VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

MACHINE LEARNING - CO3117

ANALYSING MACHINE LEARNING AND DEEP LEARNING METHODS FOR TIME SERIES FORECASTING

Semester 231

Lecturer: NGUYỄN ĐỨC DŨNG
Students: Trần Công Huy Hoàng - 2052482.
Nguyễn Duy Bảo - 2052399.

HO CHI MINH CITY, JANUARY 2024

Abstract

Time series forecasting is a crucial capability for many real-world decision-making processes. Modern statistical and machine learning approaches offer viable and accurate methodologies for time series prediction tasks.

This report provides a comprehensive empirical evaluation of both classical machine learning and deep neural network architectures on time series forecasting. Six machine learning algorithms, including linear regression, K-nearest neighbors (KNN) regression, random forest regression, XGBoost, LightGBM, and CatBoost, as well as three deep neural network architectures, specifically multi-layer perceptron (MLP), long short-term memory networks (LSTM), and gated recurrent units (GRU), are analyzed. These models encompass both traditional and leading-edge techniques for modeling time-sequenced data. We conduct a comparative evaluation of predictive accuracy and computational performance on the "Walmart recruiting - store sales forecasting" dataset and examine predictive accuracy and computational efficiency to determine the strengths and weaknesses of the examined predictive models. The key results indicate that ensemble tree-based methods and long short-term neural architectures achieve top performance.

Overall, the empirical insights construct a comprehensive applied perspective on leveraging statistical and deep learning methodologies for data-driven time series prediction.

Project's Github: `https://github.com/hontrn9122/asm_machine_learning`

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Time series forecasting plays an integral role in a massive array of real-world applications, spanning domains such as economics, finance, production operations, meteorology, and many more. The ability to accurately predict future values in a time series allows organizations and researchers to make data-driven decisions regarding inventory planning, revenue projections, infrastructure capacity planning, targeted marketing, anomaly detection, and climate change analysis to name some examples. As such, developing effective computational methods for time series forecasting constitutes a problem of substantial practical importance.

Both classical machine learning techniques as well as modern deep neural network architectures have shown promising results in modeling temporal data. However, applied research often focuses on comparing ML-based only or DL-based only without a detailed cross-comparison between the two methods.

To address these knowledge gaps, in this assignment, we provide a rigorous, comprehensive benchmark between major machine learning and deep learning techniques for time series analysis using the "Walmart recruiting - store sales forecasting" dataset with a significant number of records. By evaluating method performance across metrics such as predictive accuracy and scalability, we construct applied insights around leveraging both traditional and cutting-edge time series modeling techniques on different training conditions. This head-to-head empirical comparison yields a crucial reference for industry and academic practitioners regarding deploying the most effective machine learning solutions for their temporal prediction challenges.

1.2 Goals

This research aims to provide a thorough, empirical performance analysis and comparison of leading machine learning and deep learning techniques on time series forecasting tasks. Specifically, we benchmark six prominent statistical and machine learning methods - linear regression, K-nearest neighbors (KNN) regression, random forest regression, XGBoost, LightGBM, and CatBoost -

alongside four popular deep neural network architectures - multi-layer perceptrons (MLP), long short-term memory (LSTM) networks, and gated recurrent units (GRU). Leveraging the "Walmart recruiting - store sales forecasting" dataset, we evaluate each time series modeling approach over two primary dimensions:

- **Predictive Accuracy:** Measuring out-of-sample forecasting performance across datasets using metrics such as RMSE, MAPE, sMAPE, MAE, R2, and execution time.
- **Computational Efficiency:** Quantifying model training and inference times, allowing comparison of scalability across small and large time series modeling problems.

Through this extensive empirical analysis, our goal is to determine the overall best-performing machine learning solutions for time series forecasting as well as examine which methods excel given problem constraints such as dataset size and available computation time. These insights fill a gap in applied time series analytics guidance for practitioners. Our findings construct both focused conclusions per modeling technique as well as higher-level practical recommendations for selecting appropriate classes of data-driven forecasting methods given characteristics of time series problems. Overall, this work aims to advance the understanding of modern time series analysis techniques for real-world application.

1.3 Scopes

Data Scope

- Leverage open-source time series dataset.
- Stimulate different training conditions by different training/validating/testing splits for out-of-sample evaluation.

Methods Scope

- Cover six standard machine learning algorithms - linear regression, KNN regression, random forest regression, XGBoost, LightGBM, CatBoost.

- Cover four leading deep neural network architectures - MLPs, RNNs, LSTMs, GRUs.
- Implement common time series data preprocessing techniques.
- Tune hyperparameters for each model during training.

Evaluation Scope

- Compare predictive accuracy on split test sets per series via RMSE, MAPE etc.
- Measure and compare training computation time.
- Visualize key performance differences across techniques.

Analysis Scope

- Statistically test for significant performance differences.
- Identify the best overall algorithms and top techniques per data domain.
- Study model sensitivity to time series length, frequency, stationarity.

1.4 Report Structure

Our project is structured across four chapters, each serving a distinct purpose.

Chapter 1 establishes the groundwork by presenting the motivation behind the problem, clearly articulating the goals and defining the project's scope.

In Chapter 2, we conduct an extensive review of the preliminary knowledge of the model we use for a comprehensive understanding of the project.

In Chapter 3, we introduce the dataset, metrics for evaluation, and the detailed experiment result.

The final chapter, Chapter 4, serves as a comprehensive summary of our project work, encapsulating the essential findings and insights.

2 Theoretical Background

2.1 Machine Learning Models

In the first part of this section, we introduce six different classic machine learning models that we use for our analysis - linear regression, K-nearest neighbors (KNN) regression, random forest regression, XGBoost, LightGBM, and CatBoost.

2.1.1 Linear Regression Model

Linear regression is one of the most basic and interpretable machine learning models in time series forecasting. In linear regression, the target variable is modeled as a linear combination of the input variables, representing the relationship using a straight-line equation. The model coefficients are estimated by minimizing the residual sum of squares between predicted and actual historical values.

For time series forecasting, in particular, linear regression models the future value of the series as a weighted sum of previous lagged values of the series. This exploits autocorrelation, where a time series correlates with past values. Additional related time series can also be included as predictor variables. Using lagged values rather than direct previous values also models inertia where changes manifest over time.

Linear regression explicitly fits trend and seasonal components and effectively handles smoothed, relatively stable series with minimal outliers. It is a simple yet strong baseline forecasting model before exploring more complex nonlinear alternatives. Weaknesses include the inability to adapt to nonlinearity, data outliers, or structural breaks in the time series. It also risks overfitting without proper regularization of the coefficient parameters.

Overall, linear regression delivers a low-barrier entry point for forecasters to evaluate the predictive capacity within their time series. The interpretation of variable importance is more direct compared to nonlinear black-box models. Linear regression strikes a pragmatic balance between accuracy, transparency, and ease of implementation.

2.1.2 K-Nearest Neighbors (KNN) Regression Model

Instances-based procedures foregoing explicit parametric assumption specification prospectively constitute efficacious time series extrapolation techniques – particularly amidst multidimensionality. The K-Nearest Neighbors (KNN) algorithm approximates by detecting historical instances most analogous to query input vectors. Time series realizations compose inputs with future values representing targets. Analogues are established using distance calculations between lagged endogenous observations and exogenous indicators.

KNN automatically adapts to inherent data patterns absent formal modeling, intuitively suiting timeseries. Optimal neighbor numerosity balances noise and bias. Weighting enhances recent prominence. Query computational inefficiencies necessitate optimization. KNN flexibility balances linear techniques' restrictive assumptions and complex neural networks' opaque infrastructures when approximating multidimensional interdependencies [Bontempi et al., 2013].

This analysis implements multivariate KNN exploitation of store metadata and prior sales to construct responsive retail sales forecasts. The ability to automatically recognize predictive temporal and indicator interactions underpins an adaptable modeling approach without solid assumptions that may falter amidst nonlinear multidimensional retail data.

2.1.3 Random Forest Regression Model

Machine learning methodologies for probabilistic forecasting of multivariate time series remain contingent on balancing model flexibility and overfitting defenses. Ensemble approaches combining discrete base predictors offer an efficacious middle ground. Random forest constitutes an ensemble methodology devised by Breiman [2001] aggregating predictions from numerous de-correlated decision tree models trained on bootstrap instance samples considering random feature subsets when deriving splits.

Trees intrinsically capture interactions and non-linear relationships within heterogeneous indicators unrecognizable to linear techniques like ARIMA. Bagging and feature sampling introduce diversity, improving generalizability. Hyperparameter tuning constrains flexibility. Random forest demonstrates consistent accuracy gains over the singular decision and regression trees. Time series traits like lag effects, seasonality, structural breaks, and outlier resiliently fit the functional flexibility and ensemble smoothing effects.

