Final Report - Store Sales Predictions

Executive Summary

For the team's final project, we aimed to increase forecast accuracy by utilizing time series forecasting. To begin this project, we used a data set with sales information from a large retailer. This data set also included information ranging from oil pricing to holiday sales to cover various external factors that may influence predictions. To evaluate our models and success, we used the RMSLE equation, which is further detailed in the evaluation section of this report. Our methods for this project included trying several different regression-based models to determine which model had the most significant success and smallest RMSLE value. The Multi-Layer Perceptron model was the most successful and garnered a RMSLE value of 0.65. This model's runtime and accuracy made it a great and reusable model for the data set.

Improvements After the Last Presentation

Since the last presentation, we have focused on creating better graphical displays for the results of the MLP model. We also improved our RMSLE value by tuning the model's parameters to fit the data set better.

Problem Statement and Data Description

Globally, retailers have a firm reliance on sales forecasting to help predict the demands of their customers and be prepared. This reliance also requires strict accuracy in forecasts to ensure minimal profit loss. However, predicting the future demands of customers is a challenge as several factors affect their wants and needs. These factors include holidays, seasons, economic stress, and product promotions. Each one of these factors complicates predicting future sales, and ignoring these can lead to inaccurate forecasts. For the team's final project, we aimed to increase forecast accuracy by utilizing time series forecasting.

The data-set we used includes the following information:

- Training data (Store identities, Product types, Total product sales, and Product promotions)
- Test data (Store identities, Product types, Total product sales, and Product promotions)
- Store metadata (City, State, Type, and Cluster of similar stores)
- Oil price data (Daily oil price)
- Holidays and Events

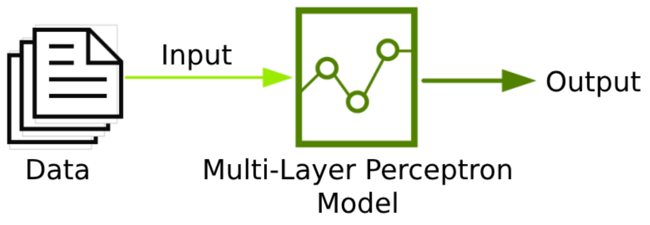
Team Approaches and Tools

At the beginning of the final project, the team decided to divide and conquer as we approached solving the problem. This plan included several team members coding basic versions of regression models to see if any model stood out as more successful initially. The models we tested included Linear Regression, Random Forest, Multi-Layer Perceptron, and XGBoost. The team specifically chose these models as research showed other data sets similar to ours encountered high success with the models.

The team used a variety of tools to create the models. The following is a brief outline of them:

- Pandas: For data loading, cleaning, and manipulation.
- NumPy: For numerical computations.
- Matplotlib: For data visualization.
- Scikit-learn: For machine learning model implementation and evaluation.

The source code includes a Python file for each model we tested on the data set. For example, MLP.py includes all Multi-Layer Perceptron computations. This includes data preprocessing, scaling, normalization, fitting, predicting, and graphing. The team chose this structure for organizational purposes and to allow easy separation of the different models. It also allowed the team to work on our respective models without concern of disrupting other members' work.



Metric and Evaluation

To evaluate the success of our project, we used the Kaggle competitions evaluation metric. This metric is the Root Mean Squared Logarithmic Error (RMSLE) and is represented by the following formula:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(log(1+\hat{y}_i) - log(1+y_i^2))}$$

