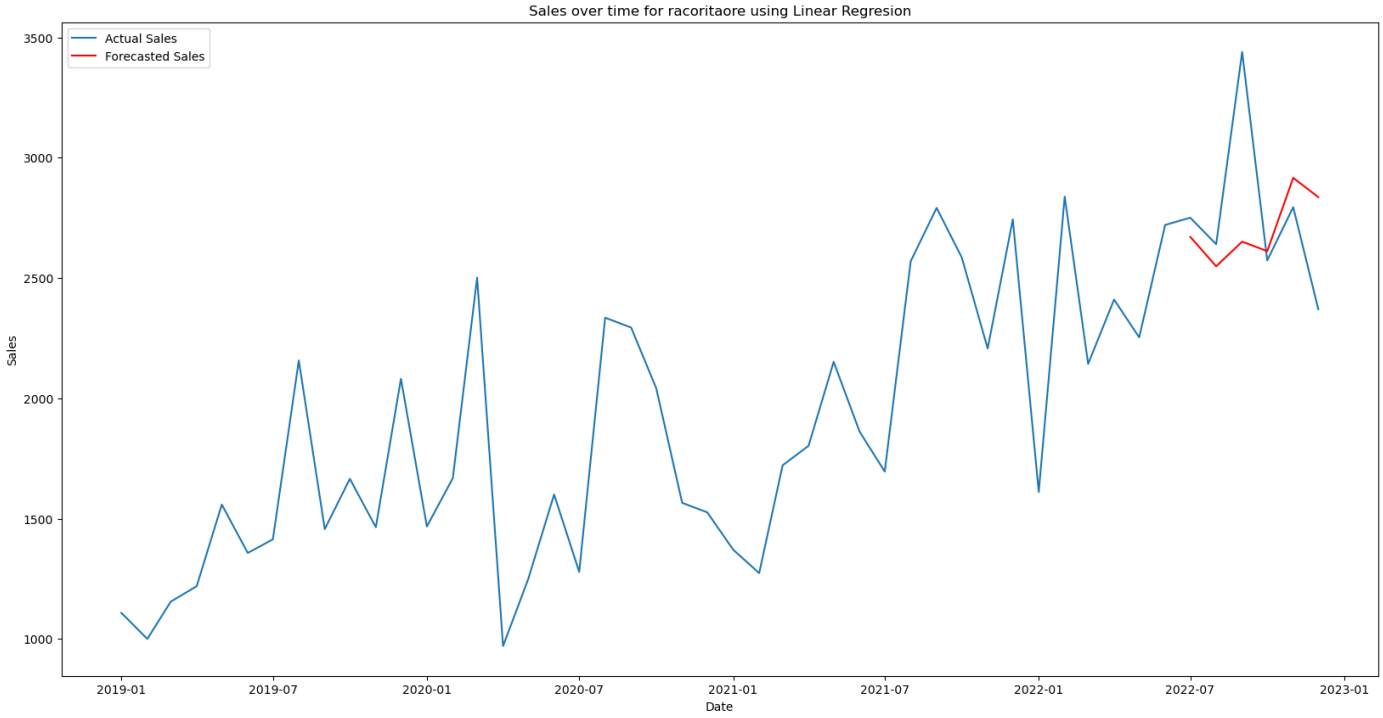
LR - racoritaore

In [114…

```
# Get the last 6 months of data
last_6_months = filter_df_sales.tail(6)

# Split the data into train and test sets
X_train = filter_df_sales.drop('sales', axis=1)
y_train = filter_df_sales['sales']
X_test = last_6_months.drop('sales', axis=1)
y_test = last_6_months['sales']

# Create a linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the error
lr_mae = mean_absolute_error(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)

print("Mean Absolute Error: ", lr_mae)
print("Mean Squared Error: ", lr_mse)
print("Root Mean Squared Error:", lr_rmse)

all_sales = filter_df_sales['sales']

# Get the index of the last 6 months of data
last_6_months_index = last_6_months.index

# Plot the entire period of sales data
plt.figure(figsize=(20,10))
plt.plot(all_sales)

# Make predictions on the last 6 months of data
```

```
y_pred = model.predict(X_test)

# Plot the forecasted sales on the same chart
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Linear Regresion')
plt.legend(['Actual Sales', 'Forecasted Sales'])

plt.show()
```

```
Mean Absolute Error:  264.2732741599513
Mean Squared Error:  144927.85931854288
Root Mean Squared Error: 380.69391815281585
```



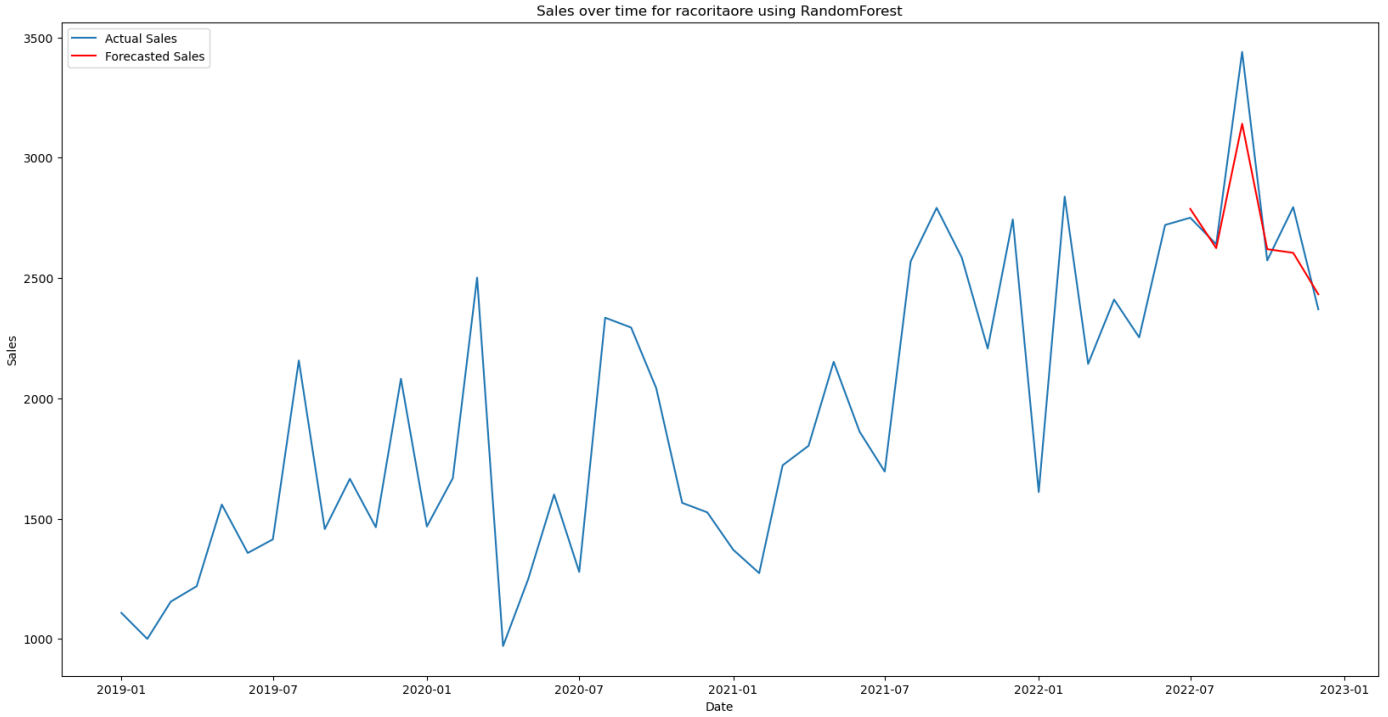
RF - racoritaore

In [115…

```
last_6_months = filter_df_sales.tail(6)

X_train = filter_df_sales.drop(columns=["sales"])
y_train = filter_df_sales["sales"]
X_test = last_6_months.drop(columns=["sales"])
y_test = last_6_months["sales"]

# Create a random forest model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Use the model to make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate the model's accuracy
accuracy = rf.score(X_test, y_test)
print("Accuracy: ", accuracy)

# Calculate error
rf_mae = mean_absolute_error(y_test, y_pred)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rf_mse)
```

```
print("Mean Absolute Error: ", rf_mae)
print("Mean Squared Error: ", rf_mse)
print("Root Mean Squared Error:", rf_rmse)

# Line graph for the forcast
plt.figure(figsize=(20,10))

plt.plot(all_sales)
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using RandomForest')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
Accuracy:  0.8001122362360873
Mean Absolute Error:  108.3312666666668
Mean Squared Error:  22128.19051275338
Root Mean Squared Error: 148.7554722111203
```



