EXPERIMENT REPORT - Forecasting

Student Name	Kritika Dhawale
Project Name	AT2 - Machine Learning as a Service
Date	7th October 2023
Deliverables	dhawale_kritika-24587661-arima.ipynb `*arima.joblib*`
Github Link	adv_mla_at2
Heroku API link	at2_api

---

1. EXPERIMENT BACKGROUND	
Provide information about the problem/project such as the scope, the overall objective, expectations. Lay down the goal of this experiment and what are the insights, answers you want to gain or level of performance you are expecting to reach.	
1.a. Business Objective	- This project aims to create a predictive model that forecasts upcoming sales revenue for all stores and items over 7 days. The proposed model should improve inventory management, resource allocation, and financial planning by providing business intelligence. - The outcomes of this project will be leveraged to optimise inventory management, enhance supply chain efficiency, and augment sales and revenue planning for the business. The use of precise forecasting models will result in reduced occurrences of stockouts, mitigated instances of overstock situations, enhanced levels of customer satisfaction, and amplified levels of profitability. Inaccurate outcomes may result in suboptimal inventory levels, thereby impacting customer satisfaction and potential revenue generation. - Hence, the precision of the model holds utmost significance in facilitating well-informed business determinations and attaining optimal operational efficacy.
1.b. Hypothesis	- “What are the expected sales for the next week? Do we need to adjust our inventory levels? How much revenue can we anticipate?” - Considering this hypothesis is essential because it addresses a critical business need and has the potential to deliver significant operational and financial benefits.

1.c. Experiment Objective	The experiment's goal is to deploy a reliable forecasting model that can predict the total sales revenue for all items and stores for the upcoming seven days with an acceptable difference in the revenue. This follows; attaining a Root Mean Squared Error (RMSE) that indicates a high level of forecasting accuracy. While the specific numerical goal may vary depending on the dataset and business context, the objective is to minimize forecasting errors and maximize the model's precision. 1. Successful model: The model achieves lower error values, with MSE within an acceptable range - forecasts closely matching actual sales; this leads to better inventory control and financial planning. 2. Moderate Accuracy: The model offers insightful information for decision-making despite its mediocre performance and occasional forecasting errors. In this case, iterative model refinement and feature engineering may be needed to enhance performance. 3. Poor Accuracy: There are large forecasting errors/RMSE is high. in the model's performance, which may call for additional model optimisation or the use of different strategies. 4. Overfitting or Underfitting: The model exhibits signs of overfitting or underfitting, indicating that the model complexity needs adjustment. In order to improve the business's operational efficiency and financial performance, the objective is to minimise scenario 3 and maximise scenarios 1 and 2.

---

2. EXPERIMENT DETAILS	
Elaborate on the approach taken for this experiment. List the different steps/techniques used and explain the rationale for choosing them.	
2.a. Data Preparation	**Data Cleaning & transformation:** - Removing unnecessary columns ["id", "dept_id", "cat_id", "state_id"] - Added item_id and store_id to test set - test set was an extension to train set - Pivot the data frame: transform days' columns in rows, the units sold each day in a separate column - in order to calculate revenue for each day - Repeated the same steps for the test set. - Removed outliers from train set - improve model performance by nornmalizing the mean. - There were 55% of zero sales revenue in train set - had to remove them as it resulted in bias. - Saved the interim cleaned data into a pickle file **Data Preprocessing (Included in model pipeline):** - Set the index of both sets as date - requisite for time series analysis - Normalised data using the Box-Cox transformation to move it closer to a normal distribution. **Future Steps:** - I performed batch processing to reduce the training time and memory usage while loading the model. With a batch size of 512, there were 3 batches. The results are discussed in the final report.