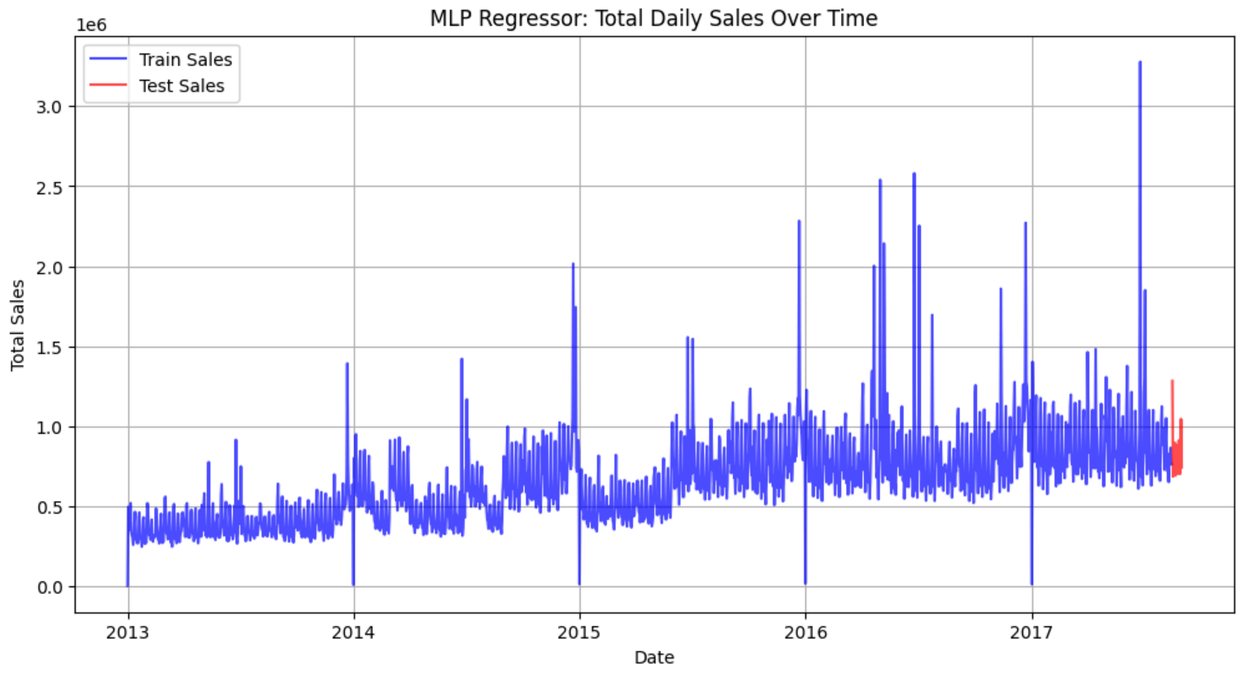
Where n is the total number of instances, $\hat{y}_i$ is the predicted value of the target for the instance, y_i is the actual value of the target for instance, and log is the natural logarithm.

To evaluate the model using this metric, the team has utilized the on website competition submission page. Upon submission of our predicted values, the Kaggle competition website uses the RMSLE metric and reports the score.

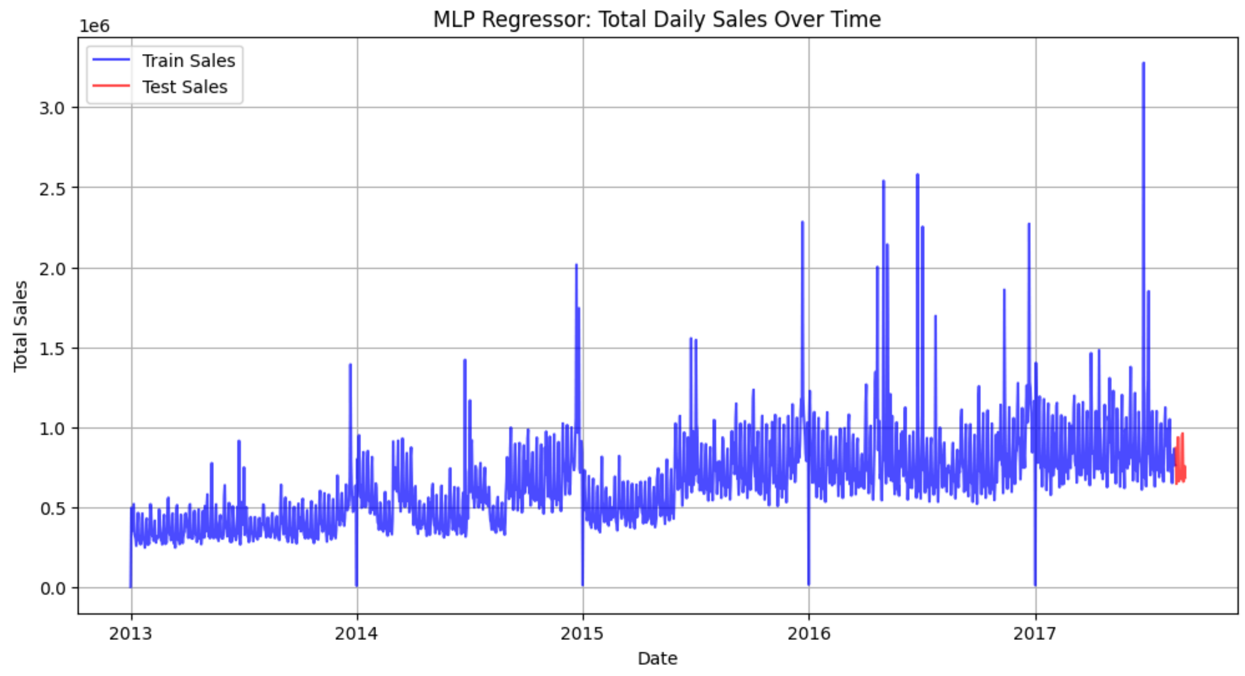
Project Results

Main Model - MLP

After testing several models with the provided data set, the team settled on using the Multi-Layer Perceptron after promising initial results. The team used a model with an initial framework that included scaling the data, normalizing the data, and categorizing specific values in the data set. This version of the model resulted in a RMSLE value of 0.66668 and an average run time of 12 minutes. The following is the sales prediction result in graph form:



With the project deadline approaching, the team focused on fine-tuning the parameters inputted into the model. The changes included adding an activation function, trying new learning …, increasing the number of hidden layers used, and the maximum iterations used. Combined with the preprocessing of the earlier sprint, we landed on our best model, resulting in an RMSLE value of 0.65078 with a run time of 14 minutes and 25 seconds. Though not as significant of an improvement as desired, this was a success. The following graph is the final version of our model and the resulting regression:



Comparisons and Analysis

The models we tested included Linear Regression, Random Forest, XGBoost, and Multi-Layer Perceptron. While the data produced an uptrend that could be generalized with a linear regression model, we found that the chart fluctuated too dramatically to be accurately modeled with a straight line. Therefore we decided to use different models better suited for this behavior. The Random Forest and XGBoost models were more appropriate for accurate predictions, utilizing decision trees for better results. These generated RMSLE values of about 0.83 and 1.38 respectively, which we evaluated as too high and therefore seeked further improvement. The Multi-Layer Perceptron model, which utilized neural networks to learn complex patterns, generated the most accurate prediction with RMSLE values settling below 0.67.

The team made several observations from the two MLP model graphs. Firstly, the initial model before tuning has a slight spike at the beginning of the time range. This spike differs from our final graph and indicates that our initial model might have suffered from overfitting or was overly reactive to sales fluctuations that can occur during specific seasons. Further, this spike would explain the slightly higher RMSLE value, as the deviation skewed the resulting sales predictions and accuracy. In contrast, the final graph demonstrates a smoother transition in sales per day that is more accurate for the time range. There are no spikes or any graphical indicators that show overfitting, which explains the smaller RMSLE value. Both models took over 10 minutes, with the longest being our tuned model. However, allowing the model ample time to fit into the data proved beneficial. The result improved the accuracy of the forecast and captured future sales for the retailer significantly better.

Challenges and Lessons Learned

Throughout the project, the team encountered several challenges that impacted project development. As we tried different models, we discovered that the run time for each varied significantly. Simple linear regression would fit a model very quickly, whereas MLP often takes 10 to 20 minutes. The high run time made it difficult to fine-tune our models because running and checking the output of the changed model took too long.

Other challenges within the project included determining what parameters would perform the best inside of our models. The MLP model has several input parameters, including hidden layers and activation functions. Choosing the best options was a challenge, which resulted in a significant amount of trial and error.

The team has learned from these challenges that it takes a significant amount of time to perfect a model. The multitude of input parameters specific to each model makes model tuning a complicated task but one with the highest importance. Furthermore, beginning these projects as soon as possible is recommended, as run-time can be incredibly high depending on the data set and model chosen.

References

Chamuditha Kekulawala. “Implementing Multilayer Perceptrons (MLPs) in Deep Learning.” *Medium*, 7 July 2024, medium.com/@ckekula/implementing-multilayer-perceptrons-a-mathematical-guide-c528ea89704d. Accessed 21 Apr. 2025.

GeeksforGeeks. “MultiLayer Perceptron Learning in Tensorflow.” *GeeksforGeeks*, 3 Nov. 2021, www.geeksforgeeks.org/multi-layer-perceptron-learning-in-tensorflow/. Accessed 21 Apr. 2025.

“MLPRegressor.” *Scikit-Learn*, 2015, scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html. Accessed 21 Apr. 2025.