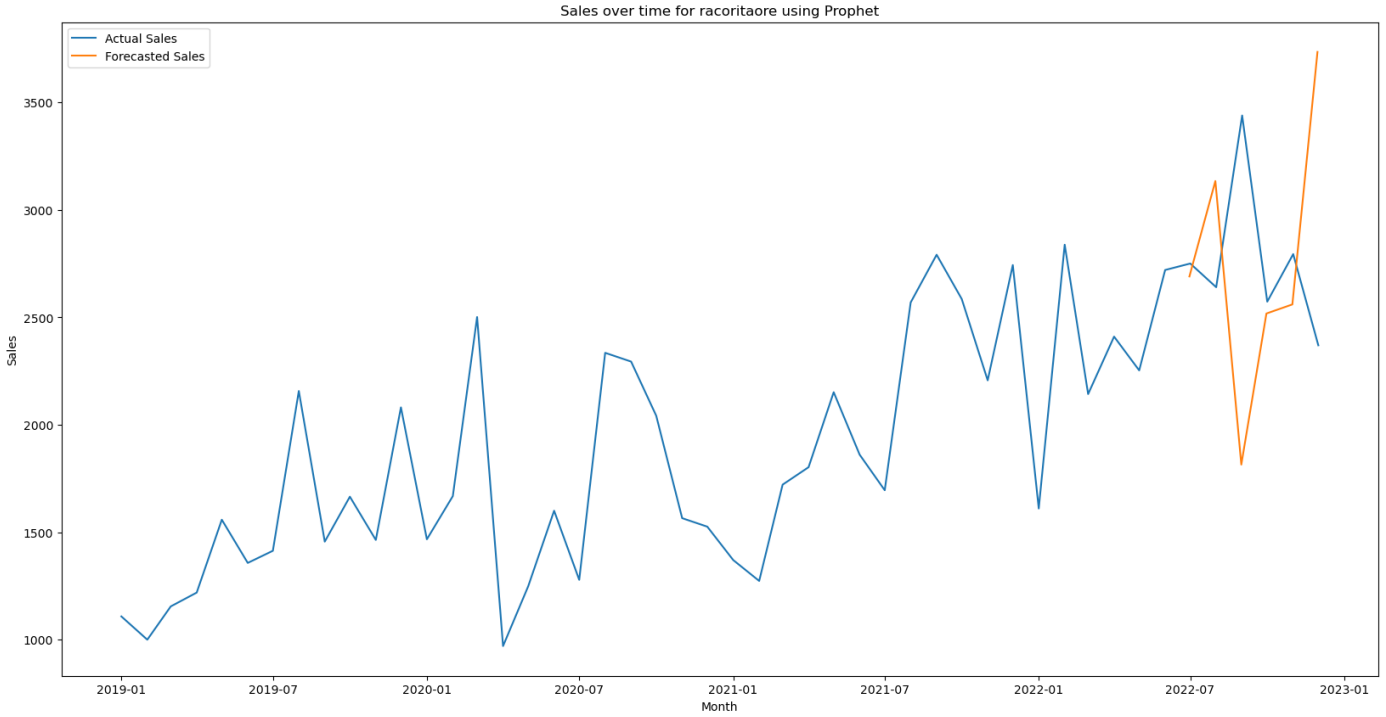
PR - racoritoare

In [116…

```
# create a copy of the dataframe with a column named 'ds' with the index values
filter_df_sales_prophet = filter_df_sales.reset_index().rename(columns={'month_id': 'ds'
filter_df_sales_prophet = filter_df_sales_prophet.rename(columns={'sales':'y'})

train_data = filter_df_sales_prophet[:-6]
test_data = filter_df_sales_prophet[-6:]

model = Prophet()

# Fit the model on the data
model.fit(train_data)

# Create a dataframe to hold the forecast
future_data = model.make_future_dataframe(periods=6, freq='m')

# Make predictions
forecast = model.predict(future_data)
forecast = forecast.tail(6)
```

```
# Calculate error
pr_mae = mean_absolute_error(test_data['y'], forecast[-6:]['yhat'])
pr_mse = mean_squared_error(test_data['y'], forecast[-6:]['yhat'])
pr_rmse = np.sqrt(pr_mse)
print("Mean Absolute Error:", pr_mae)
print("Mean Squared Error:", pr_mse)
print("Root Mean Squared Error:", pr_rmse)

# Plot the forecast
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index, filter_df_sales['sales'], label='Actual Sales')

# Plot the predicted values for the last six months
plt.plot(forecast.ds, forecast.yhat, label='Predicted Sales')

plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Prophet')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
21:35:27 - cmdstanpy - INFO - Chain [1] start processing
21:35:28 - cmdstanpy - INFO - Chain [1] done processing
Mean Absolute Error: 638.8678453031873
Mean Squared Error: 801310.8002244445
Root Mean Squared Error: 895.1596506905595
```



In [117…

```
# Add the errors from all methods to the DataFrame
new_row_df =  pd.DataFrame({'product': [product], 'LR-MAE': [lr_mae], 'LR-MSE': [lr_mse]
errors_df = pd.concat([errors_df, new_row_df], ignore_index=True)
print(errors_df)
```

	product	LR-MAE	LR-MSE	LR-RMSE	RF-MAE	\
0	apa_plata	887.015829	988186.848307	994.075877	377.426267	
1	apa_minerala	44.324535	2453.366899	49.531474	10.533100	
2	bere	283.109792	95844.002658	309.586826	65.444150	
3	cafea	287.818261	94385.213013	307.221765	134.035500	
4	carnati	264.936590	99377.947534	315.242680	155.934000	
5	frigider	13.054626	245.441803	15.666582	8.523900	
6	lapte	490.769164	269944.557868	519.561890	193.309633	
7	legume	317.334317	168366.068324	410.324345	124.748417	

8	masina_de_spalat	18.592357	606.867326	24.634677	8.862467
9	orez	376.601361	165639.594918	406.988446	98.474217
10	oua	725.835859	635033.703803	796.890020	307.595917
11	paste	181.288571	45954.816231	214.370745	79.459933
12	peste	175.548619	51772.454330	227.535611	98.965167
13	pizza	333.543153	167517.484340	409.288999	100.119867
14	racoritaore	264.273274	144927.859319	380.693918	108.331267

	RF-MSE	RF-RMSE	PR-MAE	PR-MSE	PR-RMSE
0	200759.770117	448.062239	1107.910649	1.661801e+06	1289.108561
1	153.080170	12.372557	1703.258881	1.172504e+07	3424.185038
2	6221.823280	78.878535	437.297885	2.623325e+05	512.184061
3	22921.110989	151.397196	554.567175	3.976641e+05	630.606141
4	33523.190407	183.093393	370.439785	1.702760e+05	412.645094
5	105.584959	10.275454	34.075515	1.834104e+03	42.826436
6	48061.717358	219.229828	968.181840	1.101085e+06	1049.325744
7	23345.264739	152.791573	605.913460	4.861077e+05	697.214275
8	139.974418	11.831078	40.934713	2.225668e+03	47.176986
9	17005.647771	130.405705	562.437890	5.253923e+05	724.839518
10	114084.843273	337.764479	1055.734550	1.583208e+06	1258.255738
11	9798.345174	98.986591	159.511806	3.246713e+04	180.186386
12	13836.018312	117.626605	517.236332	3.505682e+05	592.087976
13	18062.917925	134.398355	454.191376	4.105910e+05	640.773742
14	22128.190513	148.755472	638.867845	8.013108e+05	895.159651

Servetele DataFrame

In [118…

```
# Filter the initial DataFrame
product = 'servetele'
filter_df = df.loc[df['denumire_produs'] == product]
filter_df = filter_df.drop(columns=['colli', 'store_county'])
#print(apa_plata_df)

# Group by month_id and aggregate the other columns
filter_df_sales = filter_df.groupby("month_id").agg({'sales': 'sum','Inflatie anuala': '
filter_df_sales = filter_df_sales.sort_values(by='month_id', ascending=True)

# Print the apa_plata_sales dataframe
print(filter_df_sales)
```

month_id	sales	Inflatie anuala	unnemployment	\
2019-01-01	1183.13	3.8	1.42	
2019-02-01	1134.30	3.8	3.32	
2019-03-01	896.43	3.8	3.31	
2019-04-01	673.79	3.8	3.19	
2019-05-01	875.55	3.8	3.00	
2019-06-01	882.71	3.8	2.95	
2019-07-01	862.73	3.8	2.95	
2019-08-01	1111.41	3.8	3.01	
2019-09-01	1318.50	3.8	3.03	
2019-10-01	917.12	3.8	3.00	
2019-11-01	1074.64	3.8	2.98	
2019-12-01	996.02	3.8	2.98	
2020-01-01	1090.51	2.6	2.97	
2020-02-01	834.06	2.6	2.98	
2020-03-01	1856.48	2.6	2.95	
2020-04-01	1013.73	2.6	2.88	
2020-05-01	1119.27	2.6	2.90	
2020-06-01	1136.44	2.6	2.87	
2020-07-01	1543.55	2.6	3.00	
2020-08-01	1336.74	2.6	3.02	
2020-09-01	1236.73	2.6	3.30	
2020-10-01	1139.18	2.6	3.26	