2.b. Feature engineering	- Merged date from calendar data - Merged item's weekly sell price from weekly price data. - Calculated daily total revenue (units sold x items weekly sell price for that particular week) - expected in the task outcome - Grouped the items and stores by date to calculate total revenue across all stores. - Repeated the same steps for the test set. **Potential ad-ons:** Can add the lag features, weekdays, etc as the exogenous features in the time series model. For example: from the calender event data, we can pull out the public holidays and events like NBA to get extra features that can potentially affect the sales revenue. Due to time constraint, it was difficult to merge these features, but something not to miss out on.
2.b. Modelling	For this experiment, three models (one baseline) were trained and evaluated: 1. Exponential Weighted Moving Average (EWMA): This model calculates the Exponential Weighted Moving Average on the training data with an alpha value of 0.1. The choice of alpha was made based on typical values used for smoothing time series data. This simple model serves as a **baseline**. 2. **Auto ARIMA:** An Auto ARIMA model was chosen for its ability to automatically determine the optimal ARIMA parameters. The hyperparameters tuned include: - "Stepwise" set to True - fits the model faster using a subset of hyperparameter combinations (tips). - "Seasonal" set to True as it considers seasonality in the data. - "m" is set to 12 as the data appears to have a yearly seasonality. - "n_jobs" set to -1 to maximize CPU usage for faster computation. 3. SARIMAX: A SARIMAX model was chosen to capture seasonality, and it was combined with seasonal decomposition. The hyperparameters were manually selected from the best fit ARIMA model as follows: - ARIMA order: (5, 1, 5) - Seasonal order: (2, 0, 2, 12) **Not Trained Models:** Complex machine learning models like Random Forest or Gradient Boosting. These models were not chosen due to the primary focus on time-series analysis, which often requires specialized models like SARIMA or ARIMA. Future experiments could explore the integration of Neural Netwroks for additional insights and model optimisation. **Future Tests:** Incorporating exogenous features: Including additional features such as holidays, promotions, or external factors for improved forecasting accuracy. As well as, carefully balance the range of hyperparameters to search and the available computational resources to achieve a reasonable trade-off between model performance and training time.

3. EXPERIMENT RESULTS

Analyse in detail the results achieved from this experiment from a technical and business perspective. Not only report performance metrics results but also any interpretation on model features, incorrect results, risks identified.

3.a. Technical Performance

The models were evaluated using Root Mean Square Error (RMSE) as the performance metric. Here are the RMSE scores for each model on both the training and test sets:

1. Exponential Weighted Moving Average (EWMA):
- Train Data RMSE: 14,174.62
- Test Data RMSE: 18,781.73

2. Auto ARIMA:
- Train Data RMSE: 56,688.42
- Test Data RMSE: 17,238.10

3. SARIMA with Seasonal Decomposition (SARIMAX):
- Train Data RMSE: 27,509.99
- Test Data RMSE: 21,173.81

Analysis:
ARIMA model has significantly higher training RMSE compared to test set. This could be one of the reasons: **Data Characteristics -** The training data may have some unique characteristics or outliers that make it challenging for the ARIMA model to fit accurately. These characteristics might not be as prevalent in the test data.

SARIMAX also provides good performance on the test data, although it has a higher RMSE compared to ARIMA. The RMSE on the training data is closer to the test data, indicating better generalization.

Overall, ARIMA currently stands out as the best-performing model, but it may require optimization to prevent underfitting. SARIMAX also performs well and could be fine-tuned for further improvement. Future experiments could explore ensembling techniques or incorporating additional exogenous features to enhance forecasting accuracy.

3.b. Business Impact

Inaccurate revenue forecasts can have cascading effects on inventory management, supply chain optimization, and sales strategy.
Underestimating revenue might result in stockouts, missed sales opportunities, and customer dissatisfaction.
Overestimating revenue could lead to overstocking, excess inventory costs, and potentially, discounting to clear excess stock.
While the ARIMA and SARIMAX models show improvements over the baseline EWMA model, there is still room for enhancement in forecasting accuracy. Continual monitoring, model refinement, and potentially exploring advanced forecasting methods are essential for achieving more precise revenue forecasts, minimizing the impact of incorrect results on business operations, and maximizing revenue potential.

