Schedule Recommender Tool for Servicing Express

Jeremy Flics and Nisarg Polra

Wholesale Lending Services (WLS) is a large division within JPMorgan Chase (JPMC) that services thousands of loans every year. With about 2,200 WLS employees across the world, having an analytics tool that monitors and evaluates the task work assignments using current and historical data will increase the overall productivity of the organization. As part of the Multidisciplinary Design Program (MDP), our team provided a machine learning model integrated with a stand-alone application that predicts future workload and recommends work assignment to improve on-time delivery. Currently, supervisors have limited visibility of the data and do not have access to any tools to support this decision-making. We aim to provide better access to crucial information for allocating work, such as the volume of tasks, and number of available employees with a certain skill set to assist supervisors in making strategic decisions.

Our model was integrated with a data pipeline and a user-friendly front-end application to assist supervisors in assigning cases to processors. The machine learning model was trained on data from an internal database which holds long-term historical data. After training, the model runs in the back-end using recent short-term data to make predictions while the supervisors access the user-friendly GUI where statistics and graphics based on the predictions are displayed. We provided additional functionality for supervisors to be able to configure the tool to display information according to their specific needs.

Success was measured based on model performance and reliability, seamlessness of integration into existing user interfaces, and usability testing to determine how useful the tool is for supervisors in making strategic work allocation decisions. We selected metrics to assess performance, including accuracy, AUC, precision, recall, specificity, and F1 score. Ultimately, our “North Star Metric” was the average completion time of loan processing, as aligned with the business-wide intention of WLS.

Overall, the main goal of the project was fixed by our mentors. This goal is to support supervisors in the WLS division in making strategic decisions about allocating work to processors across the world. Moreover, the technology and the database used to implement our ideas both come from internal sources. A joint responsibility of the sponsor mentors and our team was to decide the primary use cases that our model will target: forecasting the volume of tasks, making recommendations about allocating work, etc. Our team had the authority to make decisions regarding the implementation of the model, while being informed by our faculty and sponsor mentors.

Features Offered:

1) Machine Learning model for forecasting the volume of loan servicing requests 3 weeks in advance.
2) Permanent schedule for the “follow the sun” model.
3) Data Pipeline built using AWS services
4) User Tool built using React and TypeScript

Initially, we divided ourselves into four sub-teams: analysis, machine learning, user tool design, and data pipeline architecture. Substantial progress was made on a statistical analysis of our data to recommend a baseline schedule. Insights generated from the analysis informed our end of summer presentation to the WLS leadership team, who are now using our work to continue generating a baseline schedule. With our analysis in production, we are now able to direct our full focus to iterating on our user tool.

From a machine learning perspective, we compared the performance of several models before developing one. Of our early prospects, long short-term memory (LSTM), and decision trees of various types were the most promising. After comparing the models using walk forward validation, we determined that gradient boosted regression trees (GBRT) were the best choice. This is because they had the lowest error on our dataset which we think is due to the fact that they tolerate a combination of categorical and numerical variables better than the other models. Our GBRT model had a mean absolute percent error of 26.9%. While this is above our target value, we have currently deprioritized model optimization in favor of completing our system architecture. When we do return to modeling in the latter half of the semester (weeks 24 - 29), we think our goal can be achieved by using hyperparameter tuning techniques like grid search.

For the user tool we worked iteratively on designing and mocking up a user interface (UI), getting feedback from our product owner and a user experience designer along the way. Based on our final mockup, we completed this interface using React as well as UI Toolkit, an internal user interface library that adheres to JPMC style conventions. In fact, one stretch goal that we think we will hit close to the end of the semester is giving the user the option to look at volume predictions for payments and advances separately. The team working on the user application are now transitioning their focus to writing the algorithm that transforms volume predictions into schedule recommendations.

Figure 1. Landing page of the user application. An alert count is on the top with a date selector underneath. Two graphs are shown. The left shows the hourly loan servicing request volume prediction of the selected day. The right shows the schedule recommendation for that day.

In order to connect our application to the model, we had to design a data pipeline. Given that our tool is designed to only be used by senior management, we thought running a server would be costly and wasteful. Instead, we developed an event-driven serverless solution that hosts our tool as a single page application (SPA). In line with firm standards, this architecture was designed using Amazon Web Services.