2020-11-01	1779.96	2.6	3.27
2020-12-01	1375.81	2.6	3.32
2021-01-01	1483.58	5.1	3.38
2021-02-01	1321.80	5.1	3.34
2021-03-01	1626.54	5.1	3.35
2021-04-01	1408.73	5.1	3.33
2021-05-01	1657.32	5.1	3.16
2021-06-01	1152.05	5.1	3.06
2021-07-01	2110.81	5.1	3.00
2021-08-01	1695.21	5.1	2.96
2021-09-01	1830.61	5.1	2.93
2021-10-01	1793.77	5.1	2.85
2021-11-01	1721.16	5.1	2.76
2021-12-01	1982.02	5.1	2.72
2022-01-01	1980.04	13.8	2.69
2022-02-01	2602.12	13.8	2.68
2022-03-01	2298.99	13.8	2.67
2022-04-01	1783.00	13.8	2.64
2022-05-01	2807.14	13.8	2.57
2022-06-01	1686.94	13.8	2.55
2022-07-01	2417.86	13.8	2.55
2022-08-01	1769.41	13.8	2.56
2022-09-01	2552.77	13.8	2.56
2022-10-01	1802.01	13.8	2.88
2022-11-01	1949.98	13.8	2.96
2022-12-01	2047.38	13.8	3.04

month_id	Inflatie(de la o luna la alta)	net_average_salary
2019-01-01	0.83	2936.0
2019-02-01	0.79	2933.0
2019-03-01	0.49	3075.0
2019-04-01	0.61	3115.0
2019-05-01	0.46	3101.0
2019-06-01	-0.23	3142.0
2019-07-01	-0.20	3119.0
2019-08-01	0.06	3044.0
2019-09-01	0.09	3082.0
2019-10-01	0.43	3116.0
2019-11-01	0.23	3179.0
2019-12-01	0.42	3340.0
2020-01-01	0.41	3189.0
2020-02-01	0.25	3202.0
2020-03-01	0.50	3294.0
2020-04-01	0.26	3182.0
2020-05-01	0.05	3179.0
2020-06-01	0.08	3298.0
2020-07-01	0.00	3372.0
2020-08-01	-0.05	3275.0
2020-09-01	-0.14	3321.0
2020-10-01	0.22	3343.0
2020-11-01	0.13	3411.0
2020-12-01	0.34	3620.0
2021-01-01	1.33	3395.0
2021-02-01	0.41	3365.0
2021-03-01	0.38	3547.0
2021-04-01	0.45	3561.0
2021-05-01	0.53	3492.0
2021-06-01	0.27	3541.0
2021-07-01	0.97	3545.0
2021-08-01	0.24	3487.0
2021-09-01	0.84	3517.0
2021-10-01	1.78	3544.0
2021-11-01	0.00	3645.0
2021-12-01	0.71	3879.0
2022-01-01	1.48	3698.0

2022-02-01	0.58	3721.0
2022-03-01	1.88	3937.0
2022-04-01	3.74	3967.0
2022-05-01	1.18	3928.0
2022-06-01	0.76	3977.0
2022-07-01	0.89	3975.0
2022-08-01	0.56	3933.0
2022-09-01	1.33	4003.0
2022-10-01	1.28	4008.0
2022-11-01	1.25	4141.0
2022-12-01	0.37	4141.0

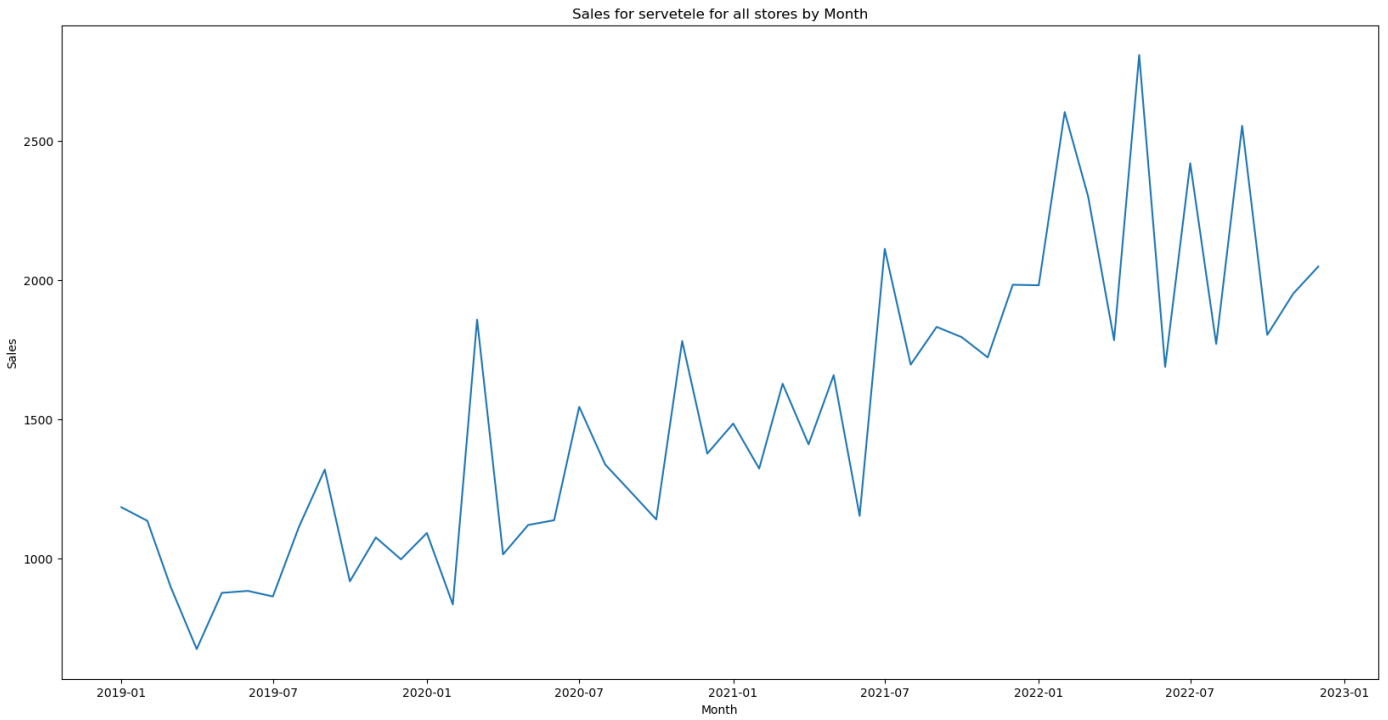
month_id	IPC Marfuri alimentare (%)	IPC Marfuri nealimentare (%)	\
2019-01-01	101.30	100.63	
2019-02-01	101.27	100.57	
2019-03-01	100.48	100.54	
2019-04-01	100.68	100.53	
2019-05-01	100.79	100.20	
2019-06-01	99.53	99.76	
2019-07-01	99.34	99.98	
2019-08-01	99.71	100.22	
2019-09-01	99.91	100.13	
2019-10-01	100.70	100.32	
2019-11-01	100.43	100.12	
2019-12-01	100.84	100.28	
2020-01-01	100.99	100.02	
2020-02-01	100.63	99.94	
2020-03-01	101.46	99.91	
2020-04-01	101.27	99.67	
2020-05-01	100.34	99.82	
2020-06-01	99.62	100.28	
2020-07-01	99.55	100.19	
2020-08-01	99.59	100.08	
2020-09-01	99.45	99.99	
2020-10-01	100.11	100.31	
2020-11-01	99.92	100.29	
2020-12-01	100.29	100.51	
2021-01-01	100.63	102.24	
2021-02-01	100.46	100.47	
2021-03-01	100.37	100.46	
2021-04-01	100.45	100.47	
2021-05-01	101.10	100.28	
2021-06-01	100.25	100.29	
2021-07-01	99.71	102.02	
2021-08-01	99.95	100.34	
2021-09-01	100.95	100.73	
2021-10-01	101.06	102.78	
2021-11-01	100.73	99.47	
2021-12-01	100.84	100.74	
2022-01-01	101.15	101.73	
2022-02-01	101.96	99.70	
2022-03-01	102.54	101.86	
2022-04-01	102.56	105.45	
2022-05-01	101.73	101.00	
2022-06-01	100.62	100.92	
2022-07-01	100.92	100.87	
2022-08-01	101.82	99.81	
2022-09-01	101.72	101.28	
2022-10-01	102.29	100.80	
2022-11-01	101.54	101.03	
2022-12-01	101.26	99.68	

month_id	IPC Servicii (%)
2019-01-01	100.57

```
2019-02-01            100.55
2019-03-01            100.40
2019-04-01            100.72
2019-05-01            100.55
2019-06-01            100.17
2019-07-01            100.10
2019-08-01            100.25
2019-09-01            100.27
2019-10-01            100.25
2019-11-01            100.15
2019-12-01            100.12
2020-01-01            100.43
2020-02-01            100.39
2020-03-01            100.35
2020-04-01            100.00
2020-05-01            100.11
2020-06-01            100.34
2020-07-01            100.31
2020-08-01            100.21
2020-09-01            100.20
2020-10-01            100.22
2020-11-01            100.07
2020-12-01            100.04
2021-01-01            100.25
2021-02-01            100.20
2021-03-01            100.24
2021-04-01            100.40
2021-05-01            100.27
2021-06-01            100.23
2021-07-01            100.39
2021-08-01            100.43
2021-09-01            100.96
2021-10-01            100.42
2021-11-01            100.20
2021-12-01            100.42
2022-01-01            101.37
2022-02-01            100.60
2022-03-01            100.67
2022-04-01            100.94
2022-05-01            100.61
2022-06-01            100.54
2022-07-01            100.88
2022-08-01            100.37
2022-09-01            100.72
2022-10-01            100.70
2022-11-01            101.31
2022-12-01            100.67
```

In [119…

```python
# Line graph for apa_plata for all stores
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index.get_level_values(0), filter_df_sales['sales'])
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales for ' + product + ' for all stores by Month')
plt.show()
```