This analysis leverages the random forest technique for constructing weekly Walmart sales forecasting systems exploiting store metadata and prior sales trajectories. The method balances modeling nonlinear multivariate interdependencies, overtraining susceptibility, computational demands, and result transparency - appropriate for accurate enterprise-level retail projection.

2.1.4 Extreme Gradient Boosting Machine (XGBoost) Regression Model

Extreme Gradient Boosting (XGBoost) by Chen and Guestrin [2016] is an optimized, scalable implementation of gradient-boosted decision trees, a powerful machine learning approach built on ensembling weak learners. It has become a popular regression choice for time series forecasting problems.

The algorithm trains decision trees sequentially, each new tree correcting errors in those preceding it to incrementally improve prediction. Trees split input features recursively to minimize squared error. Regularization helps avoid overfitting. Key advantages over basic tree regressors are managing complexity and computational optimization to enhance scale, speed and accuracy.

For forecasting, XGBoost models inherently tackle interaction effects, non-linearity, variable lag effects, and irregular patterns, which are hard for linear models. Tree complexity tuning balances under and overfitting. The boosted structure suits time dependency well, adapting to trends and multiple seasonalities. XGBoost can ingest a combinatoric mix of historical target values, lagged targets, moving averages, and other explanatory variables.

Overall, XGBoost brings world-class nonlinear regression capability to time series problems. The regularization and computational properties provide practical scalability. Compared to neural networks, XGBoost retains some model interpretability while handling many complex forecasting scenarios beyond linear technique capacity. Its flexibility and performance make XGBoost a routinely competitive forecasting approach if not the best solution.

2.1.5 Light Gradient Boosting Machine (LightGBM)

Gradient boosting denotes an ensemble technique whereby subsequent models attempt to compensate for predecessors' deficiencies incrementally to minimize loss. LightGBM constitutes an accelerated, distributed gradient boosting framework concentrating on efficiency, scalability, and accuracy advanced by Ke et al. [2017]. Optimization commences from leaf nodes rather than level-wise tree traversal, economizing computation. Additionally, histogram-based algorithms summarize continuous features into discrete bins, accelerating training. Exclusive focus on cis-learners as opposed to trans-learners obviates maintaining candidate scores. Instances are partitioned randomly rather than by level, neutering overfitting.

Empirical evaluations validate LightGBM's speed, memory utilization and accuracy gains over contemporary boosters like XGBoost and baseline GBMs. Hyperparameter tuning checks complexity. The combination of leaf-wise tree growth, histogram-based feature binning, cis-objective functions, GPU compatibility, and distributed computing provides a high-performance gradient boosting package suited for problems involving high dimensionality, numerous data instances and real-time predictions. As such, LightGBM constitutes a framework appropriately matching the analytical and computational stipulations imposed by multivariate production line data sequencing.

This investigation applies LightGBM exploiting store department indicators and prior sales to construct responsive weekly retail forecasting systems. The flexibility and optimization underpins impactful multivariate time series modeling.

2.1.6 CatBoost Algorithm

Categorical variables represent ubiquitous fixtures within real-world datasets, which conventional modeling techniques rebut via high-cardinality one-hot encoding swelling dimensionality and skirting inter-dependencies. CatBoost constitutes an innovative open-source boosting algorithm devised by Prokhorenkova et al. [2018] imparting intrinsic categorical variable support absent transformations. The method applies ordered boosting rather than symmetric or oblivious approaches to curtail prediction deviation. Categorical splits are determined in the partitioning process itself rather than initial pre-processing, retaining associated statistical relationships.

Evaluations demonstrate consistent accuracy improvements over functional boosting alternatives like XGBoost and LightGBM across high-cardinality categorical modeling tasks, including click predictions and document ranking. Hyperparameter tuning constraints like depth and learning rate operate similarly. Key benefits include automated embedding of categorical variable interactions without swelling vector size as branches enumerate category combinations. The CatBoost framework delivered the championship solution in the Couplet machine learning competition. This analysis harnesses CatBoost natively conjoining categorical department indicators and numerically encoded sensor production analytics to construct responsive, generalizable multivariate forecasting systems.

2.2 Deep Learning Networks

In this second part of the section, we provide an overview of three different deep learning networks we used in the next section - multi-layer perceptrons (MLP), long short-term memory (LSTM) networks, and gated recurrent units (GRU).

2.2.1 Multi-Layer Perceptron (MLP) Network

Multi-layer perceptrons (MLP) constitute one of the most fundamental and established architectures within the domain of artificial neural networks. Proposed in research by Rumelhart et al. [1985], MLPs refer to feedforward neural network models comprised of an input layer, one or more hidden layers consisting of nonlinear activation functions, and an output layer. They leverage backpropagation to adjust the connection weights between network nodes and minimize losses during training. Their hierarchical structure enables MLPs to learn sophisticated feature representations and model complex relationships between input and output variables. Owing to these capabilities, prior research has explored the utility of MLP neural networks for predicting future points in time series across use cases, including electricity load forecasting [Park et al., 1991], traffic flow prediction [Smith and Demetsky, 1994], financial time series forecasting [Saad et al., 1998], etc. In this work, we build two simple variances of standard feedforward MLP with three and six layers, respectively, for comparison with more recent advanced DL networks specialized for temporal data.

2.2.2 Recurrent Neural Network (RNN)

Neural networks that recursively apply the same weights over a sequence are called recurrent networks. Rather than feedforward varieties, recurrent networks employ an internal memory, rendering them intrinsically well-suited to model sequential dependencies [Rumelhart et al., 1985]. This self-referencing facilitates discovering dynamic statistical patterns within trails of inputs unfolding over time.

A key benefit involves inheriting implicit temporal representations absent from preceding layers. Standard RNN configurations entail a single hidden layer computing activation outputs recursively applied as inputs into the next sequence step. Gradient descent tunes network weights to minimize predictive error over multiple epochs. Once trained, the internal state theoretically encodes informative trends regarding historical progression proper for extrapolating future tendencies.

However, basic RNNs falter in capturing long-term temporal interactions in practice due to exploding and vanishing gradients over lengthening backward propagation sequences [Hochreiter, 1998]. Numerous architectural variants, including Long Short-Term Memory Networks (LSTM) and Gated Recurrent Units (GRU), overcome this limitation through purpose-built memory cells that isolate relevant signals. Hence, in this project, we decided to analyze only LSTM and GRU.

2.2.3 Long Short-Term Memory (LSTM

Network) Long Short-Term Memory (LSTM) models underpin a highly flexible approach to modeling sequential time series data. LSTM encompasses a specialized recurrent architecture explicitly developed to capture long-range temporal dependencies within time series [Hochreiter and Schmidhuber, 1997]. Conventional feedforward networks struggle to model such temporal contexts. Equally, basic recurrent networks grapple with vanishing and exploding gradients over lengthy sequences [Bengio et al., 1994].

To overcome these challenges, LSTM introduces a memory cell encasing four neural network layers interacting via multiplication gates. An input gate controls cell updates. The forget gate filters needless historical signals. An output gate disseminates relevant latent signals. These components allow the LSTM cell to discard outdated signals and isolate pertinent temporal contexts necessary for anticipating future sequence developments. Multiple LSTM layers stack to deepen sequential representation learning.

Configuring optimal network topology and hyperparameter settings poses the central implementation challenge. Nonetheless, empirical assessments substantiate the exceptional capabilities of properly-tuned LSTM for multivariate forecasting and sequence labeling undertakings contingent on historical signals [Brownlee, 2018]. This investigation harnesses LSTM to extract informative multivariate and temporal predictive insights from production line sensor data.

2.2.4 Gated Recurrent Unit (GRU) Network

Gated Recurrent Unit (GRU) constitutes a gating mechanism variation of conventional recurrent networks proposed by Cho et al. [2014] to emulate Long Short-Term Memory (LSTM) architectures while minimizing computational overhead. As opposed to four distinct neural network layers used in LSTM units, GRU incorporates only two gating units: an update gate and a reset gate. The reset gate conditions the activation function input, while the update gate controls how much of the previous state to maintain.

Internally, the candidate activation employs the reset gate to regulate the flow of historical state signals. The update gate then condenses the candidate activation and past state into a final memory output. These components equip GRU to moderate the prominence of previous trajectory signals, thereby accentuating pertinent sequential dependencies – much like LSTM units. Empirical assessments of GRU configurations indicate competitive performance to LSTM across an array of sequence modeling tasks, including neural machine translation and speech recognition, with the benefit of reduced parameters to estimate [Chung et al., 2014].

This investigation has chosen GRU networks for performance analysis of multivariate production sensor data as they balance representational adaptability and computational tractability. To achieve responsive sequence learning, the simplified GRU design requires tuning fewer hyperparameters during optimization relative to more complex LSTM formulations.

