Forecasting Analysis
Report File
Group 4

Submitted by:

Kanak Rana AU759379,
Purbak Sengupta AU759326

10 May 2024

Contents

1 Introduction

This project delves into the pivotal domain of time series forecasting, situated within the framework of Explainable Statistical Learning. Our primary objective is to tackle the formidable challenge of predicting future outcomes by devising forecasting algorithms that prioritize accuracy and simplicity while ensuring interpretability. By bridging the chasm between intricate statistical methodologies and pragmatic decision-making processes, we aim to furnish end-users with easily understandable insights.

To achieve our objectives, we aim to explore any of these forecasting techniques, ranging from traditional models like ARMA, ARIMA, SARIMA, and SARIMAX to more sophisticated approaches such as LSTM, Prophet and XGBoost.

Through this endeavor, our aim is two-fold: to advance our comprehension of time series forecasting and to empower stakeholders across diverse domains with robust tools for informed decision-making. By elucidating the underlying mechanisms driving our predictions, we aspire to instill trust and confidence in the forecasting process, thereby facilitating positive outcomes.

In essence, this student-led initiative represents a comprehensive expedition into time series forecasting, underscored by the principles of accuracy, simplicity, and explainability. Through hands-on experimentation and analysis, we endeavor to unearth actionable insights that will underpin more informed decision-making in an ever-evolving and complex world.

2 Methods and Materials

In this project, we adopted a systematic approach to develop and evaluate forecasting algorithms for time series data, focusing on predicting diverse phenomena such as stock prices, weather patterns, and disease progression. Our methodology emphasized the integration of advanced machine learning techniques with principles of interpretability and explainability to provide actionable insights into future trends. The following outlines the key components of our methodology:

1. **Data Preparation and Exploration:** We commenced by collecting time series datasets from reputable source(Kaggle). Extensive data preprocessing ensured data integrity and consistency, involving tasks such as handling missing values and standardizing formats. Exploratory Data Analysis (EDA) facilitated insights into data structures, distributions, and potential relationships, guiding subsequent modeling decisions.

2. **Forecasting Algorithms:** Two of forecasting algorithms was curated, including:

    - Seasonal ARIMA (SARIMA)
    - XGBoost (eXtreme Gradient Boosting)

3. **Model Evaluation and Interpretability:** Comprehensive model evaluation was conducted using objective metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). Comparative analyses against benchmarks and advanced techniques shed light on algorithmic strengths and limitations. Interpretive visualizations, such as time series plots and feature importance analyses, aid in intuitive understanding and communication of forecasting results.

4. **Performance Assessment and Improvement:** Rigorous performance assessments were conducted on both training and testing datasets to validate model effectiveness. Iterative refinement and optimization processes aimed to enhance model performance and address identified shortcomings. Emphasis was placed on transparency and documentation throughout the experimentation process.

3 Experiment Setup

3.1 Installation of Required Libraries

We utilized the Python programming language along with several libraries including:
pandas, numpy, matplotlib.pyplot, seaborn, pyspark.sql.functions, pyspark.sql.SparkSession, xgboost and statsmodels.[1]

3.2 Data Acquisition

- The experiment utilizes time series data from Walmart's weekly sales, collected from Kaggle. The dataset includes various features such as store type, whether it's a holiday, temperature, fuel price, and CPI, among others. The target variable is the 'Weekly_Sales'.

```
In [11]: df = spark.read.options(delimeter=',', inferSchema=True, header=True).csv('/users/purbaksengupta/downloads/train.csv
         df.limit(3).toPandas().style.hide(axis="index")

In [14]: stores = spark.read.options(delimeter=',', inferSchema=True, header=True).csv('/users/purbaksengupta/downloads/store
         stores.limit(3).toPandas().style.hide(axis="index")

In [16]: features = spark.read.options(delimeter=',', inferSchema=True, header=True).csv('/users/purbaksengupta/downloads/fea
         features.limit(3).toPandas().style.hide(axis="index")
```

Figure 1: Loading Datasets into DataFrame

3.3 Exploratory Data Analysis (EDA)

The subsequent step involves exploratory data analysis (EDA) to gain insights into the dataset's characteristics, such as trends, seasonality, and correlations among variables.

```
In [22]: plt.figure(figsize=(10,5))
         sns.lineplot(data=pdf,
                      x='Date',
                      y='Weekly_Sales',
                     ).set_title('Walmart Weekly Sales from Feb 2010 to Oct 2012')
         plt.axhline(pdf['Weekly_Sales'].mean(), color='r', alpha=0.2, linestyle='--')
         plt.show()
```