Development Process

In this project we intended to create a predictive tool to assist JPMC processors with their tasks. Specifically, we leveraged large amounts of historical data to inform a tool that performs the desired task independently. Broadly, machine learning is a branch of artificial intelligence in which systems can learn from data, identify patterns and make decisions, largely independent of human intervention. In the following section, we will discuss the machine learning workflow, model methods, feature selection, and success evaluation.

The Workflow of Building a Machine Learning Product

Like any project, product management strategies are important to the development of a machine learning model. Before getting wrapped in the weeds of massive databases and intense algorithms, it's important to establish the key steps of creating your model. Below, we find an example outline of a machine learning workflow, sourced from Yael Gavish's Medium article, *"Developing a Machine Learning Model from Start to Finish."* This is the workflow that our team committed to, and the project management style with which we have become familiar.

1. ***Ideation***
    a. Align on the problem
    b. Choose an objective function of the model
    c. Define quality metrics
    d. Brainstorm inputs/features
        i. Determine Phase #1 Features
        ii. Determine Phase #2 Features
    e. Create a backup plan + product.
2. ***Data Preparation***
    a. Data collection
    b. Data cleanup and normalization
3. ***Prototyping and Testing***
    a. Build prototype
    b. Validate and test prototype
    c. Iterate
4. ***Productization***
    a. Increased data coverage
    b. Scale data collection
    c. Refresh data
    d. Scale models
    e. Check for outliers
5. ***Build Out + Delegate Fulfillment of System Requirements***
    a. Real time requirements.
    b. Data and model dependencies.
    c. Data collection frequency.
    d. Data collection methods.
    e. Pipeline Dependencies

6. ***Presenting Results to Stakeholders***
    a. Backdating
    b. Explaining your methods and inputs
    c. Exposing some of the underlying data
    d. Simplifying and only showing select results to facilitate decision making
    e. Defining the new metric (ie uplift, or whatever the model outputs)
    f. Precision doesn't always matter.
    g. Strategically providing access to raw data.
7. ***Future***
    a. Ensure sustainability of the model
    b. Share further product ideas

Feature Selection

In machine learning, feature selection is the process of selecting features (also called "characteristics" or "variables") to be factored into your model. Any machine learning problem starts with massive amounts of data, with each datapoint having a multitude of features to its identity. If you're trying to answer or predict a certain feature of a datapoint, it is reasonable to only take related features into account while training your model.

As a simplified example: let's say we have a massive historical database of fruit that has been sold in a grocery store. Each datapoint has a fruit name, weight, color, shape, flavor, farming region, time of sale, date of sale and grocery store location. Now, we want to train a model on this historical data to classify new data points. These new data points have all the same features as listed before, but without the fruit name. This is what the model is trying to predict. In building this model, we would want to include features so that the model can learn, for example, that if a fruit is 70 grams, red, round, grown in washington, and sweet, it is likely called an "apple." However, we do not want to include time of sale, because it is likely largely irrelevant to the fruit name. Also, we do not want to include *both* farming regions and grocery store locations, because these features might contribute the same insights to the model.

This was a simple example about how a data scientist may approach feature selection for a model. However, there are many more considerations and strategies when it comes to feature selection. As a general rule, adding more features makes models more complex, and thus increases the chance of overfitting. When overfitting happens, the model is suited to give ideal output on the training dataset, but the model fails to generalize to data it has not seen. To allow for simpler models and better generalizations – especially with high-dimensional datasets – it's wise to cut out all features except for the most useful ones. To do this, there are three basic supervised methods.

First, and often the simplest of the three: *Univariate Statistics*. This method computes whether there is a statistically significant relationship between an individual feature and the target (ie, the thing we're trying to predict). The features related with the highest confidence are selected. Though this method is simple and quick, it fails to consider multivariate relationships in which

two features may contribute to the model when combined, but not when independent (Müller, 2018).

Next, *Model-Based Feature Selection* uses a supervised machine learning model to determine the importance of each feature, and rank by the same. This method considers all features at once, and is capable of capturing relationships between features that may contribute to the model (Müller, 2018).