3 Experiment

3.1 Walmart sales dataset

The Walmart Recruiting: Store Sales Forecasting dataset contains historical sales data for 45 Walmart stores located in different regions between 2010 and 2012. Each store contains department-level weekly sales data covering general merchandise, fresh products, apparel, and more. Additionally, nominal features are provided, including store and department details, temperature, unemployment rate, and holidays. In total, the structured dataset consists of over 1 million observations with 14 usable fields. The data was initially made public as part of a Kaggle competition centered around developing accurate forecasting models to predict department-wide sales for each store. Competitors were challenged to leverage machine learning, time series analysis, and potentially supplementary data to minimize the root mean squared error over a finite forecast period. The public availability has enabled extensive research experimentation in sales forecasting techniques applicable to multi-store retail corporations.

Some key details about the dataset:

- Sales data at the department level for 45 Walmart stores.
- Weekly data from 2010 - 2012.
- 14 fields including sales, store details, temperature, and holidays.
- Over 1 million rows.
- Originally used for a Kaggle forecasting competition.
- Enables research into multi-store retail sales forecasting techniques.

3.1.1 Data exploration

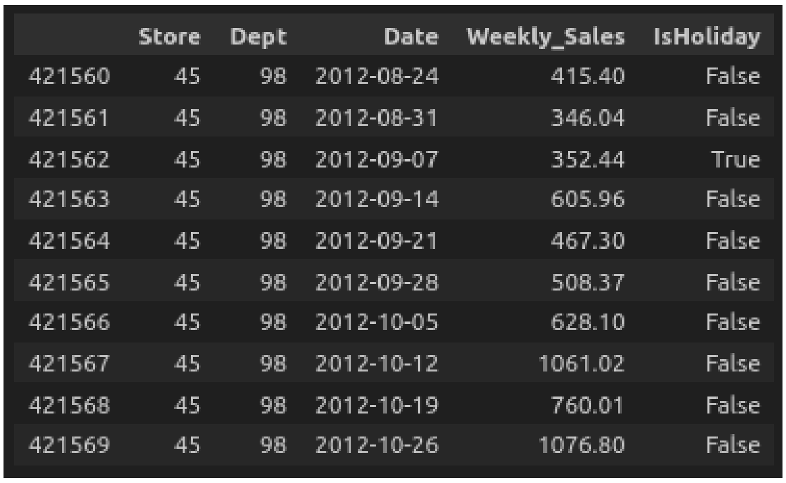
Our data includes:

1. **stores.csv:** This file contains anonymized information about the 45 stores, indicating the type and size of store.
2. **train.csv:** This is the historical training data, which covers to 2010-02-05 to 2012-11-01. Within this file you will find the following fields:
    - Store - the store number
    - Dept - the department number
    - Date - the week
    - Weekly_Sales - sales for the given department in the given store
    - IsHoliday - whether the week is a special holiday week
3. **test.csv:** This file is identical to train.csv, except we have withheld the weekly sales. You must predict the sales for each triplet of store, department, and date in this file.
4. **features.csv:** This file contains additional data related to the store, department, and regional activity for the given dates. It contains the following fields:
    - Store - the store number
    - Date - the week
    - Temperature - average temperature in the region

- Fuel_Price - cost of fuel in the region
- MarkDown1-5 - anonymized data related to promotional markdowns that Walmart is running. MarkDown data is only available after Nov 2011, and is not available for all stores all the time. Any missing value is marked with an NA.
- CPI - the consumer price index
- Unemployment - the unemployment rate
- IsHoliday - whether the week is a special holiday week

3.1.2 Data cleaning

The training set includes 421570 rows of non-null sale records, so we do not have to find the null values to discard them. There are five examples:



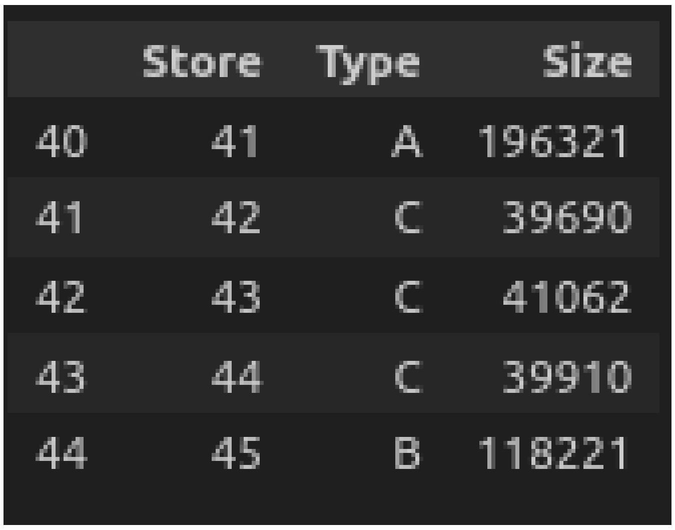
	Store	Dept	Date	Weekly_Sales	IsHoliday
421560	45	98	2012-08-24	415.40	False
421561	45	98	2012-08-31	346.04	False
421562	45	98	2012-09-07	352.44	True
421563	45	98	2012-09-14	605.96	False
421564	45	98	2012-09-21	467.30	False
421565	45	98	2012-09-28	508.37	False
421566	45	98	2012-10-05	628.10	False
421567	45	98	2012-10-12	1061.02	False
421568	45	98	2012-10-19	760.01	False
421569	45	98	2012-10-26	1076.80	False

Figure 3.1: Sale record samples

There are 45 stores with corresponding types and sizes. The data is clean too.

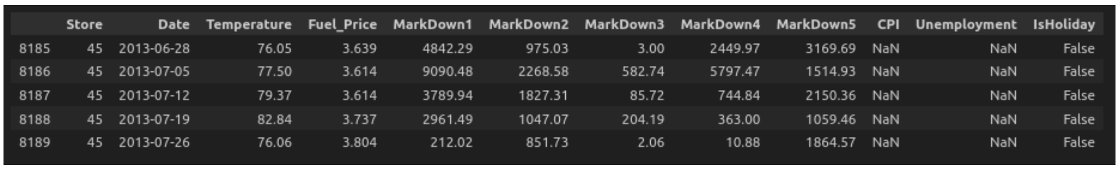
Additional features of stores:

There are null values in CPI and Unemployment attributes, we will replace these with the median of corresponding attributes. In the MarkDown columns, there are many of null values, we replace them with 0. The handling missing values are described below. The left table is the before feature data and the right is the after one.



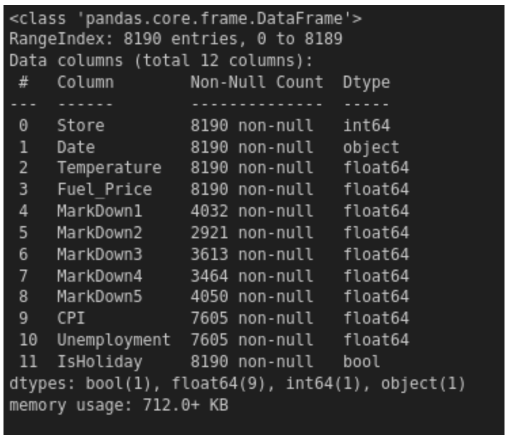
	Store	Type	Size
40	41	A	196321
41	42	C	39690
42	43	C	41062
43	44	C	39910
44	45	B	118221

Figure 3.2: Store data

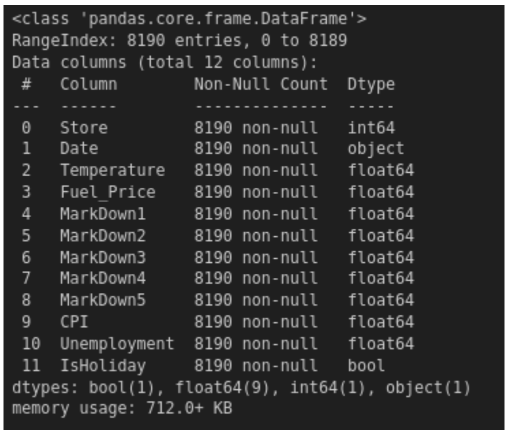


	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsHoliday
8185	45	2013-06-28	76.05	3.639	4842.29	975.03	3.00	2449.97	3169.69	NaN	NaN	False
8186	45	2013-07-05	77.50	3.614	9090.48	2268.58	582.74	5797.47	1514.93	NaN	NaN	False
8187	45	2013-07-12	79.37	3.614	3789.94	1827.31	85.72	744.84	2150.36	NaN	NaN	False
8188	45	2013-07-19	82.84	3.737	2961.49	1047.07	204.19	363.00	1059.46	NaN	NaN	False
8189	45	2013-07-26	76.06	3.804	212.02	851.73	2.06	10.88	1864.57	NaN	NaN	False