Figure 2: Sales data trends over time

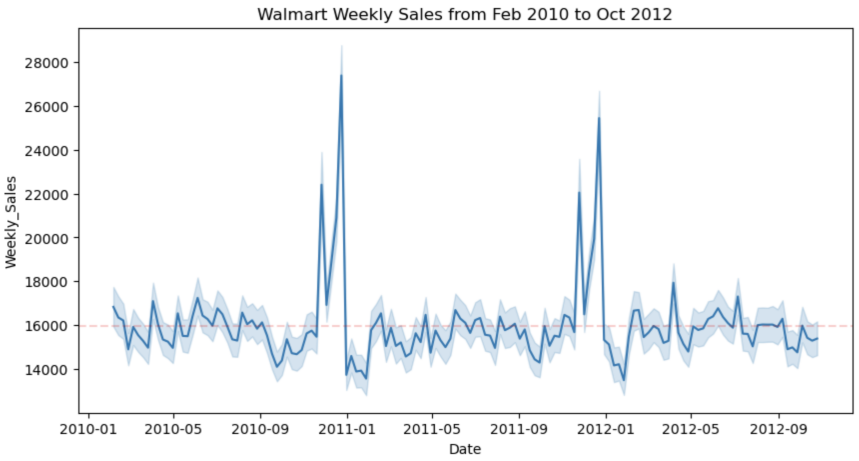


Figure 3: Visualization of Sales data trends over time

Other EDA visualizations were also created including sales by month, quarter, year, and store type.

3.4 Preprocessing

The experiment then involves preprocessing steps to prepare the data for analysis and modeling. This includes handling missing values, creating new features, converting categorical variables, and removing outliers.

```
In [18]: for i in range(1,6):
             features = features.withColumn(
                 f'MarkDown{i}', f.when(f.col(f'MarkDown{i}')=='NA', None).otherwise(
                     f.col(f'MarkDown{i}')).cast('float'))
```

Figure 4: Preprocessing Steps

```
In [19]: pdf = df.join(
             stores, on='Store', how='inner').join(
             features.drop('IsHoliday'), on=['Date','Store'], how='left').toPandas()
         pdf['Date'] = pd.to_datetime(pdf['Date'])
         pdf.set_index('Date', inplace=True)
         pdf["CPI"] = pd.to_numeric(pdf["CPI"], downcast="float")
         pdf["Unemployment"] = pd.to_numeric(pdf["Unemployment"], downcast="float")
         pdf.head()
```

Figure 5: Merging datasets and transforming data types

```
pdf['Date'] = pd.to_datetime(pdf['Date'])
pdf.set_index('Date', inplace=True)
pdf["CPI"] = pd.to_numeric(pdf["CPI"], downcast="float")
pdf["Unemployment"] = pd.to_numeric(pdf["Unemployment"], downcast="float")
pdf['Day'] = pdf.index.day
pdf['Week'] = np.ceil(pdf.index.dayofyear/7).astype(int)
pdf['Month'] = pdf.index.month
pdf['Quarter'] = pdf.index.quarter
pdf['Year'] = pdf.index.year

pdf = pdf.query('Weekly_Sales >= 0 and Weekly_Sales < 200000')
pdf.head().style.hide(axis="index")
```

Figure 6: Date manipulation and feature engineering

```
In [45]: old_count = pdf.shape[0]
         print('Dataset_size before removing outliers: ', old_count)
         pdf = pdf.query('Weekly_Sales >= 0 and Weekly_Sales < 200000')
         new_count = pdf.shape[0]
         print('Dataset_size after removing outliers: ', pdf.shape[0])
         print('Percent decrease in data: ', round((old_count - new_count)*100/old_count,2))
```

Figure 7: Outlier Removal

3.5 Model

Following these steps, the experiment progresses to modeling and evaluation using machine learning techniques. In this case, XGBoost and SARIMA regressions are employed for time series forecasting, and performance metrics are computed to assess the model's accuracy.

```
In [57]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(pdf[FEATURES], pdf[TARGET],
                                                             test_size=0.2, random_state=130)
```

```
In [58]: import xgboost as xgb
         reg = xgb.XGBRegressor(n_estimators=2000, early_stopping_rounds=50,
                                learning_rate=0.5)
         model = reg.fit(X_train, y_train,
                eval_set=[(X_train, y_train), (X_test, y_test)],
                verbose=100)
```

Figure 8: XGBoost regression model fitting and evaluation

```
In [60]: y_pred = reg.predict(X_test)
```

```
In [62]: from sklearn import metrics
xgboost_accuracy = reg.score(X_test,y_test)*100
print("XGB Regressor Evaluation Metrics: ")
print("Accuracy", round(xgboost_accuracy,3))
print("MAE \t" , round(metrics.mean_absolute_error(y_test, y_pred),3))
print("MSE \t" , round(metrics.mean_squared_error(y_test, y_pred),3))
print("RMSE \t" , round(np.sqrt(metrics.mean_squared_error(y_test, y_pred)),3))
print("R2 \t" , round(metrics.explained_variance_score(y_test, y_pred),5))
```