The final method is *Iterative Feature Selection.* With the highest computational cost, this tactic involves a series of models, with each model containing varying feature subsets. As a model is run, it determines the least important feature, creates a new model without that feature, and runs the model again until a predetermined number of selected features has been reached (Müller, 2018).

Beyond these three computational methods of feature selection, it is important to consider expert knowledge – the heuristics and intuitions that people working in the field of application often have and depend on. Returning to our apple example, a grocer may be able to recognize that pineapples sell much more before on the first day of summer vacation. This is an insight that a computational feature selection method may be unable to identify. But, if we add a boolean feature indicating that a fruit was or was not sold on the first day of summer, we may be able to factor that information into our model. Adding a feature does not force a machine learning model to use it, and even if the first day of summer information turns out to be noninformative for fruit names, augmenting the data with this information doesn't hurt. With the correct combination of these strategies and methods, feature selection can be an important contributor to the construction and success of your model.

Training the Model

The first step in training a model is to split the dataset into three different categories: training, validation, and testing. This is to ensure that we reserve some data for evaluating the model performance on data that it has not seen yet. Testing the model on the same data it was trained on does not accurately reflect its true performance. As a starting point, the dataset can be split by randomly assigning 80% of the data in the training set, 10% in the validation set, and 10% in the test set (Prasanna, 2020).

To establish a useful machine learning model, a model that can turn an input into a desired output, we first need to conduct training. Training a model involves analyzing a training dataset, which is a set of input data with the corresponding correct output data. Through training, the model tries to learn the optimal parameter values that can generate the ideal output. However, it is important to not train the model on the entire dataset because that makes it difficult to evaluate the performance of the model.

Following the training dataset, we apply the validation dataset to tune the hyperparameters, such as the number of hidden units in each layer for the Neural Network, or the regularization factor to minimize overfitting, etc. Moreover, the validation dataset can help obtain some

performance characteristics. For example, for the classification models, the validation dataset can provide performance measures such as accuracy, sensitivity, specificity, etc. (Prasanna, 2020).

Evaluating Success

The final stage is to apply the testing dataset to evaluate the performance of the model with its final parameters and hyperparameters. This is done by first feeding each individual datapoint from the test set into the model and recording the prediction. Then depending on the problem and the model, different performance metrics can be used. For our project, these are the metrics that best reflect the success of the model:

- Forecasting model to predict the volume of tasks in the future:
    - Mean Squared Error (MSE): This metric measures the average squared difference between the model prediction and the actual value across all data points in the test set. A lower MSE indicates a better forecasting model (Jordan, 2018).
    - Root Mean Squared Error (RMSE): RMSE is basically the square root of the MSE. In some cases, it is beneficial to report the RMSE as opposed to the MSE since the units of RMSE are the same as the desired output data (Brownlee, 2021).
- Classification model to assign tasks to processors (Jordan, 2018):
    - For binary classification (classifying as either A or B) or checking the presence of a condition
        - True Positive: number of data points correctly predicted as having the condition or belonging to category A
        - False Positive: number of data points predicted as having the condition or belonging to category A but actually have the condition or belonged to category B
        - True Negative: number of data points correctly predicted as not having the condition or not belonging to category A
        - False Negative: number of data points predicted as not having the condition or not belonging to category A but actually have the condition or belonged to category A
        - Sensitivity (True Positive Rate): True Positive / (True Positive + False Negative). Sensitivity measures among all the points classified as having a condition or belonging to category A, how many were predicted correctly
        - Specificity (True Negative Rate): True Negative / (True Negative + False Positive). Specificity measures among all the points classified as not having a condition or not belonging to category A, how many were predicted correctly
    - For multi-class classification (Verma, 2021):

- Accuracy: the number of points predicted correctly / total number of predictions conducted
- Weighted Accuracy: For a particular class, it receives more weight, meaning if classified correctly, weighted accuracy would be higher than the regular accuracy, or vice versa.

Machine Learning Methods

When constructing a machine learning product, it's extremely important to select the correct model method. If your selected model does not fit your data and problem space, any results produced will be trivial and inaccurate. Most models fall under two categories: Regression or Classification.