3.c. Encountered Issues	1. Setup - Setting the correct versions in both model building and deployment in the requirements file. When I first deployed the model, there was issue with the OHE in deployed model. Resolved: Saw the docker logs, and found out its the version mis-match error. 2. Memory Error: There were a lot of instances where the system would freeze and give memory usage error. Resolved: Downcasted float64 to float32, categorical data to “pandas category type”. 3. Auto Arima takes 10 minutes to train which is quite long: changed parameter “stepwise”=True to enable faster training. **Update:** Training time did not reduce much: 9.9 min. 4. Heroku deployment issue: I was not and am still not able to deploy the model using the Heroku git commands, nor with GitHub. It says: ”Your app does not include a heroku.yml build manifest”. (Everything being in the right directory) Resolved: Deployed the app using Container Registry (Built docker first and then pushed the image to Heroku.) Ref: stackoverflow, maybe there was an issue with the git LFS I set up. 5. Heroku memory exceeded - Resolved: Imported the model (size ~300mbs) inside the forecast function (only be loaded when forecast api is called) and implemented cache miss. So if the model was already present in the cache, it would not be loaded again.

---

4. FUTURE EXPERIMENT	
Reflect on the experiment and highlight the key information/insights you gained from it that are valuable for the overall project objectives from a technical and business perspective.	
4.a. Key Learning	The experiment helped us learn a lot about how to make a forecasting sales revenue model. Key points to remember are: - Model Performance: The ARIMA and SARIMAX models showed promising performance on test data, while comparing to the simple EWMA baseline. However, the ARIMA model's high training error raises concerns about its ability to capture training data complexity. - Model Could Be Better: The model could be better with handling data quality issues, advanced feature engineering, and hyperparameter tuning. - Effects on Business: Accurate revenue forecasting is crucial for inventory and sales management. Incorrect forecasts could lead to stockouts, overstocking, and associated costs. Why more experiments should be done: More testing is necessary because there are possible benefits and ways to make the model better. It's not a dead end; it's a chance to improve the model, make it work better, and get the most out of it for the business. To get the most out of the model, future experiments should focus on suggested potential improvements.

4.b. Suggestions / Recommendations	

1. As suggested in the Feature engineering, we can produce more pertinent features that will assist the model to perform better by better capturing the variation in the data. This will help the time series models learn better.

Deployment Scope: Advanced modelling, feature engineering, and data augmentation should improve predictive accuracy if this model needs to be deployed. Model interpretability, class imbalance, privacy, and ethics are also crucial.

1. Model Evaluation: Perform a thorough evaluation of the model's performance on a held-out validation dataset to ensure it meets business requirements.
2. Production Implementation: Deploy the model as a production-ready API or service, making it accessible to relevant stakeholders.
   Whats already been done: The model was first deployed into a docker container, tested the performance and any server issues. Further, it was advisable to deploy the model on Heroku web app (as a part of the assignment).
3. Monitoring and Maintenance: Implement a monitoring system to track model performance and retrain it periodically to adapt to changing patterns.
4. Integration with Business Processes: Integrate the model's forecast into inventory management, sales planning, and decision-making processes.
5. Feedback Loop: Establish a feedback loop to continuously improve the model based on real-world performance and evolving business needs.
6. Documentation and Training: Provide documentation and training to users and stakeholders to ensure seamless adoption.
7. Security and Compliance: Ensure data security and compliance with relevant regulations during deployment.
8. Scalability: Design the deployment to handle increased data volume and user load as the model's usage grows.

Appendix

1. McKinney, W., & others. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56).

2. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2

3. Waskom, M., Botvinnik, Olga, O'Kane, Drew, Hobson, Paul, Lukauskas, Saulius, Gemperline, David C, … Qalieh, Adel. (2017). mwaskom/seaborn: v0.8.1 (September 2017). Zenodo. https://doi.org/10.5281/zenodo.883859

4. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95.

5. Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.

6. Van Rossum, G. (2020). The Python Library Reference, release 3.8.2. Python Software Foundation.

7. Joblib Development Team (2023) Joblib/joblib: Computing with python functions., GitHub. Available at: https://github.com/joblib/joblib.

8. Tiangolo/FASTAPI: FASTAPI framework, high performance, easy to learn, fast to code, ready for production, GitHub. Available at: https://github.com/tiangolo/fastapi

9. Encode/uvicorn: An ASGI web server, for Python. 🦄, GitHub. Available at: https://github.com/encode/uvicorn

10. Asteriou, D., & Hall, S. G. (2011). ARIMA models and the Box–Jenkins methodology. Applied Econometrics, 2(2), 265–286.