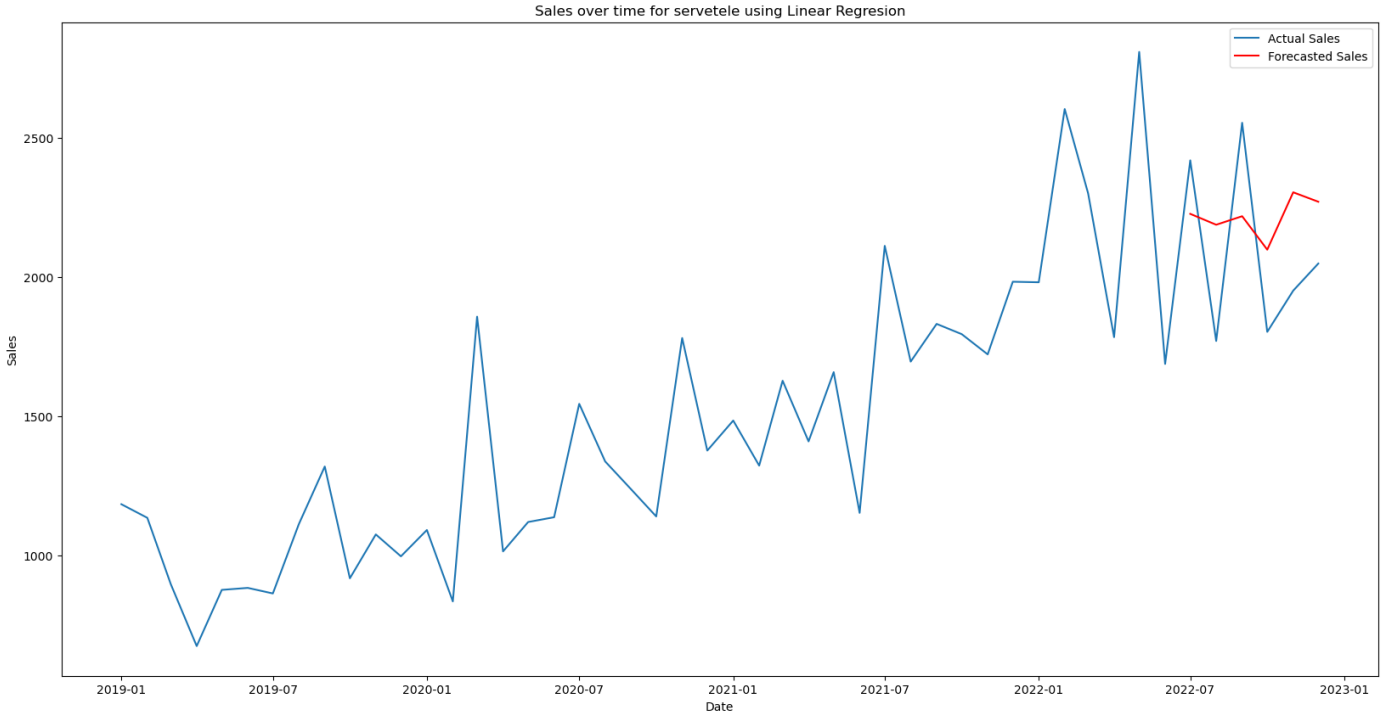
LR - servetele

In [120…

```
# Get the last 6 months of data
last_6_months = filter_df_sales.tail(6)

# Split the data into train and test sets
X_train = filter_df_sales.drop('sales', axis=1)
y_train = filter_df_sales['sales']
X_test = last_6_months.drop('sales', axis=1)
y_test = last_6_months['sales']

# Create a linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the error
lr_mae = mean_absolute_error(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)

print("Mean Absolute Error: ", lr_mae)
print("Mean Squared Error: ", lr_mse)
print("Root Mean Squared Error:", lr_rmse)

all_sales = filter_df_sales['sales']

# Get the index of the last 6 months of data
last_6_months_index = last_6_months.index

# Plot the entire period of sales data
plt.figure(figsize=(20,10))
plt.plot(all_sales)

# Make predictions on the last 6 months of data
```

```
y_pred = model.predict(X_test)

# Plot the forecasted sales on the same chart
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Linear Regresion')
plt.legend(['Actual Sales', 'Forecasted Sales'])

plt.show()
```

```
Mean Absolute Error:  302.49348089076034
Mean Squared Error:  97426.86032678693
Root Mean Squared Error: 312.1327607393798
```



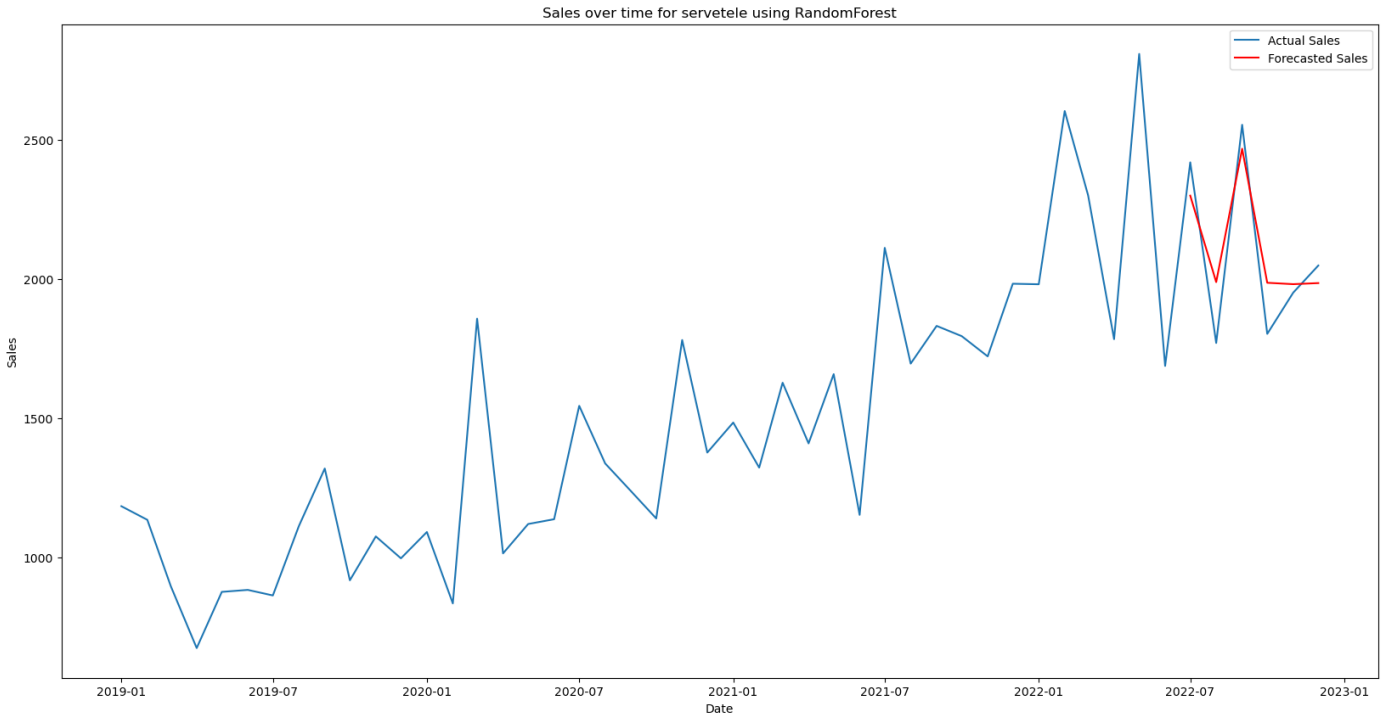
RF - servetele

In [121…

```
last_6_months = filter_df_sales.tail(6)

X_train = filter_df_sales.drop(columns=["sales"])
y_train = filter_df_sales["sales"]
X_test = last_6_months.drop(columns=["sales"])
y_test = last_6_months["sales"]

# Create a random forest model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Use the model to make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate the model's accuracy
accuracy = rf.score(X_test, y_test)
print("Accuracy: ", accuracy)

# Calculate error
rf_mae = mean_absolute_error(y_test, y_pred)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rf_mse)
```

```
print("Mean Absolute Error: ", rf_mae)
print("Mean Squared Error: ", rf_mse)
print("Root Mean Squared Error:", rf_rmse)

# Line graph for the forcast
plt.figure(figsize=(20,10))

plt.plot(all_sales)
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using RandomForest')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
Accuracy:  0.7957653072159412
Mean Absolute Error:  116.84568333333493
Mean Squared Error:  17999.422941688743
Root Mean Squared Error: 134.16192806339936
```