Figure 3.3: Store features



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

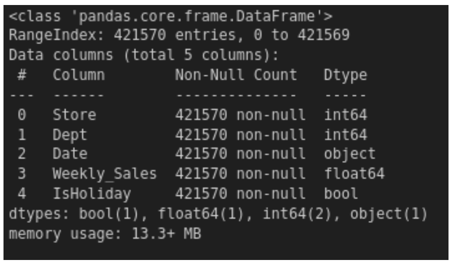


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8190 entries, 0 to 8189
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Store         8190 non-null   int64
 1   Date          8190 non-null   object
 2   Temperature   8190 non-null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

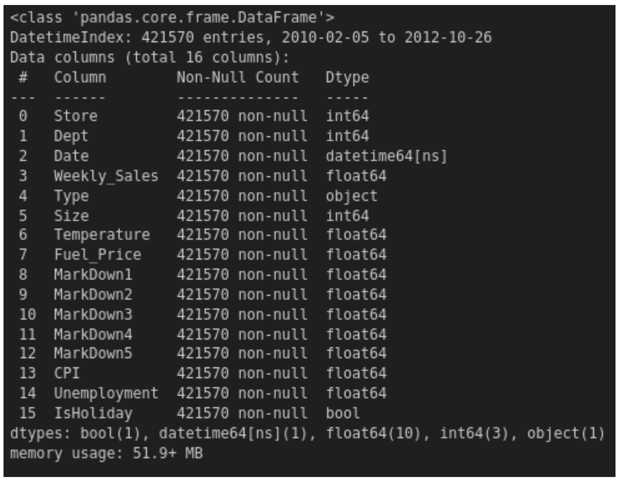
Figure 3.4: Remove null values of additional features

3.1.3 Data transformation

We add the attributes from the stores and features table to the training data corresponding to the Store, Date columns. The transformation goes from left to right.



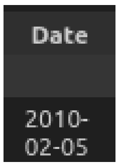
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 421570 entries, 0 to 421569
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Store         421570 non-null  int64
 1   Dept          421570 non-null  int64
 2   Date         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



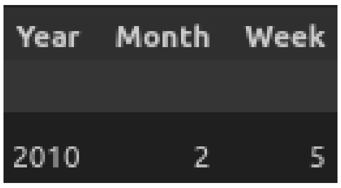
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 421570 entries, 2010-02-05 to 2012-10-26
Data columns (total 16 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Store         421570 non-null  int64
 1   Dept          421570 non-null  int64
 2   Date          421570 non-null  datetime64[ns]
 3   Weekly_Sales  421570 non-null  float64
 4   Type          421570 non-null  object
 5   Size          421570 non-null  int64
 6   Temperature   421570 non-null  float64
 7   Fuel_Price    421570 non-null  float64
 8   MarkDown1     421570 non-null  float64
 9   MarkDown2     421570 non-null  float64
 10  MarkDown3     421570 non-null  float64
 11  MarkDown4     421570 non-null  float64
 12  MarkDown5     421570 non-null  float64
 13  CPI           421570 non-null  float64
 14  Unemployment  421570 non-null  float64
 15  IsHoliday     421570 non-null  bool
dtypes: bool(1), datetime64[ns](1), float64(10), int64(3), object(1)
memory usage: 51.9+ MB
```

Figure 3.5: Merge additional data

The date data is not in the format that we can train the model for predicting. We will do the Datetime Feature Engineering for the date data. Specifically, the date will be separated into Year, Month, Week. The generated data is in the right, which comes from the left data.



Date
2010-02-05

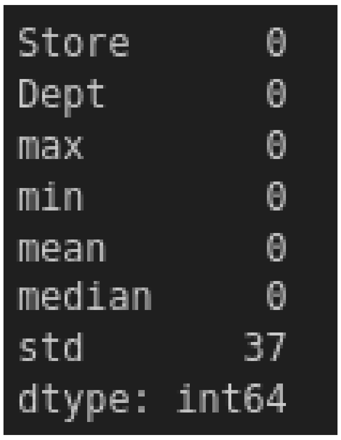


Year	Month	Week
2010	2	5

Figure 3.6: Datetime transformation

3.1.4 Outlier and abnormalities detection

We calculate various statistics for the 'Weekly_Sales' column within each group, including maximum, minimum, mean, median, and standard deviation based on unique combinations of 'Store' and 'Dept'. The result shows that there are 37 missing values in the 'std' column, meaning that for 37 unique combinations of 'Store' and 'Dept', there wasn't enough data to calculate the standard deviation, resulting in NaN values. We decides to drop rows with NaN values. This ensures that your subsequent analysis is based on a dataset with complete information, minimizing the impact of missing values on your results. This step removes 37 rows which are displayed below.



```
Store       0
Dept        0
max         0
min         0
mean        0
median      0
std        37
dtype: int64
```

Figure 3.7: Detecting outliers

Z-scores, also known as standard scores or z-values, are a measure of how far away a particular data point is from the mean of a group of data points, expressed in terms of standard deviations. The formula for calculating the z-score of a data point x in a distribution with mean μ and standard deviation σ is given by: $(x - \mu)/\sigma$

We sum up the individual 'MarkDown' columns to the 'Total_MarkDown' and then drop the individual 'MarkDown' columns. We can detect the outliers in 'MarkDown' by using a z-score with a threshold of 2.5. The rows that have the z-score greater or equal to 2.5 are considered potential outliers and are removed from the dataset. After this step, the size of the dataset decreased from 421533 to 375438

Finally, we check the distribution of the "Negative Weekly Sales" if any abnormality appears. We found that there are about 1191 records that have negative weekly sales, which are, compared to the total data, too small to affect the prediction. Hence, we eliminated these records to make the training process easier. So, the final number of data is 374247.

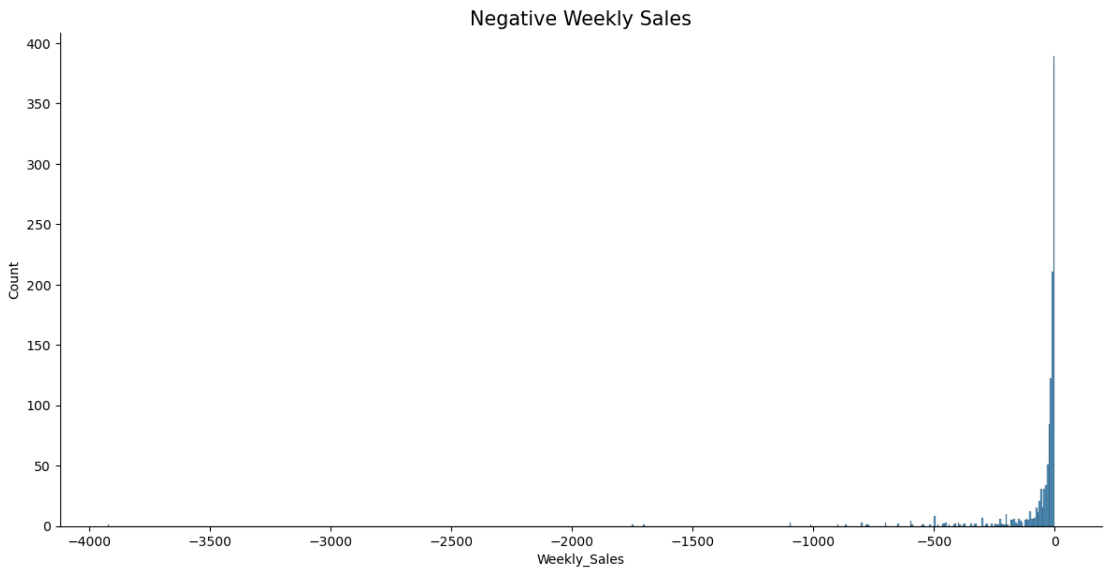


Figure 3.8: Negative weekly sales distribution

3.1.5 Data visualization

Sales appear to reach local peaks in months 6 and 12. However, the overall monthly sales do not exhibit significant fluctuations.

There are significant fluctuations in each month's sales per year, so we can't find a clear trend based on this.

There are huge differences between stores. The graph shows that the stores 3, 5, 33, and 44 have low average sales, while stores 2, 4, 6, 10, 20, and 28 have high average sales. The biggest difference is about 15000.