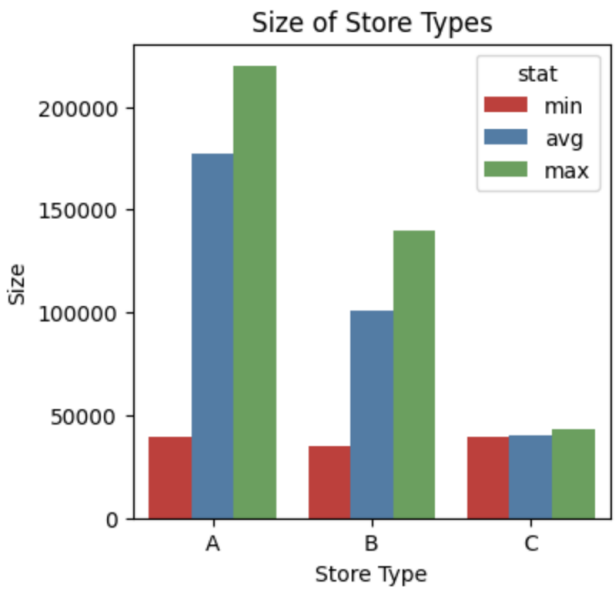
Figure 9: Evaluation Metrics

```
In [75]: # The model with the best p and q found from pervious step
import statsmodels.api as sm
model = sm.tsa.statespace.SARIMAX(train.Weekly_Sales, order=(52, 0, 5))
# Fit model
results = model.fit()
```

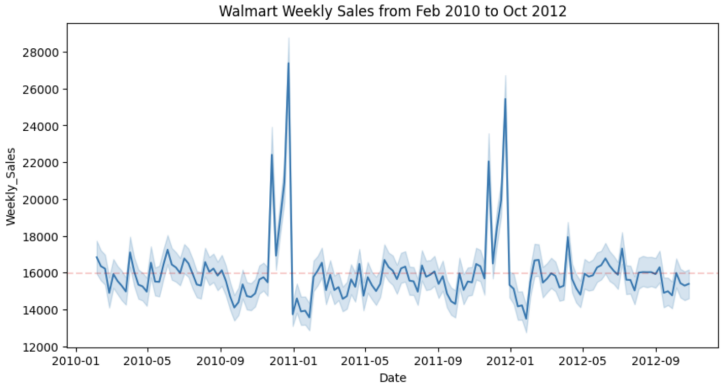
Figure 10: SARIMA Model Fitting

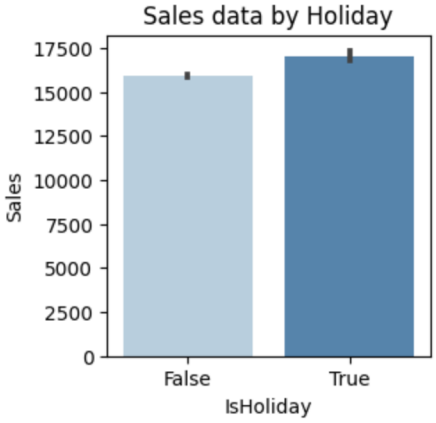
4 Results

- Type 'A' are the biggest store types with highest 'max' and 'average'
- Type 'C' are the smallest stores with lowest 'max' and 'average'

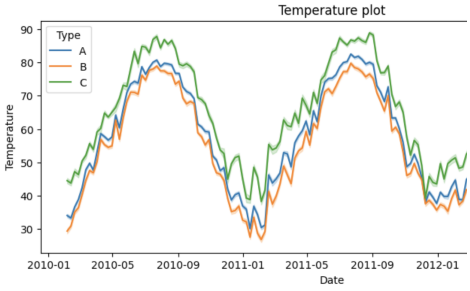


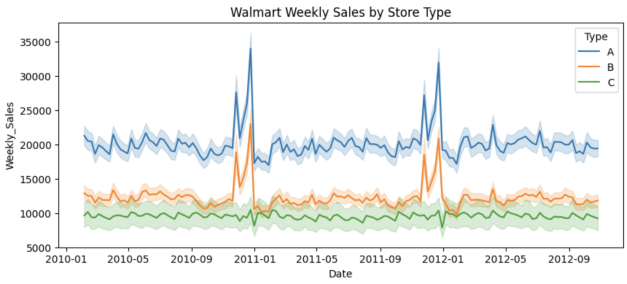
- We see a spike in Sales during the holiday season (November to December) every year
- Sales are slightly higher during Holidays