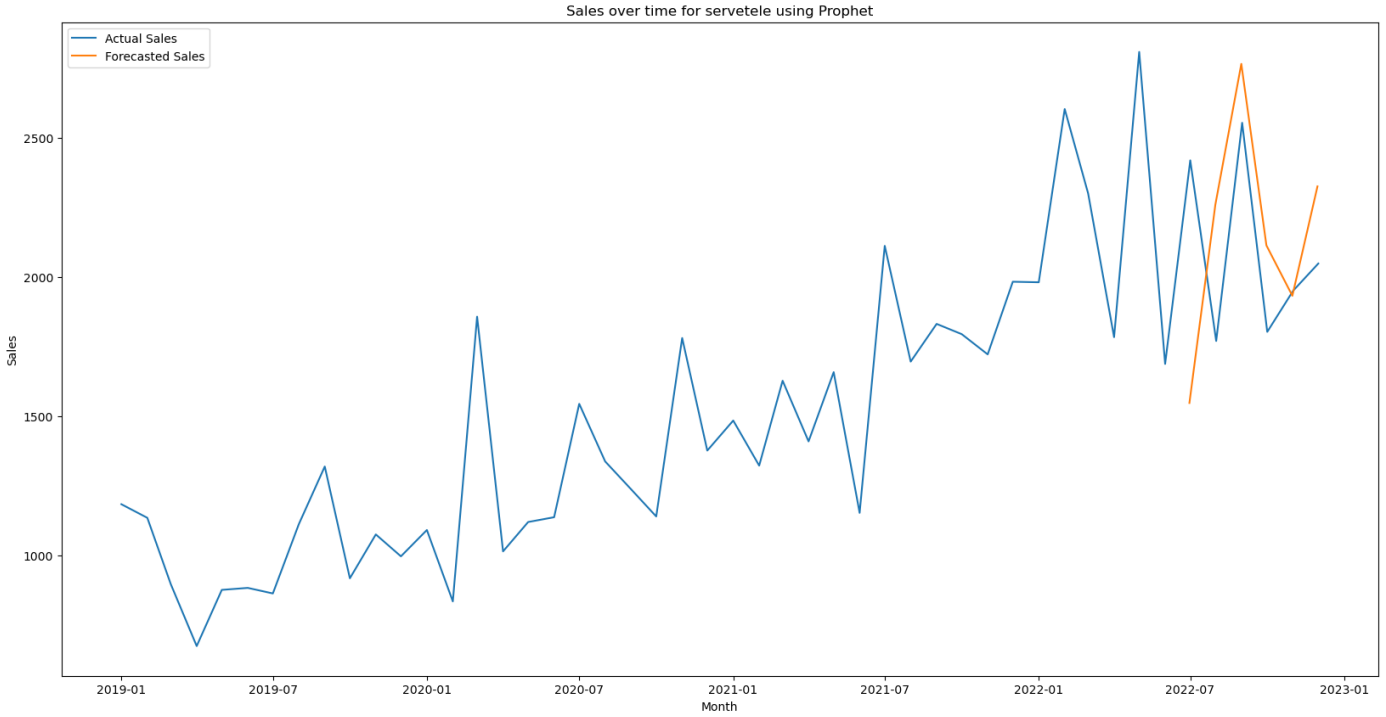
PR - servetele

In [122…

```
# create a copy of the dataframe with a column named 'ds' with the index values
filter_df_sales_prophet = filter_df_sales.reset_index().rename(columns={'month_id': 'ds'
filter_df_sales_prophet = filter_df_sales_prophet.rename(columns={'sales':'y'})

train_data = filter_df_sales_prophet[:-6]
test_data = filter_df_sales_prophet[-6:]

model = Prophet()

# Fit the model on the data
model.fit(train_data)

# Create a dataframe to hold the forecast
future_data = model.make_future_dataframe(periods=6, freq='m')

# Make predictions
forecast = model.predict(future_data)
forecast = forecast.tail(6)
```

```
# Calculate error
pr_mae = mean_absolute_error(test_data['y'], forecast[-6:]['yhat'])
pr_mse = mean_squared_error(test_data['y'], forecast[-6:]['yhat'])
pr_rmse = np.sqrt(pr_mse)
print("Mean Absolute Error:", pr_mae)
print("Mean Squared Error:", pr_mse)
print("Root Mean Squared Error:", pr_rmse)

# Plot the forecast
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index, filter_df_sales['sales'], label='Actual Sales')

# Plot the predicted values for the last six months
plt.plot(forecast.ds, forecast.yhat, label='Predicted Sales')

plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Prophet')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
21:35:29 - cmdstanpy - INFO - Chain [1] start processing
21:35:29 - cmdstanpy - INFO - Chain [1] done processing
Mean Absolute Error: 363.00607744708856
Mean Squared Error: 202785.05043420495
Root Mean Squared Error: 450.3166113238606
```



In [123…

```
# Add the errors from all methods to the DataFrame
new_row_df =  pd.DataFrame({'product': [product], 'LR-MAE': [lr_mae], 'LR-MSE': [lr_mse]
errors_df = pd.concat([errors_df, new_row_df], ignore_index=True)
print(errors_df)
```

	product	LR-MAE	LR-MSE	LR-RMSE	RF-MAE	\
0	apa_plata	887.015829	988186.848307	994.075877	377.426267	
1	apa_minerala	44.324535	2453.366899	49.531474	10.533100	
2	bere	283.109792	95844.002658	309.586826	65.444150	
3	cafea	287.818261	94385.213013	307.221765	134.035500	
4	carnati	264.936590	99377.947534	315.242680	155.934000	
5	frigider	13.054626	245.441803	15.666582	8.523900	
6	lapte	490.769164	269944.557868	519.561890	193.309633	
7	legume	317.334317	168366.068324	410.324345	124.748417	

8	masina_de_spalat	18.592357	606.867326	24.634677	8.862467
9	orez	376.601361	165639.594918	406.988446	98.474217
10	oua	725.835859	635033.703803	796.890020	307.595917
11	paste	181.288571	45954.816231	214.370745	79.459933
12	peste	175.548619	51772.454330	227.535611	98.965167
13	pizza	333.543153	167517.484340	409.288999	100.119867
14	racoritaore	264.273274	144927.859319	380.693918	108.331267
15	servetele	302.493481	97426.860327	312.132761	116.845683

	RF-MSE	RF-RMSE	PR-MAE	PR-MSE	PR-RMSE
0	200759.770117	448.062239	1107.910649	1.661801e+06	1289.108561
1	153.080170	12.372557	1703.258881	1.172504e+07	3424.185038
2	6221.823280	78.878535	437.297885	2.623325e+05	512.184061
3	22921.110989	151.397196	554.567175	3.976641e+05	630.606141
4	33523.190407	183.093393	370.439785	1.702760e+05	412.645094
5	105.584959	10.275454	34.075515	1.834104e+03	42.826436
6	48061.717358	219.229828	968.181840	1.101085e+06	1049.325744
7	23345.264739	152.791573	605.913460	4.861077e+05	697.214275
8	139.974418	11.831078	40.934713	2.225668e+03	47.176986
9	17005.647771	130.405705	562.437890	5.253923e+05	724.839518
10	114084.843273	337.764479	1055.734550	1.583208e+06	1258.255738
11	9798.345174	98.986591	159.511806	3.246713e+04	180.186386
12	13836.018312	117.626605	517.236332	3.505682e+05	592.087976
13	18062.917925	134.398355	454.191376	4.105910e+05	640.773742
14	22128.190513	148.755472	638.867845	8.013108e+05	895.159651
15	17999.422942	134.161928	363.006077	2.027851e+05	450.316611

Ulei de floarea soarelui DataFrame

In [124…

```
# Filter the initial DataFrame
product = 'ulei_soare'
filter_df = df.loc[df['denumire_produs'] == product]
filter_df = filter_df.drop(columns=['colli', 'store_county'])
#print(apa_plata_df)

# Group by month_id and aggregate the other columns
filter_df_sales = filter_df.groupby("month_id").agg({'sales': 'sum','Inflatie anuala': '
filter_df_sales = filter_df_sales.sort_values(by='month_id', ascending=True)

# Print the apa_plata_sales dataframe
print(filter_df_sales)
```

month_id	sales	Inflatie anuala	unnemployment	\
2019-01-01	3519.02	3.8	1.42	
2019-02-01	3876.90	3.8	3.32	
2019-03-01	3003.25	3.8	3.31	
2019-04-01	3653.85	3.8	3.19	
2019-05-01	3186.63	3.8	3.00	
2019-06-01	4584.11	3.8	2.95	
2019-07-01	3828.17	3.8	2.95	
2019-08-01	4272.62	3.8	3.01	
2019-09-01	4023.97	3.8	3.03	
2019-10-01	4117.05	3.8	3.00	
2019-11-01	3904.27	3.8	2.98	
2019-12-01	5165.67	3.8	2.98	
2020-01-01	4337.38	2.6	2.97	
2020-02-01	5202.26	2.6	2.98	
2020-03-01	6122.84	2.6	2.95	
2020-04-01	4140.08	2.6	2.88	
2020-05-01	4191.49	2.6	2.90	
2020-06-01	5449.78	2.6	2.87	
2020-07-01	5823.69	2.6	3.00	
2020-08-01	5872.72	2.6	3.02	

