# EXPERIMENT REPORT - Predictive

| | |
|---|---|
| **Student Name** | Kritika Dhawale |
| **Project Name** | AT2 - Machine Learning as a Service |
| **Date** | 7th October 2023 |
| **Deliverables** | dhawale_kritika-24587661-sgd_pipeline.ipynb<br><br>`*sgd_pipeline.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** | - The objective of this project entails the development of a predictive model utilising Machine Learning techniques to effectively anticipate sales revenue for a specific item within a designated store on a given date.<br>- 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 predictive 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.<br>- Hence, the precision of the model holds utmost significance in facilitating well-informed business determinations and attaining optimal operational efficacy. |
| **1.b. Hypothesis** | - “Is it possible to use historical sales data, item attributes, and store information in order to construct a dependable predictive model that effectively predicts the daily sales revenue for individual items in particular stores?”<br>- 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 ultimate goal is to deploy a reliable predictive model that aids in revenue forecasting, leading to better inventory management and improved business decisions. This follows; attaining a Mean Squared Error (MSE) that indicates a high level of prediction accuracy. While the specific numerical goal may vary depending on the dataset and business context, the objective is to minimize prediction errors and maximize the model's precision.<br>- Possible scenarios:<br>1. Successful Model: The model achieves lower error values, with MSE within an acceptable range. This scenario would result in improved inventory management, sales planning, and overall operational efficiency.<br>2. Moderate Accuracy: The model provides moderately accurate predictions but may have room for improvement. In this case, iterative model refinement and feature engineering may be needed to enhance performance.<br>3. Poor Accuracy: The model's predictions are not accurate/ MSE is high. This scenario would require a reevaluation of the modeling approach, data quality, or the need for additional features.<br>4. Overfitting or Underfitting: The model exhibits signs of overfitting or underfitting, indicating that the model complexity needs adjustment.<br>5. Data Limitations: Unexpected data issues or limitations may affect the model's performance, requiring data preprocessing or collection adjustments. |

---

| **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:**<br>- Removing unnecessary columns ["id", "dept_id", "cat_id", "state_id"]<br>- Added item_id and store_id to test set - test set was an extension to train set<br>- Filtered sales revenue by minimum non-zero percentage - Reduce low sales data samples to enhance data quality.<br>- Pivot the data frame: transform days' columns in rows, the units sold each day in a separate column - Input to model was given in the task.<br>- Repeated the same steps for the test set.<br>- There were 55% of zero sales revenue in train set - had to remove them as it resulted in bias.<br>- Saved the interim cleaned data into a pickle file<br>**Data Preprocessing (Included in model pipeline):**<br>- Separated numerical and categorical columns<br>- Performed Standard scaling on numeric data & one hot encoding on categorical<br>- Added "handle_unknown" as "infrequent_if_exist" (parameter in OHE) to handle unknown categories during prediction. By default it's set to "error", meaning it will rise error when the model is given any unseen value during transform.<br>**Future Steps:**<br>- I used the dask dataframe to make batches of data, it was working well in a different environment. But as soon as I tried to use it in poetry shell, there was |

| | |
|---|---|
| | some issue with the installation (No module named…). Could not resolve the problem due to time constraint. It can definitely save considerable memory while handling large datasets. |
| **2.b. Feature engineering** | - Merged date from calendar data<br>- Merged item's weekly sell price from weekly price data.<br>- Calculated daily total revenue (units sold x items weekly sell price for that particular week) - expected in the task outcome<br>- Extracted weekday, day, and year from the date to get date features - important as it affects the revenue.<br><br>**Potential ad-ons:**<br>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** | The **'SGDRegressor'** model from scikit-learn was used to make the prediction model. It is a linear regression model with Stochastic Gradient Descent (SGD) optimisation. These hyperparameters were picked out and tuned for this model:<br><br>- "loss": Set to "squared_error" to use the squared error loss function for regression.<br>- "penalty": Set to "elasticnet" to use both L1 (Lasso) and L2 (Ridge) regularisation, which gives you more options for choosing features.<br>-- "early_stopping": Set to "true" to allow early stopping during training, which helps prevent over fitting.<br>-'max_iter': Use 10,000 to set the maximum number of training iterations.<br>- 'tol': Set the tolerance for early stopping to 1e-3.<br>- "alpha": The level of regularisation is set to 0.001.<br>- "validation_fraction": Set to 0.2, this is the percentage of the training data that will be used for validation.<br>- 'n_iter_no_change': Set to 10 to show the number of iterations that did not improve the validation set before stopping.<br><br>**Reasons for this:**<br>SGD-based regression models work well with large datasets and can quickly handle feature spaces with a lot of dimensions. They are flexible and can be set up to work with different loss functions and types of regularisation.<br><br>**Tuning Hyperparameters:** We picked the hyperparameters to find a good balance between underfitting and overfitting, with the goal of making the model as simple and regular as possible. Regularisation and early stopping help keep people from overfitting.<br><br>**Hyperparameter Tuning with Grid Search (Tried didn't work):**<br>To enhance model performance and identify optimal hyperparameters, a Grid Search was conducted. The following hyperparameters were explored:<br><br>regressor__alpha: Regularization strength, with values [0.0001, 0.001, 0.01, 0.1].<br>regressor__max_iter: Maximum number of iterations, with options [1000, 5000, 10000].<br><br>This method was computationally expensive and so had to drop it out. |