Regression

Regression is applying quantitative inputs to derive a linear relationship among the different features to predict a final quantitative output. Below is an example of a regression model:

$$f(x) = \beta_0 + \beta_1 e^{x_1} + \beta_2 \ln(x_2) + \beta_3 x_1 x_3.$$

Essentially, the goal of training for regression is to find the optimal values for β, so that f(x) on average is as close as possible to the true value . The individual x values represent the corresponding values of the particular features of one case. For example, if we are interested in the number of species on an island, the predicted number of species can be considered as f(x). We choose to use geographic information to predict the number of species. Hence, x_1 can be the area of the island, x_2 can be the elevation, x_3 can be average temperature, and we get the following model:

$$\text{Species} = \beta_0 + \beta_1 \cdot \text{ Area} + \beta_2 \cdot \text{Evaluation} + \beta_3 \cdot \text{ Temperature}.$$

The goal of training will be finding the optimal values for β_0, β_1, β_2, and β_3, which describes the relationship between number of species and the area, elevation, and temperature of the island.

Classification

Classification is the process of predicting the class given the input data. For example, predicting whether an email belongs to the spam folder or not, predicting the correct animal in the image, or predicting the type of loan given some information.

Typical classification algorithms include Decision Trees, k-Nearest Neighbors (kNN), Neural Network.

Decision trees conduct binary classification, which is classifying something as either A or B. From training, decision trees derive rules that divide the input into a subsection and ultimately to either A or B (Asiri, 2018). For example, to classify if a fruit is a grape or an apple, you can first ask if the color of the fruit is green or not, then ask if the fruit has greater than 5 seeds, then if the fruit is greater than 1 pound, etc. The training process will decide the questions and order to ask the questions to derive the answer efficiently and accurately.

k-Nearest Neighbors tries to find ways to group the different clusters, and inputs in the same cluster are considered in the same category. The training process involves minimizing the distance between input in the same temporary clusters and adjusting which cluster an input data belongs to minimize the distances among input data in the same cluster. The number of clusters is set by the user (Brownlee, 2020).

Neural network tries to find non-linear relationships between the features to find a value to conduct classification. Often, neural networks conduct multi-class classification, which is that the model can conduct prediction on more than 2 objects. For example, given a picture of an animal, the model can predict whether or not it is one of a hundred animals it was trained on.

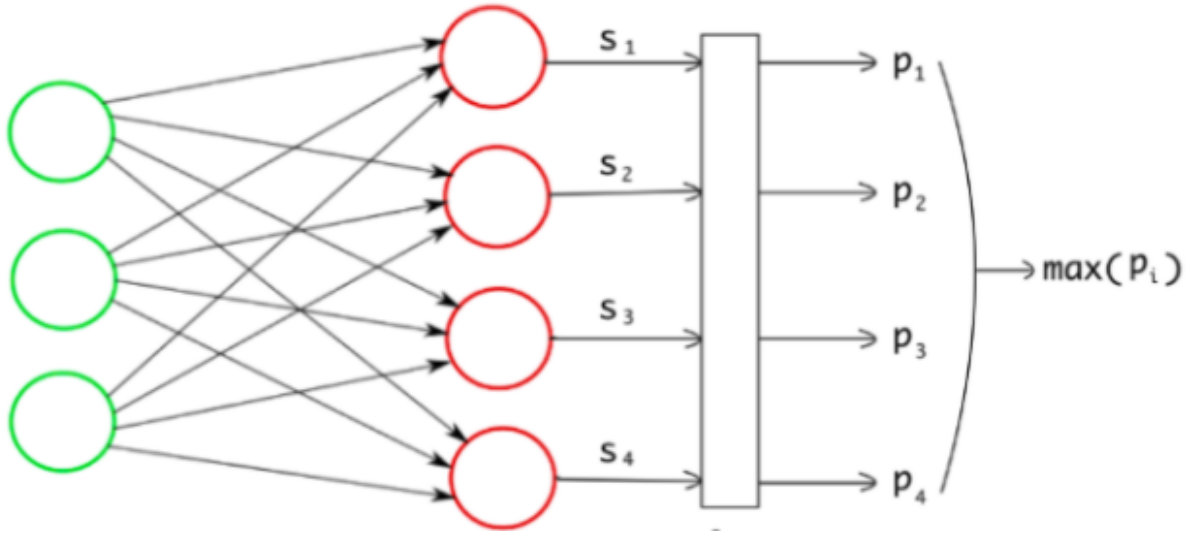

Figure 1. The first layer, made up of green nodes, is the input layer. The green nodes are the values of each feature selected for a particular data point. The layer(s) in the middle are the hidden layers. The final layer, shown as p in the above figure, is the output layer (Verma, 2021).