The graph shows the same trend for average sales per store. However, some departments, such as 39, 43, 45, 47, 51, 54, 60, 77, 78, 99, have nearly no sales. When evaluating, we can try some samples, including these attribute values. If the sales are nearly zero, the result seems very good.

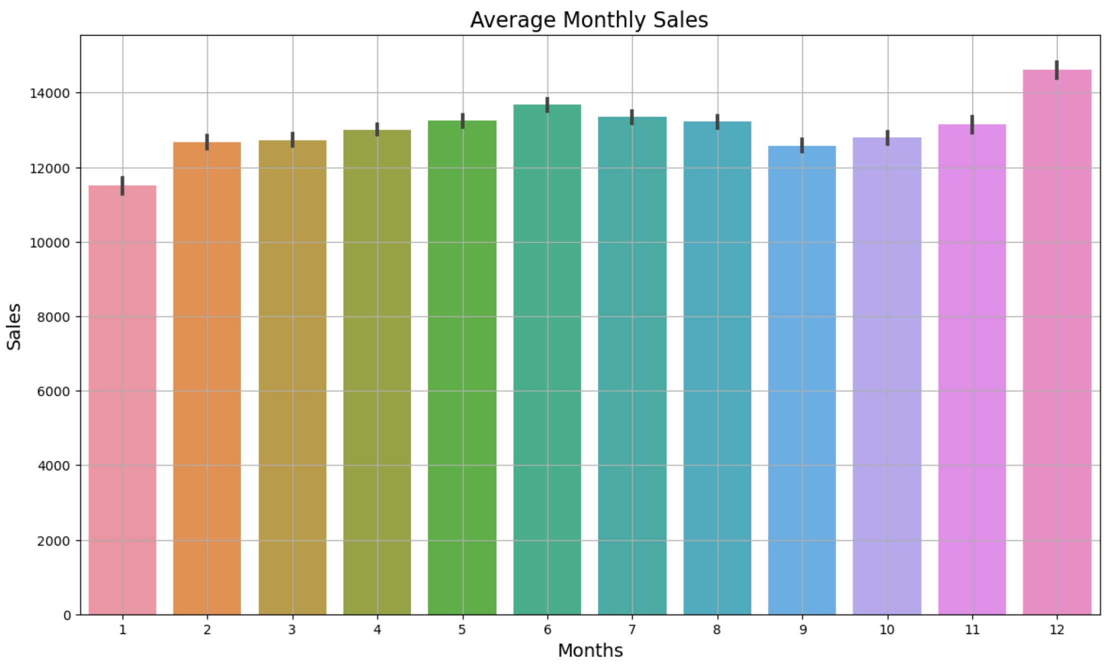


Figure 3.9: Avarage monthly sales

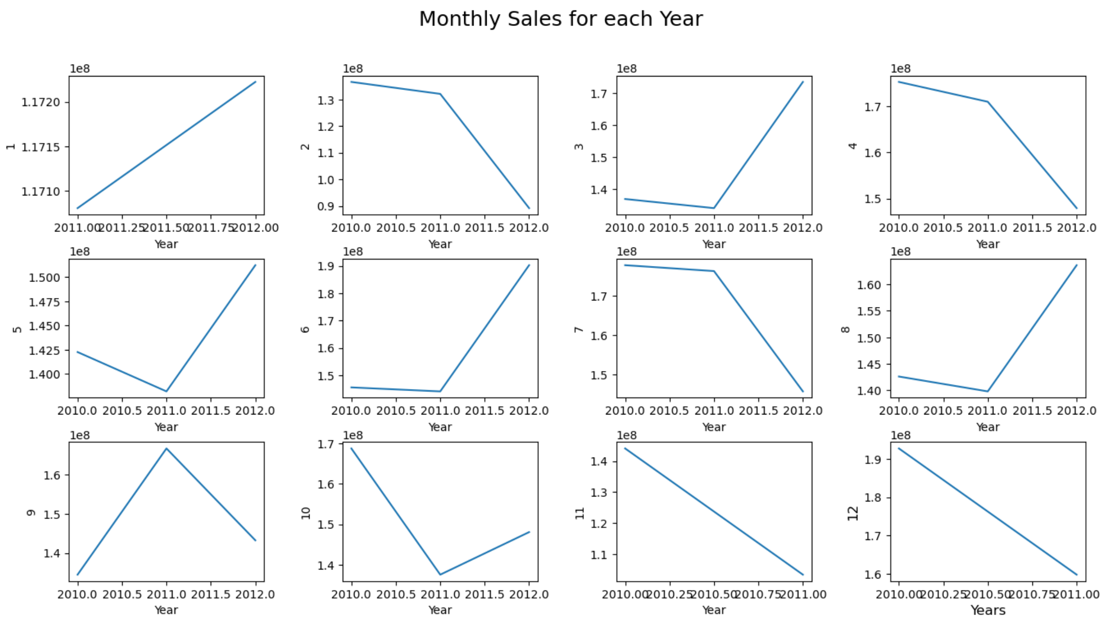


Figure 3.10: Monthly sales per year

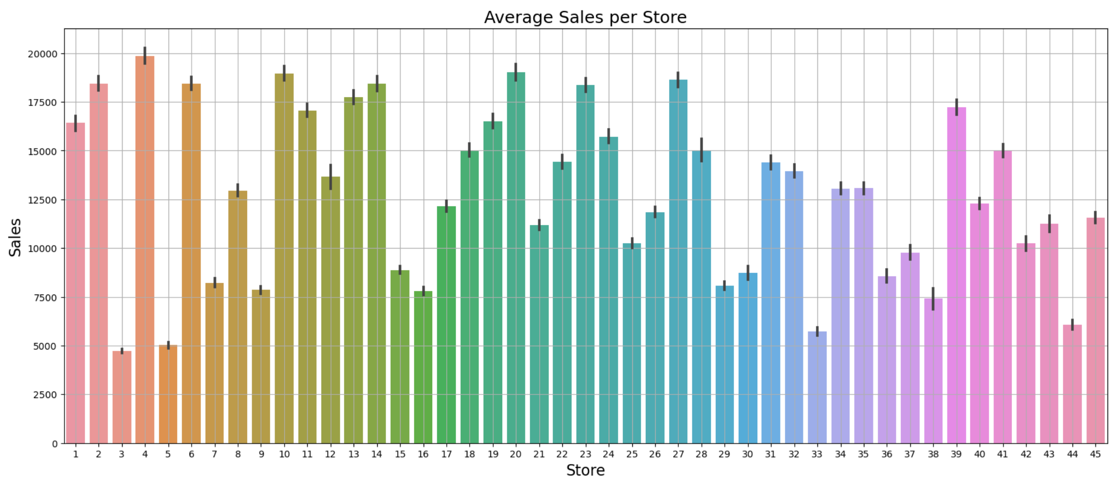


Figure 3.11: Avarage sales per store

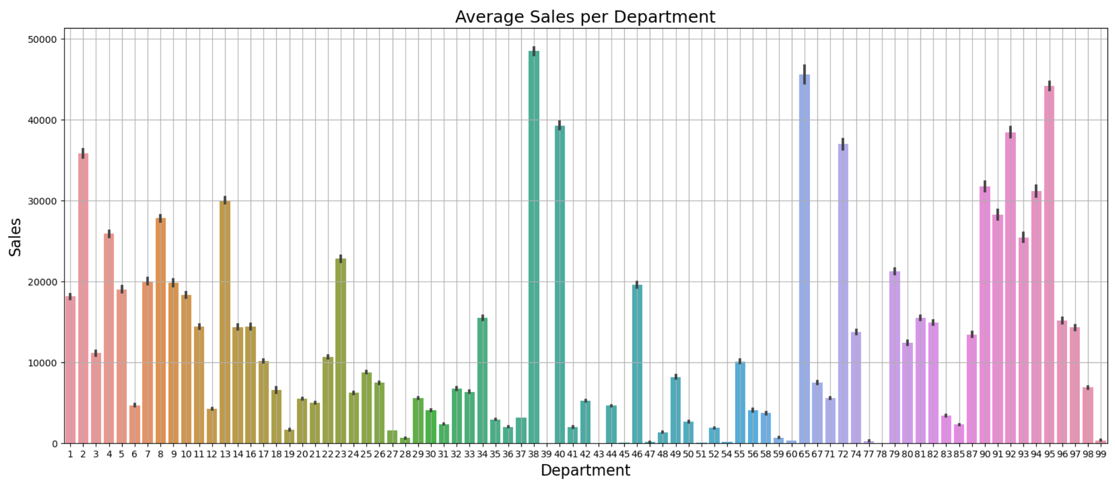


Figure 3.12: Average sales per department

3.1.6 Data normalization

Before training data, we must encode the Store, Dept, and Type into one-hot data. The Date column is redundant; therefore, we also discard this feature. After doing this, the shape of the data becomes (374247, 145). To avoid potential bias of some features because of the large number of features, we will normalize the data into (0,1).

We can use the Random Forest Regressor model to assess the feature importance

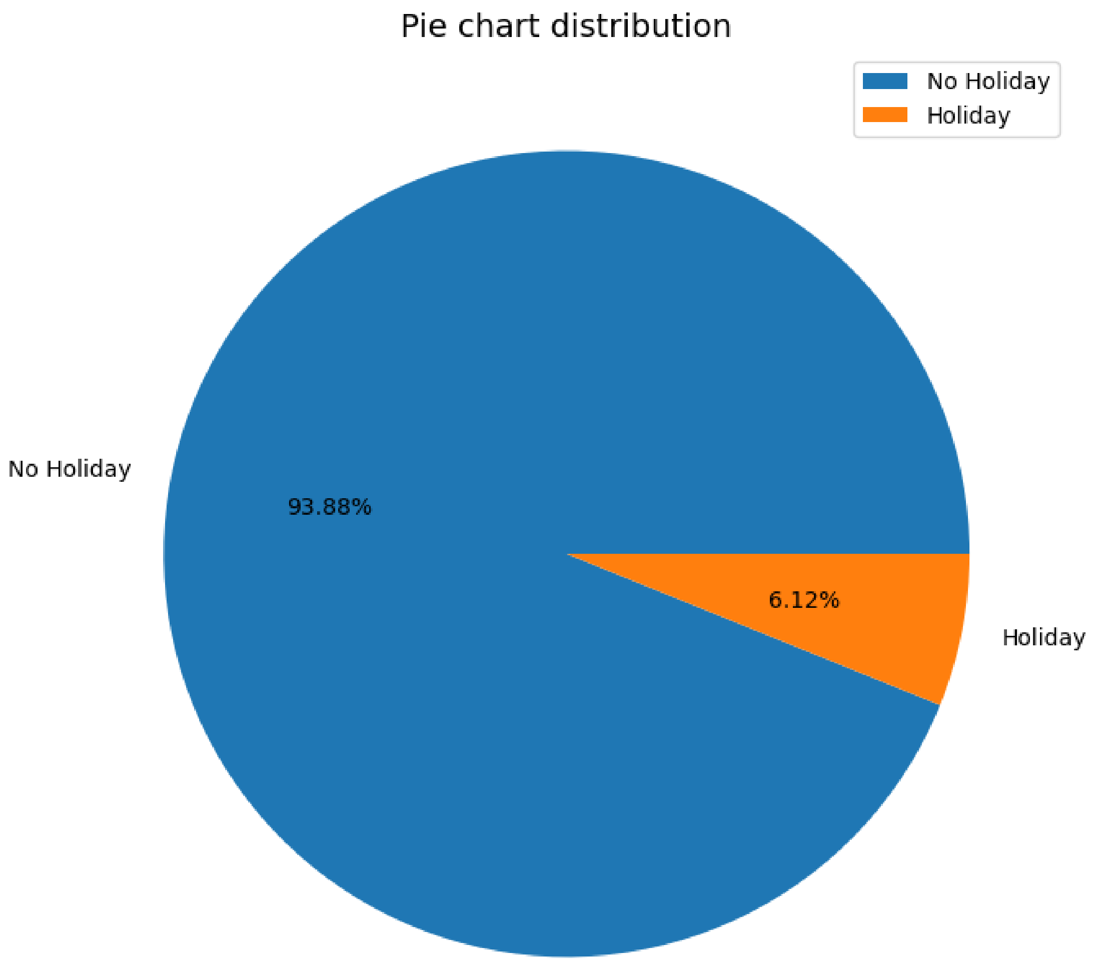
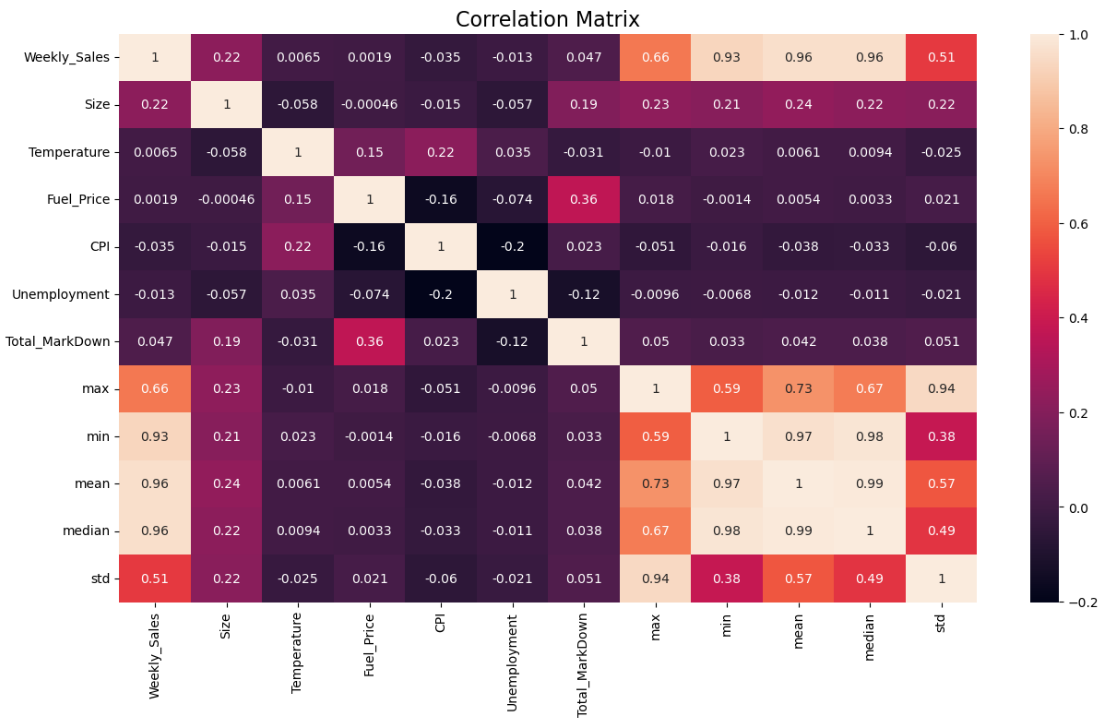


Figure 3.13: Holiday distribution

from the correlation between the dataset's features. After that, we use the top 23 features, which affects to our prediction: 'mean', 'median', 'Week', 'Temperature', 'max', 'CPI', 'Fuel_Price', 'min', 'Unemployment', 'std', 'Month', 'Total_MarkDown', 'Dept_16', 'Dept_18', 'IsHoliday', 'Dept_3', 'Size', 'Dept_9', 'Dept_11', 'Dept_1', 'Year', 'Dept_5', 'Dept_56'.

Finally, we use two splitting settings for the dataset. In the first setting, the dataset is split into 64:16:20 for training, validating, and testing. The second setting splits the dataset into 40:10:50 for training, validating, and testing.



	Weekly_Sales	Size	Temperature	Fuel_Price	CPI	Unemployment	Total_MarkDown	max	min	mean	median	std
Weekly_Sales	1	0.22	0.0065	0.0019	-0.035	-0.013	0.047	0.66	0.93	0.96	0.96	0.51
Size	0.22	1	-0.058	-0.00046	-0.015	-0.057	0.19	0.23	0.21	0.24	0.22	0.22
Temperature	0.0065	-0.058	1	0.15	0.22	0.035	-0.031	-0.01	0.023	0.0061	0.0094	-0.025
Fuel_Price	0.0019	-0.00046	0.15	1	-0.16	-0.074	0.36	0.018	-0.0014	0.0054	0.0033	0.021
CPI	-0.035	-0.015	0.22	-0.16	1	-0.2	0.023	-0.051	-0.016	-0.038	-0.033	-0.06
Unemployment	-0.013	-0.057	0.035	-0.074	-0.2	1	-0.12	-0.0096	-0.0068	-0.012	-0.011	-0.021
Total_MarkDown	0.047	0.19	-0.031	0.36	0.023	-0.12	1	0.05	0.033	0.042	0.038	0.051
max	0.66	0.23	-0.01	0.018	-0.051	-0.0096	0.05	1	0.59	0.73	0.67	0.94
min	0.93	0.21	0.023	-0.0014	-0.016	-0.0068	0.033	0.59	1	0.97	0.98	0.38
mean	0.96	0.24	0.0061	0.0054	-0.038	-0.012	0.042	0.73	0.97	1	0.99	0.57
median	0.96	0.22	0.0094	0.0033	-0.033	-0.011	0.038	0.67	0.98	0.99	1	0.49
std	0.51	0.22	-0.025	0.021	-0.06	-0.021	0.051	0.94	0.38	0.57	0.49	1

Figure 3.14: Feature correlation

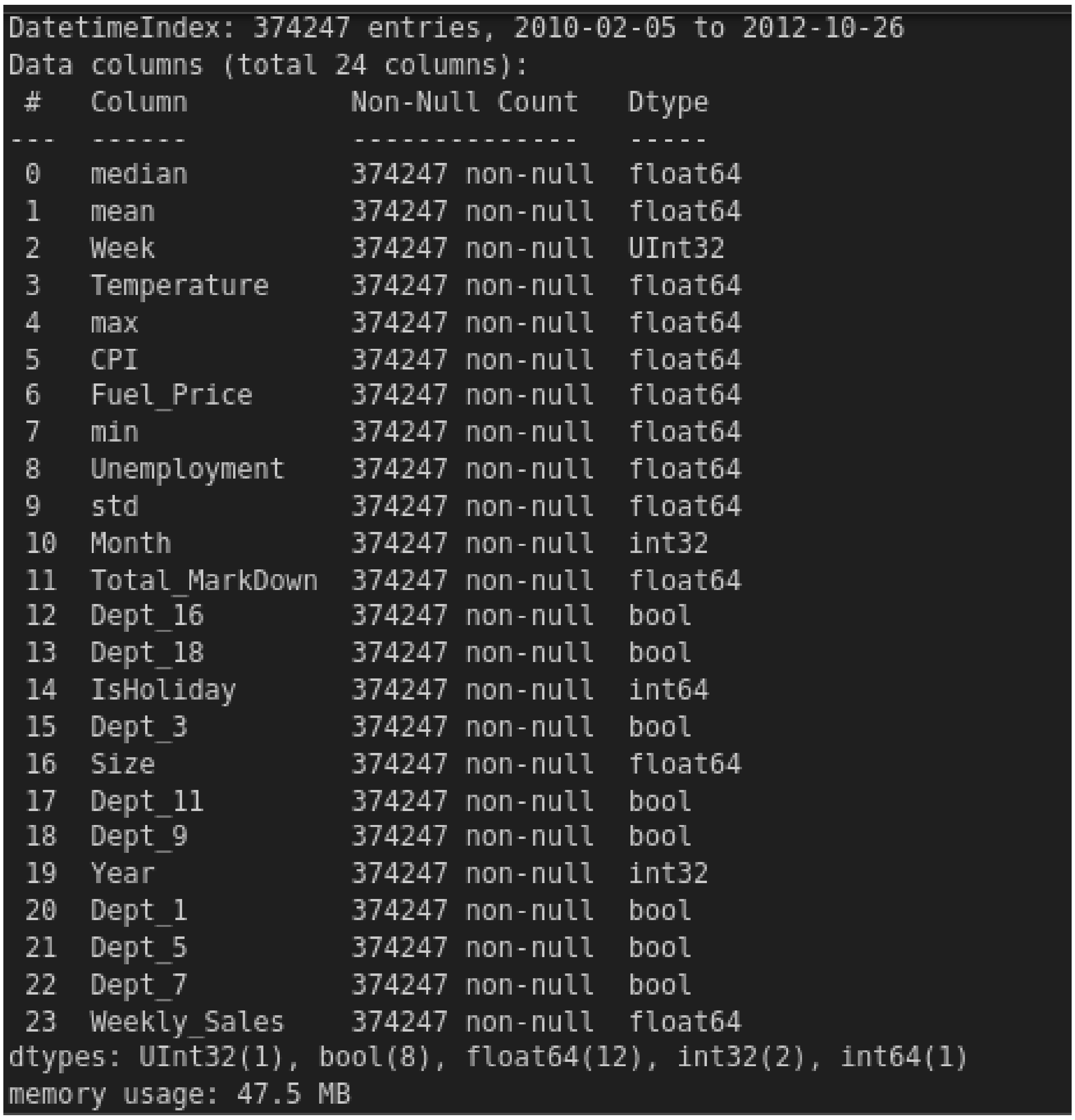
3.2 Metrics

1. **Mean Absolute Error (MAE):** MAE measures the average absolute differences between predicted and actual values. Smaller MAE values indicate better accuracy, and it is less sensitive to outliers compared to other metrics.

2. **Mean Squared Error (MSE):** MSE measures the average squared differences between predicted and actual values. Lower MSE values indicate better accuracy. Squaring emphasizes larger errors, making it sensitive to outliers.

3. **Root Mean Squared Error (RMSE):** RMSE is the square root of the MSE and provides a measure of the average magnitude of errors. Smaller RMSE values indicate better accuracy. Like MSE, it is sensitive to outliers.

4. **R-squared (R^2 or R-squared):** R-squared represents the proportion of the variance in the dependent variable that is predictable from the independent variables. R-squared measures the goodness of fit. A value of 1 indicates a perfect fit, while 0 indicates that the model does not explain any variability.



```
DatetimeIndex: 374247 entries, 2010-02-05 to 2012-10-26
Data columns (total 24 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   median          374247 non-null  float64
 1   mean            374247 non-null  float64
 2   Week            374247 non-null  UInt32
 3   Temperature     374247 non-null  float64
 4   max             374247 non-null  float64
 5   CPI             374247 non-null  float64
 6   Fuel_Price      374247 non-null  float64
 7   min             374247 non-null  float64
 8   Unemployment    374247 non-null  float64
 9   std             374247 non-null  float64
 10  Month           374247 non-null  int32
 11  Total_MarkDown  374247 non-null  float64
 12  Dept_16         374247 non-null  bool
 13  Dept_18         374247 non-null  bool
 14  IsHoliday       374247 non-null  int64
 15  Dept_3          374247 non-null  bool
 16  Size            374247 non-null  float64
 17  Dept_11         374247 non-null  bool
 18  Dept_9          374247 non-null  bool
 19  Year            374247 non-null  int32
 20  Dept_1          374247 non-null  bool
 21  Dept_5          374247 non-null  bool
 22  Dept_7          374247 non-null  bool
 23  Weekly_Sales    374247 non-null  float64
dtypes: UInt32(1), bool(8), float64(12), int32(2), int64(1)
memory usage: 47.5 MB
```

Figure 3.15: Data after preprocessing.

3.3 Evaluation

Experiment Setup

All machine learning and deep learning models were trained using predefined data setting (details provided in Section 3.1.6). The Adam optimizer was utilized to train all deep neural networks with a learning rate of 0.001, betas of 0.9 and 0.999, and a batch size of 1000. The objective function for optimization was the mean squared error, with networks trained for 100 epochs on each time series dataset.

Specifically, the 3-layer multilayer perceptron (MLP) model consists of an input layer, followed by two hidden layers with 64 and 32 units, respectively, and ending in a single node output layer for value prediction. The Xavier normal initializer and ReLU activations were used for this network. The larger 6-layer MLP expanded on this architecture using dimensions of [64, 128, 256, 64, 32, 1] units progressing from input to output.

Standardized testing procedures were applied, with all predictive models evaluated on isolated test sets not used in model fitting. Performance across essential time series metrics, including MAE, MSE, RMSE, R^2, and inference time, was recorded for rigorous modeling approach comparisons. Tables 3.1 and 3.2 summarize both in-sample and out-of-sample model performance on the time series data using these evaluation criteria.

The systematic model tuning, training paradigm, and evaluation analytics provide a fair and controlled experimentation environment to measure each model's capability for forecasting applied time series problems.

Results

The empirical results demonstrate the superiority of tree-based machine learning models, including random forest, XGBoost, LightGBM, and CatBoost, for time series forecasting using tabular data. As observed in Table 3.1, tree ensembles significantly outperformed all other classical regression techniques on the first dataset while achieving near equivalent predictive accuracy to deep learning, but with orders of magnitude (100x) faster training time. More tellingly, the tree models excelled even further on the second, more challenging data setting, as shown in Table 3.2, indicating greater generalization capability to varied time series problems.

Several innate characteristics underpin the exceptional performance of tree-based methods for modeling temporal data:

- First, the tree construction process automatically conducts hierarchical feature selection and identifies interactions between variables. This allows accurate capturing of the multilayered relationships and complex patterns embedded within time series.

- Second, ensemble approaches improve robustness and stability by aggregating predictions across diverse decision tree base models to reduce overfitting and variance. This bagging and boosting of many estimators is well-suited to the noise and uncertainty of forecasting tasks.
- Lastly, boosting algorithms like XGBoost and LightGBM in particular, chain models sequentially, with each new estimator focusing on correcting the errors from the previous model. This refinement of predictive signals over training iterations mirrors the goal of prediction into future points in time series data.

Table 3.1: Models Evaluation Metrics on the test set of the first setting (64:16:20). The **red**, **green**, and **blue** numbers denote the top 1, 2, and 3 performance among all methods. The **brown** number denotes the best performance among the DL-based methods.

Model	**MAE**	**MSE**	**RMSE**	**R^2**
ML-based Methods				
Linear Regressor	0.0301	0.0035	0.0590	0.9228
KNN Regressor	0.0331	0.0036	0.0602	0.9199
Random Forest Regressor	0.0155	0.0010	0.0309	**0.9788**
XGBoost Regressor	0.0198	0.0012	0.0349	**0.9730**
LightGBM	0.0223	0.0016	0.0401	0.9643
CatBoost	0.0169	0.0009	0.0299	**0.9802**
DL-based Methods				
MLP (3 layers)	0.0283	0.0038	0.0619	0.9183
MLP (6 layers)	0.0303	0.0025	0.0509	0.9503
LSTM	0.0248	0.0021	0.0454	**0.9504**
GRU	0.0268	0.0022	0.0474	0.9503

Intuitively, the gated recurrent architecture of long short-term memory (LSTM) networks and gated recurrent units (GRU) should confer significant advantages for sequence modeling tasks compared to generic feedforward multilayer perceptrons (MLP). However, the empirical results challenge this assumption, demonstrating extremely competitive forecasting accuracy from the conceptually more straightforward but appropriately configured MLP models, nearly on par with specialized recurrent networks.

Precisely, while the 6-layer MLP trailed GRU performance by a minor margin of 0.0001 regarding R-squared, it matched or exceeded the predictive capability of both LSTM and GRU across the sales forecasting metrics. This surprising finding highlights the underrated capabilities of MLP networks for capturing complex temporal relationships given adequate depth and nonlinearity. It also reflects the sheer representational capacity enabled by overparameterization and depth, even without intricate temporal gates or memory cells.

Moreover, adequately tuned MLP models offer inference efficiency advantages over recurrent networks, requiring fewer floating point operations during prediction. This demonstrates a skillful balance between trainability, accuracy, and deployment cost for production forecasting systems.

Table 3.2: Models Evaluation Metrics on the test set of the second setting (40:10:50). The **red**, **green**, and **blue** numbers denote the top 1, 2, and 3 performance among all methods. The **brown** number denotes the best performance among the DL-based methods.

Model	**Accuracy**	**MAE**	**MSE**	**RMSE**	**R^2**
ML-based Methods					
Linear Regressor	0.0301	0.0035	0.0590	0.9225	
KNN Regressor	0.0368	0.0045	0.0670	0.9007	
Random Forest Regressor	0.0170	0.0012	0.0340	**0.9744**	
XGBoost Regressor	0.0202	0.0013	0.0358	**0.9716**	
LightGBM	0.0226	0.0017	0.0410	**0.9626**	
CatBoost	0.0183	0.0010	0.0323	0.9768	
DL-based Methods					
MLP (3 layers)	0.0290	0.0038	0.0615	0.9199	
MLP (6 layers)	0.0490	0.0052	0.0719	0.9221	
LSTM	0.0296	0.0026	0.0511	0.9287	
GRU	0.0288	0.0032	0.0568	**0.9288**	

In summary, while innovations like LSTMs and GRUs have grabbed research attention, practitioners should not overlook or underestimate the formidable time series modeling aptitude exhibited by multilayer perceptron deep neural networks, even surpassing specialized architectures. This highlights the need for controlled benchmarking to reveal the best production solution rather than relying on potentially misguided architectural assumptions around capability.

4 Conslusion

This research presented a comprehensive benchmark between eight prominent machine learning and deep learning algorithms on real-world time series prediction tasks. Through an extensive comparative analysis over a large store sale dataset, the performance of linear regression, KNN, random forest, XGBoost, LightGBM, CatBoost, MLPs, LSTMs, and GRUs was rigorously assessed across accuracy, computational efficiency, and generalization capability dimensions.

The empirical results definitively demonstrated the effectiveness of tree-based ensemble techniques, with LightGBM emerging as the top performer by a slight margin over companions XGBoost, CatBoost, and random forest. The automatic feature engineering intrinsically conducted via hierarchical data splits and ensemble aggregation of hundreds of decision trees aptly handles noise, captures intricate data relationships, and smoothens signals over time. Meanwhile, amongst neural architectures, multilayer perceptrons matched LSTMs and outperformed GRUs with adequate depth and nonlinearity, challenging notions that specialized recurrent models are necessitated for skillful sequence modeling.

Based on the comparative analysis, tree ensembles, and deep MLPs emerge as highly accurate and scalable options for time series forecasting amongst both classical and modern techniques, distinctly outpacing linear regression, KNN, simpler MLP, LSTM, and GRU. The project provides multiple prescribed guidelines around model selection, hyperparameter configuration, and architectural decisions tailored to time series for practitioners targeting deployments across industry verticals. Additionally, high-level conclusions were derived regarding the complementary representational strengths unique to tree-based and deep densely connected networks that transfer competency for temporal modeling problems.

By benchmarking an extensive set of leading time series analysis techniques on real-world data, this project achieved its primary objective of evaluating approach viability while constructing both tailored and general prescriptive guidance to inform applied machine learning solutions for time series tasks. The insights offer value to both researchers and industry leaders navigating this pivotal subdomain of predictive analytics.

References

Almeida, A., Brás, S., Sargento, S. and Pinto, F. C. [2023], 'Time series big data: a survey on data stream frameworks, analysis and algorithms', *Journal of Big Data* **10**(1), 83.

Bengio, Y., Simard, P. and Frasconi, P. [1994], 'Learning long-term dependencies with gradient descent is difficult', *IEEE transactions on neural networks* **5**(2), 157–166.

Bontempi, G., Ben Taieb, S. and Le Borgne, Y.-A. [2013], 'Machine learning strategies for time series forecasting', *Business Intelligence: Second European Summer School, eBISS 2012, Brussels, Belgium, July 15-21, 2012, Tutorial Lectures 2* pp. 62–77.

Breiman, L. [2001], 'Random forests', *Machine learning* **45**, 5–32.

Brownlee, J. [2018], *Deep learning for time series forecasting: predict the future with MLPs, CNNs and LSTMs in Python*, Machine Learning Mastery.

Cao, L.-J. and Tay, F. E. H. [2003], 'Support vector machine with adaptive parameters in financial time series forecasting', *IEEE Transactions on neural networks* **14**(6), 1506–1518.

Chen, T. and Guestrin, C. [2016], Xgboost: A scalable tree boosting system, *in* 'Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining', pp. 785–794.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. [2014], 'Learning phrase representations using rnn encoder-decoder for statistical machine translation', *arXiv preprint arXiv:1406.1078* .

Chung, J., Gulcehre, C., Cho, K. and Bengio, Y. [2014], 'Empirical evaluation of gated recurrent neural networks on sequence modeling', *arXiv preprint arXiv:1412.3555* .

El, H., Mohamed, A. and Fawzy, H. [2021], 'Time series forecasting using tree based methods', *Journal of Statistics Applications & Probability* **10**, 229.

Guo, K., Hu, Y., Qian, Z., Liu, H., Zhang, K., Sun, Y., Gao, J. and Yin, B. [2020], 'Optimized graph convolution recurrent neural network for traffic prediction', *IEEE Transactions on Intelligent Transportation Systems* **22**(2), 1138–1149.

Hochreiter, S. [1998], ‘The vanishing gradient problem during learning recurrent neural nets and problem solutions’, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **6**(02), 107–116.

Hochreiter, S. and Schmidhuber, J. [1997], ‘Long short-term memory’, *Neural computation* **9**(8), 1735–1780.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. and Liu, T.-Y. [2017], ‘Lightgbm: A highly efficient gradient boosting decision tree’, *Advances in neural information processing systems* **30**.

Lim, B., Arık, S. Ö., Loeff, N. and Pfister, T. [2021], ‘Temporal fusion transformers for interpretable multi-horizon time series forecasting’, *International Journal of Forecasting* **37**(4), 1748–1764.

Müller, K.-R., Smola, A. J., Rätsch, G., Schölkopf, B., Kohlmorgen, J. and Vapnik, V. [1997], Predicting time series with support vector machines, *in* ‘International conference on artificial neural networks’, Springer, pp. 999–1004.

Park, D. C., El-Sharkawi, M., Marks, R., Atlas, L. and Damborg, M. [1991], ‘Electric load forecasting using an artificial neural network’, *IEEE transactions on Power Systems* **6**(2), 442–449.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V. and Gulin, A. [2018], ‘Catboost: unbiased boosting with categorical features’, *Advances in neural information processing systems* **31**.

Rumelhart, D. E., Hinton, G. E., Williams, R. J. et al. [1985], ‘Learning internal representations by error propagation’.

Saad, E. W., Prokhorov, D. V. and Wunsch, D. C. [1998], ‘Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks’, *IEEE Transactions on neural networks* **9**(6), 1456–1470.

Smith, B. L. and Demetsky, M. J. [1994], ‘Short-term traffic flow prediction: neural network approach’, *Transportation Research Record* (1453).

Taylor, S. J. and Letham, B. [2018], ‘Forecasting at scale’, *The American Statistician* **72**(1), 37–45.

Walmart Competition Admin, W. C. [2014], ‘Walmart recruiting - store sales forecasting’.
URL: *https://kaggle.com/competitions/walmart-recruiting-store-sales-forecasting*

Zhu, B., Ye, S., Wang, P., Chevallier, J. and Wei, Y.-M. [2022], 'Forecasting carbon price using a multi-objective least squares support vector machine with mixture kernels', *Journal of Forecasting* **41**(1), 100–117.