2020-09-01	4810.22	2.6	3.30
2020-10-01	5974.79	2.6	3.26
2020-11-01	5740.38	2.6	3.27
2020-12-01	6460.23	2.6	3.32
2021-01-01	5489.90	5.1	3.38
2021-02-01	5424.48	5.1	3.34
2021-03-01	5171.26	5.1	3.35
2021-04-01	5273.13	5.1	3.33
2021-05-01	5292.65	5.1	3.16
2021-06-01	3698.81	5.1	3.06
2021-07-01	3584.84	5.1	3.00
2021-08-01	4855.80	5.1	2.96
2021-09-01	5859.92	5.1	2.93
2021-10-01	5905.68	5.1	2.85
2021-11-01	6187.96	5.1	2.76
2021-12-01	6164.21	5.1	2.72
2022-01-01	5072.49	13.8	2.69
2022-02-01	6572.86	13.8	2.68
2022-03-01	5594.58	13.8	2.67
2022-04-01	5266.91	13.8	2.64
2022-05-01	3891.14	13.8	2.57
2022-06-01	3680.17	13.8	2.55
2022-07-01	6036.24	13.8	2.55
2022-08-01	4236.48	13.8	2.56
2022-09-01	3529.54	13.8	2.56
2022-10-01	4580.24	13.8	2.88
2022-11-01	4295.73	13.8	2.96
2022-12-01	4967.01	13.8	3.04

month_id	Inflatie(de la o luna la alta)	net_average_salary	\
2019-01-01	0.83	2936.0	
2019-02-01	0.79	2933.0	
2019-03-01	0.49	3075.0	
2019-04-01	0.61	3115.0	
2019-05-01	0.46	3101.0	
2019-06-01	-0.23	3142.0	
2019-07-01	-0.20	3119.0	
2019-08-01	0.06	3044.0	
2019-09-01	0.09	3082.0	
2019-10-01	0.43	3116.0	
2019-11-01	0.23	3179.0	
2019-12-01	0.42	3340.0	
2020-01-01	0.41	3189.0	
2020-02-01	0.25	3202.0	
2020-03-01	0.50	3294.0	
2020-04-01	0.26	3182.0	
2020-05-01	0.05	3179.0	
2020-06-01	0.08	3298.0	
2020-07-01	0.00	3372.0	
2020-08-01	-0.05	3275.0	
2020-09-01	-0.14	3321.0	
2020-10-01	0.22	3343.0	
2020-11-01	0.13	3411.0	
2020-12-01	0.34	3620.0	
2021-01-01	1.33	3395.0	
2021-02-01	0.41	3365.0	
2021-03-01	0.38	3547.0	
2021-04-01	0.45	3561.0	
2021-05-01	0.53	3492.0	
2021-06-01	0.27	3541.0	
2021-07-01	0.97	3545.0	
2021-08-01	0.24	3487.0	
2021-09-01	0.84	3517.0	
2021-10-01	1.78	3544.0	
2021-11-01	0.00	3645.0	

2021-12-01	0.71	3879.0
2022-01-01	1.48	3698.0
2022-02-01	0.58	3721.0
2022-03-01	1.88	3937.0
2022-04-01	3.74	3967.0
2022-05-01	1.18	3928.0
2022-06-01	0.76	3977.0
2022-07-01	0.89	3975.0
2022-08-01	0.56	3933.0
2022-09-01	1.33	4003.0
2022-10-01	1.28	4008.0
2022-11-01	1.25	4141.0
2022-12-01	0.37	4141.0

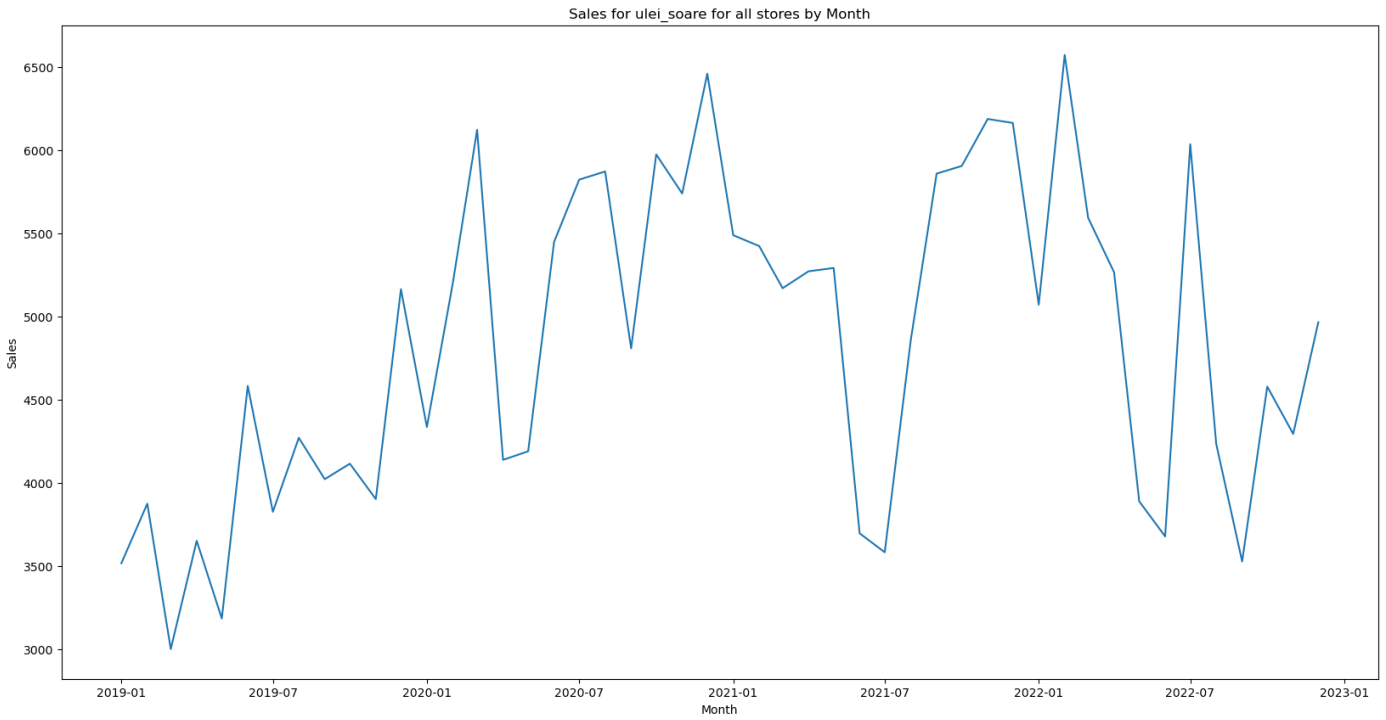
month_id	IPC Marfuri alimentare (%)	IPC Marfuri nealimentare (%)	\
2019-01-01	101.30	100.63	
2019-02-01	101.27	100.57	
2019-03-01	100.48	100.54	
2019-04-01	100.68	100.53	
2019-05-01	100.79	100.20	
2019-06-01	99.53	99.76	
2019-07-01	99.34	99.98	
2019-08-01	99.71	100.22	
2019-09-01	99.91	100.13	
2019-10-01	100.70	100.32	
2019-11-01	100.43	100.12	
2019-12-01	100.84	100.28	
2020-01-01	100.99	100.02	
2020-02-01	100.63	99.94	
2020-03-01	101.46	99.91	
2020-04-01	101.27	99.67	
2020-05-01	100.34	99.82	
2020-06-01	99.62	100.28	
2020-07-01	99.55	100.19	
2020-08-01	99.59	100.08	
2020-09-01	99.45	99.99	
2020-10-01	100.11	100.31	
2020-11-01	99.92	100.29	
2020-12-01	100.29	100.51	
2021-01-01	100.63	102.24	
2021-02-01	100.46	100.47	
2021-03-01	100.37	100.46	
2021-04-01	100.45	100.47	
2021-05-01	101.10	100.28	
2021-06-01	100.25	100.29	
2021-07-01	99.71	102.02	
2021-08-01	99.95	100.34	
2021-09-01	100.95	100.73	
2021-10-01	101.06	102.78	
2021-11-01	100.73	99.47	
2021-12-01	100.84	100.74	
2022-01-01	101.15	101.73	
2022-02-01	101.96	99.70	
2022-03-01	102.54	101.86	
2022-04-01	102.56	105.45	
2022-05-01	101.73	101.00	
2022-06-01	100.62	100.92	
2022-07-01	100.92	100.87	
2022-08-01	101.82	99.81	
2022-09-01	101.72	101.28	
2022-10-01	102.29	100.80	
2022-11-01	101.54	101.03	
2022-12-01	101.26	99.68	

	IPC Servicii (%)

```
month_id
2019-01-01            100.57
2019-02-01            100.55
2019-03-01            100.40
2019-04-01            100.72
2019-05-01            100.55
2019-06-01            100.17
2019-07-01            100.10
2019-08-01            100.25
2019-09-01            100.27
2019-10-01            100.25
2019-11-01            100.15
2019-12-01            100.12
2020-01-01            100.43
2020-02-01            100.39
2020-03-01            100.35
2020-04-01            100.00
2020-05-01            100.11
2020-06-01            100.34
2020-07-01            100.31
2020-08-01            100.21
2020-09-01            100.20
2020-10-01            100.22
2020-11-01            100.07
2020-12-01            100.04
2021-01-01            100.25
2021-02-01            100.20
2021-03-01            100.24
2021-04-01            100.40
2021-05-01            100.27
2021-06-01            100.23
2021-07-01            100.39
2021-08-01            100.43
2021-09-01            100.96
2021-10-01            100.42
2021-11-01            100.20
2021-12-01            100.42
2022-01-01            101.37
2022-02-01            100.60
2022-03-01            100.67
2022-04-01            100.94
2022-05-01            100.61
2022-06-01            100.54
2022-07-01            100.88
2022-08-01            100.37
2022-09-01            100.72
2022-10-01            100.70
2022-11-01            101.31
2022-12-01            100.67
```

In [125…

```python
# Line graph for apa_plata for all stores
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index.get_level_values(0), filter_df_sales['sales'])
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales for ' + product + ' for all stores by Month')
plt.show()
```