The lines connecting nodes from different layers represent the weight assigned to each feature feeding into the next layer. For example, the values going into the first red node in Figure 1 will be the sum of all the input node values multiplied by their respective weights. After going through the next layer, an activation function, a nonlinear function will be applied. For example, an activation function could be Relu, where negative numbers become 0 and positive numbers stay constant. The number of hidden layers can be set by the user. The output layer will have the same number of output nodes as the number of different classes it was trained on. Each output node in the output layer represents the probability that the data point belongs to a class, and the output node with the greatest probability is the final prediction for the datapoint (Verma, 202). The training process involves some sort of gradient descent as described in the optimization section.

Results and Discussion

Overall, our project was able to address several problems and inefficiencies with the scheduling policy used by supervisors on a daily basis. The permanent schedule produced as a result of our analysis was a starting point for the “follow the sun” model that has been desired by our sponsor mentor. The analysis also provided a stable benchmark for our schedule recommendations to be compared against. In the future, the permanent schedule can be modified to increase focus on factors like processor happiness and consistency.

Moreover, our work on the data pipeline in AWS resulted in a valuable example for JPMC as our team was one of the first teams to adopt the public cloud that the company desires to move to in the future to save costs and reduce complexity. As a team, we faced several challenges due to the lack of documentation and resources for this method within our sponsor organization. However, this facilitated increased communication and collaboration with full time engineers to get help in the office when needed. We reached out to the data scientist on our team on countless occasions to understand the machine learning training process within JPMC. In return, we shared valuable information with him when we understood how to host the machine learning model on the cloud as that was a relatively new procedure at the time.

Finally, we were able to deliver an integrated user tool and API to our sponsor which received immense approval from senior management. During the development process, we had a tight feedback loop with our manager and product owner to ensure that we were meeting requirements. This increased clarity and even though our project scope changed some times, it prevented any differences in expectations during the concluding phases. This project was an interesting learning experience and our team enjoyed implementing our classroom learnings to an industry project.

Bibliography

Asiri, S. (2018, June 11). *Machine learning classifiers*. Medium. Retrieved January 30, 2022, from https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623

Brownlee, J. (2020, August 19). *4 types of classification tasks in machine learning*. Machine Learning Mastery. Retrieved January 30, 2022, from https://machinelearningmastery.com/types-of-classification-in-machine-learning/

Brownlee, Jason. (2021, February 15) “Regression Metrics for Machine Learning.” *Machine Learning Mastery*. Retrieved January 30, 2022 from https://machinelearningmastery.com/regression-metrics-for-machine-learning/

Gavish, Yael. “The Step-By-Step PM Guide to Building Machine Learning Based Products | by Yael Gavish | Medium.” Medium, Medium, 25 July 2017, https://medium.com/@yaelg/product-manager-pm-step-by-step-tutorial-building-machine-learning-products-ffa7817aa8ab.

Jordan, J. (2018, August 25). *Evaluating a machine learning model.* Jeremy Jordan. Retrieved January 30, 2022, from https://www.jeremyjordan.me/evaluating-a-machine-learning-model/

Müller, Andreas Christian, and Sarah Guido. Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media, 2018.

Prasanna, S. (2020, September 24). *A quick guide to managing machine learning experiments*. Medium. Retrieved January 30, 2022, from https://towardsdatascience.com/a-quick-guide-to-managing-machine-learning-experiments-af84da6b060b

Verma, S. (2021, October 5). *Multi-label image classification with neural network: Keras*. Medium. Retrieved January 30, 2022, from https://towardsdatascience.com/multi-label-image-classification-with-neural-network-keras-ddc1ab1afede