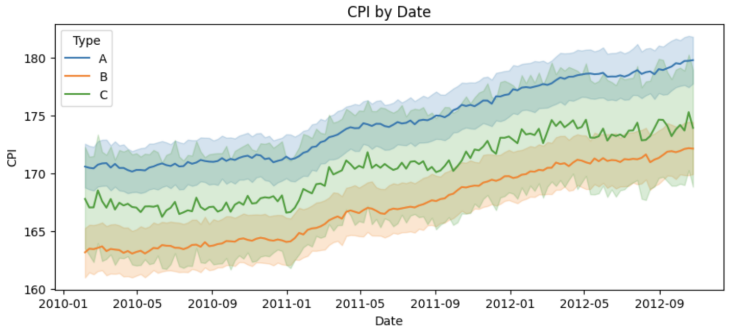
- Bigger stores (A) have higher sales compared to smaller stores (B), however Temperature around Type C stores (small stores) is high compared to other stores



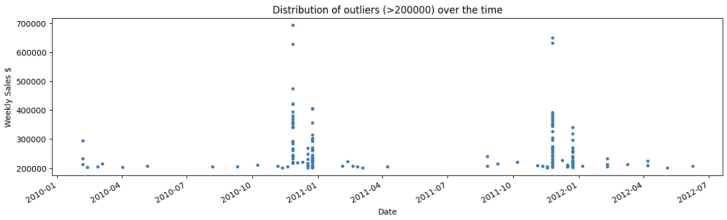


- Overall CPI (Consumer Price Index) is increase over time
- Type A stores have highest CPI index

- Type C stores have higher CPI index than Type B stores

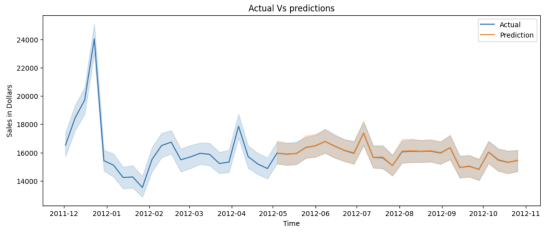


- Throughout the year, we have some positive outliers (sales more than 200000$)



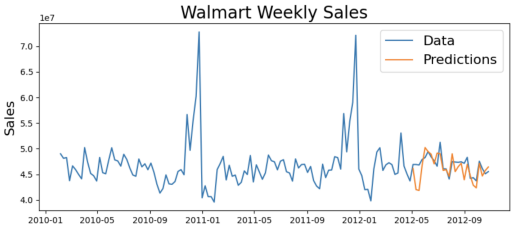
- The Predictions of XGBoost are pretty close to the actual values as we have a accuracy of 98%

```
XGB Regressor Evaluation Metrics:
Accuracy 98.348
MAE      1570.744
MSE      8134464.931
RMSE     2852.098
R2       0.98348
```



- SARIMA's prediction result:

```
MAPE:  0.0299
Mean Absolute Error:  1406495.757228183
Root Mean Squared Error:  1896605.36
```



5 Discussion, Conclusion and Perspectives

XGBoost outperformed SARIMA with an accuracy of around 98%, while SARIMA achieved a MAPE of about 2.99%. XGBoost demonstrated superior performance in MAE and RMSE, with values indicating closer predictions to actual sales. Despite attempts to optimize SARIMA parameters, it couldn't match XGBoost's performance.

XGBoost's superior performance over SARIMA can be attributed to its ability to capture complex nonlinear relationships, thanks to its powerful machine learning algorithms.[3] Its ensemble learning approach allows it to handle high-dimensional datasets effectively.[4] Additionally, XGBoost's robustness in handling missing values, outliers, and overfitting further contributes to its predictive accuracy.[3] SARIMA, on the other hand, relies on linear components that may not adequately capture the nonlinear patterns in the Walmart sales data.[2]

Overall, further investigation is necessary to delve into additional factors impacting model performance and to fine-tune parameters for specific forecasting tasks. To sum it up, this exploration was insightful and enjoyable, opening avenues for future research in the field.

References

[1] M. Peixeiro, "The Complete Guide to Time Series Forecasting Using Sklearn, Pandas, and Numpy", Towards Data Science, Tutorial, 2022

[2] R.J. Hyndman and G. Athanasopoulos, "Forecasting: Principles and Practice", 3rd ed., OTexts, Melbourne, Australia, 2021 .

[3] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. ArXiv

[4] Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." Annals of statistics (2001): 1189-1232.