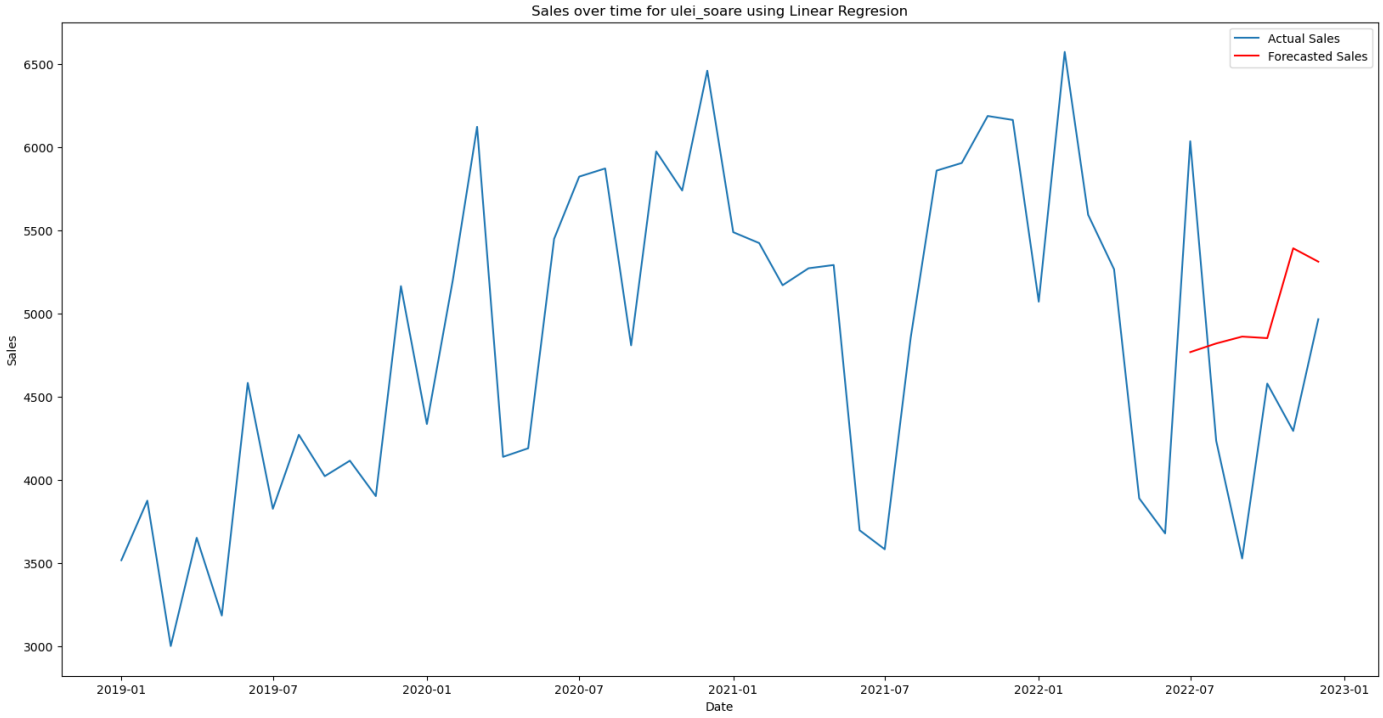
LR - ulei_soare

In [126…

```
# Get the last 6 months of data
last_6_months = filter_df_sales.tail(6)

# Split the data into train and test sets
X_train = filter_df_sales.drop('sales', axis=1)
y_train = filter_df_sales['sales']
X_test = last_6_months.drop('sales', axis=1)
y_test = last_6_months['sales']

# Create a linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the error
lr_mae = mean_absolute_error(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)

print("Mean Absolute Error: ", lr_mae)
print("Mean Squared Error: ", lr_mse)
print("Root Mean Squared Error:", lr_rmse)

all_sales = filter_df_sales['sales']

# Get the index of the last 6 months of data
last_6_months_index = last_6_months.index

# Plot the entire period of sales data
plt.figure(figsize=(20,10))
plt.plot(all_sales)

# Make predictions on the last 6 months of data
```

```
y_pred = model.predict(X_test)

# Plot the forecasted sales on the same chart
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Linear Regresion')
plt.legend(['Actual Sales', 'Forecasted Sales'])

plt.show()
```

```
Mean Absolute Error:  816.8381961158416
Mean Squared Error:  853727.7828789075
Root Mean Squared Error: 923.9739081158664
```



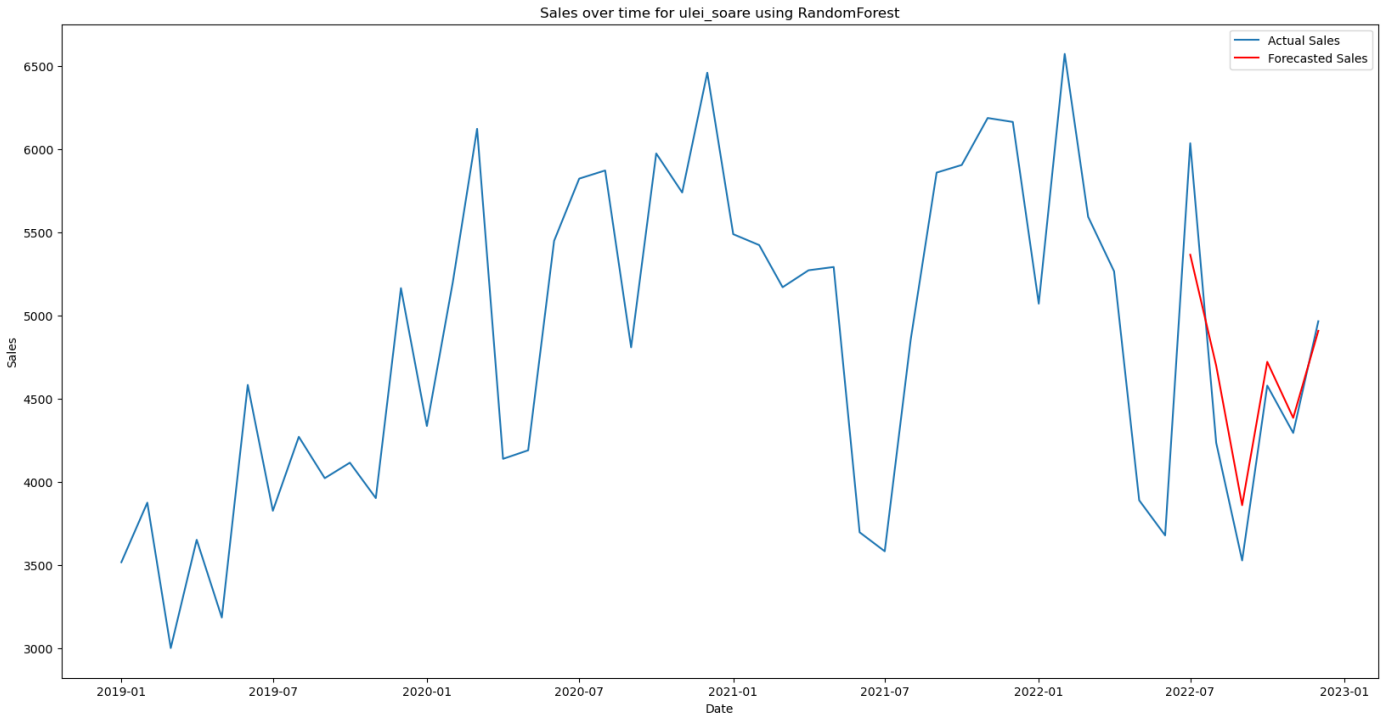
RF - ulei_soare

In [127…

```
last_6_months = filter_df_sales.tail(6)

X_train = filter_df_sales.drop(columns=["sales"])
y_train = filter_df_sales["sales"]
X_test = last_6_months.drop(columns=["sales"])
y_test = last_6_months["sales"]

# Create a random forest model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Use the model to make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate the model's accuracy
accuracy = rf.score(X_test, y_test)
print("Accuracy: ", accuracy)

# Calculate error
rf_mae = mean_absolute_error(y_test, y_pred)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rf_mse)
```

```
print("Mean Absolute Error: ", rf_mae)
print("Mean Squared Error: ", rf_mse)
print("Root Mean Squared Error:", rf_rmse)

# Line graph for the forcast
plt.figure(figsize=(20,10))

plt.plot(all_sales)
plt.plot(last_6_months_index, y_pred, 'r')

plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using RandomForest')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
Accuracy:  0.7743501596526743
Mean Absolute Error:  292.5934166666655
Mean Squared Error:  134191.80361681202
Root Mean Squared Error: 366.32199444861624
```