| | |
|---|---|
| | **Not Trained Models:**<br>Other regression models, such as Random Forest, Gradient Boosting, or Support Vector Regression, were not trained. The models were picked based on how well they could be used, interpreted, and scaled up or down.<br><br>**Future Tests:**<br>Depending on the dataset and the needs of future experiments, it might be helpful to look into other regression models to see how well they work for the job.<br>Hyperparameter Tuning Hyperparameters must be fine-tuned for the model to work well. 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 performance metric utilised for evaluating the models was Mean Square Error (MSE).<br>- Mean Squared Error (MSE) on Train Set: 142.76<br>- Mean Squared Error (MSE) on Test Set: 152.88<br><br>The MSE for the test set is a little higher than the MSE for the train set, but it is still within a good range. This means that the model works well with data it hasn't seen before, but it could be better.<br><br>**Data Quality as an underlying issue:** We had to remove 55% of the training samples as there were zero sales. This potentially adds up to less training data and removing instances when the store was shut due to public holiday or something. Adding steps to the data preparation process to deal with such cases or data that doesn't match up could make the model even more accurate. |
| **3.b. Business Impact** | The conducted experiments provide evidence supporting the performance of the predictive model in optimising inventory management, sales forecasting, customer satisfaction, operational effectiveness, and overall profitability. Nevertheless, it is imperative to engage in the process of fine-tuning the model, effectively addressing any outliers, and optimising the quality of the data. These actions are crucial in order to minimise the adverse effects of inaccurate outcomes and fully harness the model's capabilities for the benefit of the business.<br>Overall, the results indicate that the model shows promise but may benefit from further refinement. |
| **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.<br>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”. Removed zero sales samples.<br>3. Importing a class from a function (relative import) - Resolved: Had to restart the kernel if there were any changes made in source files.<br>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 setup. |

---

| 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 predictive sales revenue model. Key points to remember are:<br><br>- Model Potential: The model showed promise by getting low MSE values on both the training and test sets, which shows that it can pick up on patterns of sales revenue for the given item, store and date.<br>- Model Could Be Better: The model could be better with handling data quality issues, advanced feature engineering, and hyperparameter tuning.<br>- Effects on Business: The experiment showed how important it is to make accurate predictions about revenue when managing inventory, planning sales, making sure customers are happy, running a business efficiently, and making money.<br><br>Why more experiments should be done:<br>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. Ensuring data quality and quantity was something that was lacking in this experiment and surely has some improvement scope.<br>2. 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.<br><br>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.<br><br>1. Model Evaluation: Perform a thorough evaluation of the model's performance on a held-out validation dataset to ensure it meets business requirements.<br>2. Production Implementation: Deploy the model as a production-ready API or service, making it accessible to relevant stakeholders.<br>**What's 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).<br>3. Monitoring and Maintenance: Implement a monitoring system to track model performance and retrain it periodically to adapt to changing patterns.<br>4. Integration with Business Processes: Integrate the model's predictions into inventory management, sales planning, and decision-making processes.<br>5. Feedback Loop: Establish a feedback loop to continuously improve the model based on real-world performance and evolving business needs.<br>6. Documentation and Training: Provide documentation and training to users and stakeholders to ensure seamless adoption.<br>7. Security and Compliance: Ensure data security and compliance with relevant regulations during deployment.<br>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