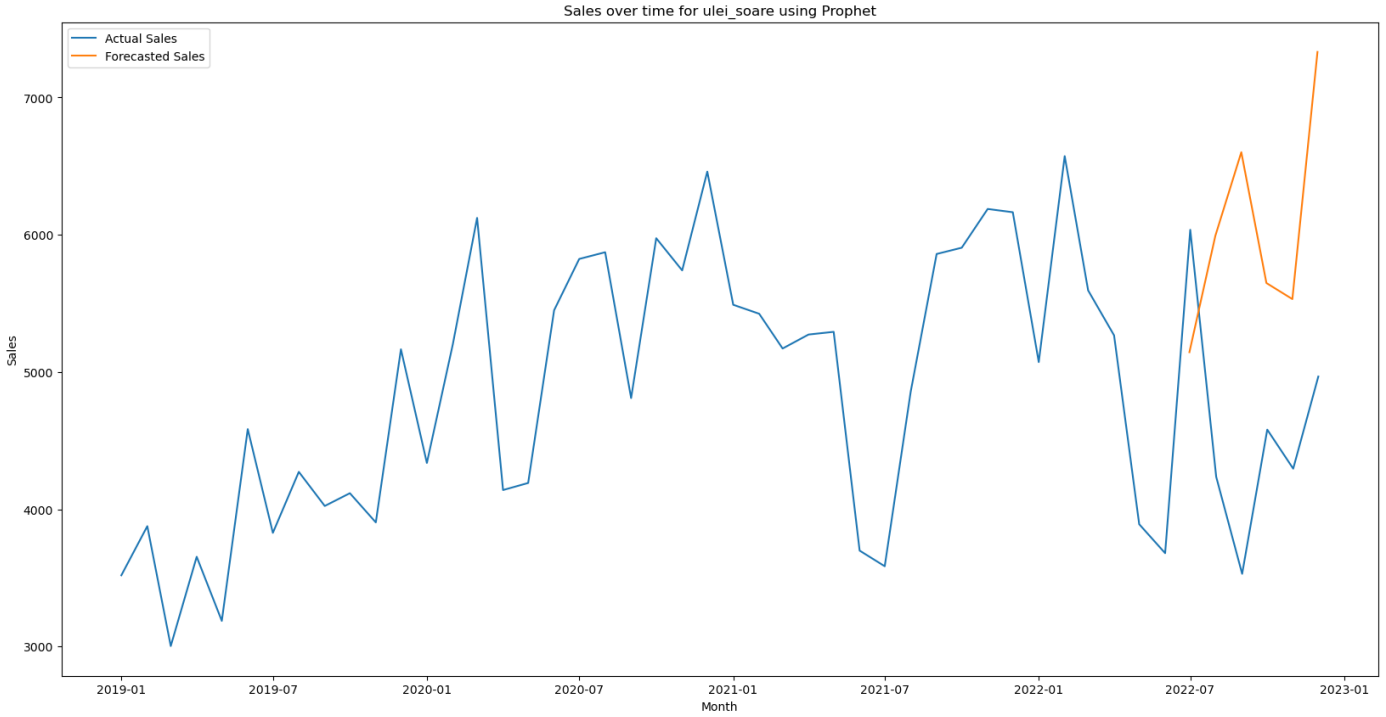
PR - ulei_soare

In [128…

```
# create a copy of the dataframe with a column named 'ds' with the index values
filter_df_sales_prophet = filter_df_sales.reset_index().rename(columns={'month_id': 'ds'
filter_df_sales_prophet = filter_df_sales_prophet.rename(columns={'sales':'y'})

train_data = filter_df_sales_prophet[:-6]
test_data = filter_df_sales_prophet[-6:]

model = Prophet()

# Fit the model on the data
model.fit(train_data)

# Create a dataframe to hold the forecast
future_data = model.make_future_dataframe(periods=6, freq='m')

# Make predictions
forecast = model.predict(future_data)
forecast = forecast.tail(6)
```

```
# Calculate error
pr_mae = mean_absolute_error(test_data['y'], forecast[-6:]['yhat'])
pr_mse = mean_squared_error(test_data['y'], forecast[-6:]['yhat'])
pr_rmse = np.sqrt(pr_mse)
print("Mean Absolute Error:", pr_mae)
print("Mean Squared Error:", pr_mse)
print("Root Mean Squared Error:", pr_rmse)

# Plot the forecast
plt.figure(figsize=(20,10))
plt.plot(filter_df_sales.index, filter_df_sales['sales'], label='Actual Sales')

# Plot the predicted values for the last six months
plt.plot(forecast.ds, forecast.yhat, label='Predicted Sales')

plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales over time for ' +product+ ' using Prophet')
plt.legend(['Actual Sales', 'Forecasted Sales'])
plt.show()
```

```
21:35:31 - cmdstanpy - INFO - Chain [1] start processing
21:35:31 - cmdstanpy - INFO - Chain [1] done processing
Mean Absolute Error: 1731.4713258090258
Mean Squared Error: 3596297.0290305763
Root Mean Squared Error: 1896.3905265083392
```



In [129…

```
# Add the errors from all methods to the DataFrame
new_row_df =  pd.DataFrame({'product': [product], 'LR-MAE': [lr_mae], 'LR-MSE': [lr_mse]
errors_df = pd.concat([errors_df, new_row_df], ignore_index=True)
print(errors_df)
```

	product	LR-MAE	LR-MSE	LR-RMSE	RF-MAE	\
0	apa_plata	887.015829	988186.848307	994.075877	377.426267	
1	apa_minerala	44.324535	2453.366899	49.531474	10.533100	
2	bere	283.109792	95844.002658	309.586826	65.444150	
3	cafea	287.818261	94385.213013	307.221765	134.035500	
4	carnati	264.936590	99377.947534	315.242680	155.934000	
5	frigider	13.054626	245.441803	15.666582	8.523900	
6	lapte	490.769164	269944.557868	519.561890	193.309633	
7	legume	317.334317	168366.068324	410.324345	124.748417	

```
8   masina_de_spalat   18.592357     606.867326   24.634677    8.862467
9               orez  376.601361  165639.594918  406.988446   98.474217
10              oua   725.835859  635033.703803  796.890020  307.595917
11            paste   181.288571   45954.816231  214.370745   79.459933
12            peste   175.548619   51772.454330  227.535611   98.965167
13            pizza   333.543153  167517.484340  409.288999  100.119867
14       racoritaore  264.273274  144927.859319  380.693918  108.331267
15         servetele  302.493481   97426.860327  312.132761  116.845683
16        ulei_soare  816.838196  853727.782879  923.973908  292.593417

           RF-MSE     RF-RMSE       PR-MAE        PR-MSE      PR-RMSE
0   200759.770117  448.062239  1107.910649  1.661801e+06  1289.108561
1      153.080170   12.372557  1703.258881  1.172504e+07  3424.185038
2     6221.823280   78.878535   437.297885  2.623325e+05   512.184061
3    22921.110989  151.397196   554.567175  3.976641e+05   630.606141
4    33523.190407  183.093393   370.439785  1.702760e+05   412.645094
5      105.584959   10.275454    34.075515  1.834104e+03    42.826436
6    48061.717358  219.229828   968.181840  1.101085e+06  1049.325744
7    23345.264739  152.791573   605.913460  4.861077e+05   697.214275
8      139.974418   11.831078    40.934713  2.225668e+03    47.176986
9    17005.647771  130.405705   562.437890  5.253923e+05   724.839518
10  114084.843273  337.764479  1055.734550  1.583208e+06  1258.255738
11    9798.345174   98.986591   159.511806  3.246713e+04   180.186386
12   13836.018312  117.626605   517.236332  3.505682e+05   592.087976
13   18062.917925  134.398355   454.191376  4.105910e+05   640.773742
14   22128.190513  148.755472   638.867845  8.013108e+05   895.159651
15   17999.422942  134.161928   363.006077  2.027851e+05   450.316611
16  134191.803617  366.321994  1731.471326  3.596297e+06  1896.390527
```

Error visualization

The following table shows the main error indicators (MAE, MSE, RMSE) by each product and ML algorithm. The lowest values per line are highlighted. Considering these highlighted values,the random forest algorithm generates the best prediction. However, the lowest accuracy value registered with this algorithm is approximately 40% for the rice product category. The highest accuracy level is 97% and is represented by the beer product. The algorithm with the lowest performance in the prediction is the Prophet algorithm.

In [130…
```
errors_df.style.apply(lambda x: ['background: green' if x.name == 'RF-MAE' else '' for i
errors_df.style.background_gradient(subset='RF-MAE')
```

Out[130]:

	product	LR-MAE	LR-MSE	LR-RMSE	RF-MAE	RF-MSE	RF-RMSE
0	apa_plata	887.015829	988186.848307	994.075877	377.426267	200759.770117	448.062239
1	apa_minerala	44.324535	2453.366899	49.531474	10.533100	153.080170	12.372557
2	bere	283.109792	95844.002658	309.586826	65.444150	6221.823280	78.878535
3	cafea	287.818261	94385.213013	307.221765	134.035500	22921.110989	151.397196
4	carnati	264.936590	99377.947534	315.242680	155.934000	33523.190407	183.093393
5	frigider	13.054626	245.441803	15.666582	8.523900	105.584959	10.275454
6	lapte	490.769164	269944.557868	519.561890	193.309633	48061.717358	219.229828
7	legume	317.334317	168366.068324	410.324345	124.748417	23345.264739	152.791573
8	masina_de_spalat	18.592357	606.867326	24.634677	8.862467	139.974418	11.831078
9	orez	376.601361	165639.594918	406.988446	98.474217	17005.647771	130.405705
10	oua	725.835859	635033.703803	796.890020	307.595917	114084.843273	337.764479

11	paste	181.288571	45954.816231	214.370745	79.459933	9798.345174	98.986591
12	peste	175.548619	51772.454330	227.535611	98.965167	13836.018312	117.626605
13	pizza	333.543153	167517.484340	409.288999	100.119867	18062.917925	134.398355
14	racoritaore	264.273274	144927.859319	380.693918	108.331267	22128.190513	148.755472
15	servetele	302.493481	97426.860327	312.132761	116.845683	17999.422942	134.161928
16	ulei_soare	816.838196	853727.782879	923.973908	292.593417	134191.803617	366.321994

In [ ]: