Reducing Oil Well Downtime with a Machine Learning Recommender System

by

Jesús Madrid
B.S. Industrial Engineering

and

Andrew Min
B.S. Industrial and Systems Engineering

SUBMITTED TO THE PROGRAM IN SUPPLY CHAIN MANAGEMENT
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE IN SUPPLY CHAIN MANAGEMENT
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2020

Signature of Author: _____
Jesús Madrid
Department of Supply Chain Management
May 8, 2020

Signature of Author: _____
Andrew Min
Department of Supply Chain Management
May 8, 2020

Certified by: _____
Dr. Cansu Tayaksi
Postdoctoral Associate
Capstone Advisor

Accepted by: _____
Prof. Yossi Sheffi
Director, Center for Transportation and Logistics
Elisha Gray II Professor of Engineering Systems
Professor, Civil and Environmental Engineering

Reducing Oil Well Downtime with a Machine Learning Recommender System

by

Jesús Madrid

and

Andrew Min

Submitted to the Program in Supply Chain Management
on May 8, 2020 in Partial Fulfillment of the
Requirements for the Degree of Master of Applied Science in Supply Chain Management

ABSTRACT

The oil and gas industry plays an important role in the world's Gross Domestic Product by providing energy resources to the world. With the price for oil commodities falling in recent years, oil and gas companies require high operational efficiency in order to maintain profits. Unplanned downtime leads to high unnecessary costs representing on average 7.95% of the cost structure of companies in this capital-intensive industry. As a solution, companies have turned to advanced analytics and Big Data to reduce downtime and maintenance costs. This study involves the development of a machine learning recommender system intended to reduce unplanned downtime at oil well facilities. The developed recommender system uses the similarity among customers to predict future purchases and make product recommendations. Predictions are a function of the k-nearest neighbors to each customer, determined using the Euclidean distance or cosine similarity. We followed a binary classification machine learning approach with imbalanced classes by first splitting historical sales data into a training and testing dataset. Then we used the F-2 score and Precision-Recall curve to validate the models' performance in making accurate recommendations. Recommendations group similar products or services together, reducing the number of times an oil well is taken down for maintenance, therefore reducing downtime. Our results show that this recommender system could lead to a reduction of 1.7 days of downtime and produce cost savings of $2.5 million per customer per year, equivalent to 6.44% savings. The additional products or services sold could lead to additional revenue of $660K per year for the sponsoring company. The recommender system was based on one specific product line within the company, so we believe there is additional opportunity to scale it for larger downtime reduction and increased revenues.

Capstone Advisor: Dr. Cansu Tayaksi
Title: Postdoctoral Associate

ACKNOWLEDGMENTS

We would like to thank the MIT Supply Chain Management program for giving us the opportunity to work with the some of the brightest students, professors, and colleagues in the industry. We would especially like to thank our advisor Dr. Cansu Tayaksi for her knowledge and support throughout our entire capstone project. We would like to thank our sponsoring company Baker Hughes for the opportunity to partner with them and work on this machine learning project. Thanks to Sandeep Pandey and Anjali Mishra for their support on our capstone project. Thank you to Toby Gooley for her support in coaching us in writing our final capstone report.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

As the price of oil has fallen in recent years, profit margins for oil and gas companies have narrowed. Operational efficiency and reducing downtime for critical equipment has become vital to maintaining profits. Companies are beginning to lean on advanced analytics utilizing Big Data and machine learning to reduce downtime and costs.

The sponsoring company is Baker Hughes, an energy technology company leading the way on using data and machine learning to improve oil and gas operations. It is one of the largest oil-field service providers in the world with operations in over 200 countries. Baker Hughes' customers seek to minimize unplanned downtime in their oil facilities due to maintenance or repairs. The main objective of this project is to use historical customer sales data to build a recommender system powered by machine learning. This system will use similarities among customers' purchase history to make predictions for additional products or services. These predictions will anticipate customers' needs, avoiding future downtime in oil facilities and create attractive savings opportunities for oil and gas organizations.

1.1 Problem statement

Oil and gas companies face major challenges in monitoring and controlling their capital expenditures to ensure their profitability. Especially when oil prices are low, these organizations strive to maintain the production of energy products at minimal costs.

These companies are capital-intensive by nature, which means they require large amounts of upfront investments in fixed assets such as plants and equipment. These assets require constant maintenance and monitoring; however, they are still subject to failures due to unexpected events. This can result in downtime and disruptions of the complex global supply chain within the energy industry.

Downtime costs are highly expensive for oil and gas companies. Whenever an oil well or a refinery shuts down for unplanned reasons, production halts, workers sit idle, parts and equipment have to be urgently purchased and the flow of materials is interrupted (Christensen, Graf, & Yeung, 2013).

Christensen et al. (2013) cite a study conducted by the Hydrocarbons Publishing Company based on data provided by the United States Department of Energy that reveals that between 2009 and 2012 there were over 1,700 refinery shutdowns in the United States, equivalent to 1.2 shutdowns per day. The same study indicates that 46% of these shutdowns were caused by mechanical breakdowns, 23% due to maintenance, 19% because of electrical power outages and 12% for other[1] reasons.

Planned or unplanned downtime causes disruptions to oil and gas companies. Planned downtime is associated with preventive maintenance, which is regularly performed to lessen the likelihood of equipment failure. Planned downtime allows managers to anticipate disruptions in the supply chain, adequately assign resources, and make the shutdown as least disruptive as possible for the company's operations. Despite oil and gas companies' attention to planned maintenance, 54% of downtime happens because of unexpected events (Christensen et al. 2013).

A public report from the sponsoring company indicates that "averaging just over 27 days of downtime each year, offshore oil and gas organizations experience $38 million in financial impacts from unplanned downtime. For the worst performers, the costs can be upwards of $88 million." (Baker Hughes Company, 2016).

These numbers stress the importance of having efficient maintenance systems in oil and gas facilities. As seen in Figure 1, 24% of organizations follow a reactive maintenance approach, where maintenance is executed after the failure has occurred. Alternatively, 46% of organizations in the industry use a planned

[1] Mostly fires and work-related incidents.

maintenance approach, where maintenance is conducted in regular time periods, independent of the state of the equipment or machinery. This second approach can yield situations where companies conduct service and replacements for assets that may not have needed maintenance in the first place, often leading to higher maintenance costs.

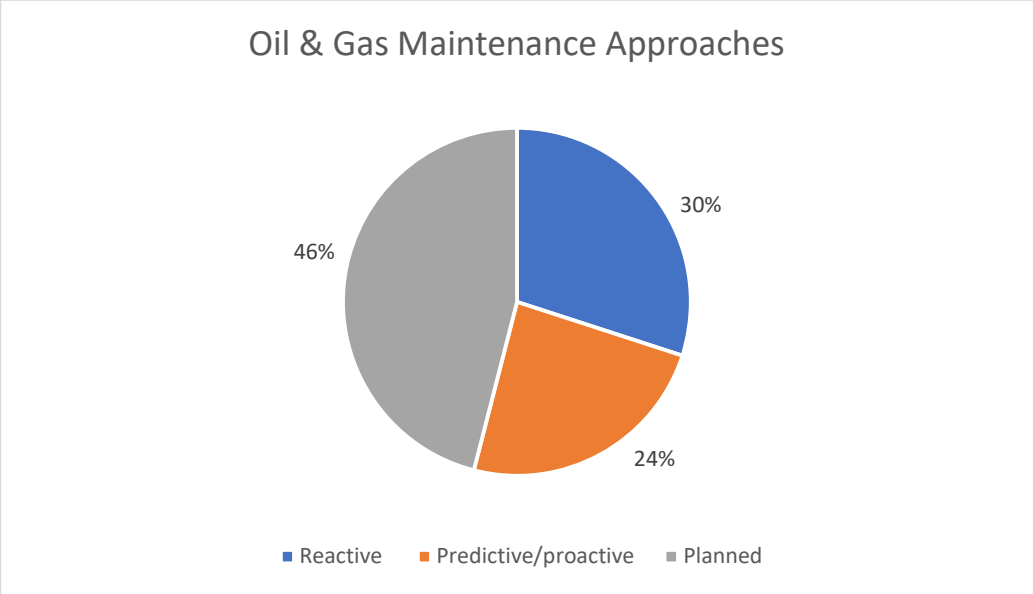

Figure 1. Approaches to equipment repair and maintenance. Source: The Impact of Digital on Unplanned Downtime. Retrieved from bhge.com: https://www.bhge.com/sites/default/files/2017-12/impact-of-digital-on-unplanned-downtime-study.pdf, 2016.

Modern technologies allow the industry to develop predictive approaches for maintenance based on data and analytics. Advanced analytics use sophisticated tools to discover insights, make predictions and generate recommendations from large amounts of data (Rose, Berndtsson, Mathiason, & Larsson, 2017). Among many techniques, machine learning can be used to predict equipment failure, anticipate maintenance needs, and execute service only when it is truly required.

Baker Hughes Company (2016) cite a private study conducted by Kimberlite International Oilfield Research revealing that the financial impact is 60% lower for companies using predictive approaches based on data

and analytics. Nevertheless, “Fewer than 24% of operators describe their maintenance approach as a predictive one based on data and analytics. The rest either took a reactive or time-based approach.”

One of the sponsoring company’s goals is to deliver the lowest cost per barrel to its customers. Baker Hughes utilizes data and technology in its products and services to improve productivity and efficiency to deliver the lowest cost. Reducing the impact of unplanned downtime in cost structures will lead to higher profitability in the industry.

How can oil and gas organizations reduce their downtime costs? This project will develop a recommender system powered by machine learning to anticipate customer needs for products and services. This system will recommend additional products or services to group together, ensuring customers make the right purchases at the right time. Combining products or services will reduce the number of times oil facilities need to be serviced, leading to a reduction in downtime. When Baker Hughes’ customers combine services, this will also lead to additional revenues for the sponsoring company.

2. LITERATURE REVIEW

This section presents a formal literature review about the current state of the oil and gas industry and how unplanned downtime is impacting the industry. First, we present the main components of the supply chain for oil and gas organizations and the different equipment maintenance strategies to compare their benefits. Next, we highlight the benefits of advanced analytics used in predictive maintenance approaches. Then, we show how machine learning and recommender systems can benefit the industry. Finally, to identify the best validation metrics, we refer to scientific works on recommender systems and collaborative filtering algorithms for a classification machine learning model with imbalanced datasets.

2.1 Overview of the oil and gas industry

The oil and gas (O&G) industry comprises of companies that explore, develop, and operate oil and gas fields, producing energy commodities that are sold across international markets. Although the oil and gas industry's contribution to the global economy has declined in recent years, it continues to be an important contributor to the world's Gross Domestic Product (GDP) (Leach, 2019).

The nature of these commodities makes their prices highly dependable on production levels, global economic growth, and market speculation. Traditionally, prices in the oil and gas industry have been very volatile. Crude oil, refined petroleum and natural gas prices are more volatile than 95% of other commodities, with volatility levels over 13% (Regnier, 2007).

In the last five years, oil prices have suffered wild fluctuations. Prices have fallen and revenues have decreased by 8.1% per year ($3.3 trillion). Although future forecasts project more stability in oil prices, total revenues are expected to be around 40% less compared to 2013 (Leach, 2019). This instability in revenues has forced oil and gas companies to make continuous efforts to ensure their profitability by reducing costs and investing in new technologies (especially in the gas segment) (Leach, 2019).

Supply chain efficiency is the main driver to reduce costs in the oil and gas industry. In the words of Christopher M. Chima from California State University: *"…the main challenge facing the oil and gas industry is not the availability of oil and gas resources, but putting these reserves into production and delivering the final products to consumers at the minimum cost possible*" (Chima, 2007). The use of tubular shipping in pipes, high vertical integration opportunities and little differentiation in the produced goods are advantages that can turn large and complex global supply chains into simpler and more cost-effective systems.

The supply chain of an O&G company can be broken into four areas (Chima, 2007):

1. Exploration: seismic, geophysical, and geological activities
2. Production: drilling, reservoir, production, and facilities engineering
3. Refining: requires complex facilities and operations
4. Commercialization: transportation and marketing activities

Production and refining require large investments in fixed assets. In fact, the oil and gas industry is considered to be one of the most capital-intensive industries. This means it relies heavily on large-scale capital equipment like offshore drilling and production platforms, rigs, and refining facilities (Chima, 2007).

These assets require high levels of maintenance and are often subject to failure. The main causes of failures stem from corrosion of pipelines, accumulation of oil in pipelines, sanding of well barriers, failure of equipment such as pumps and compressors, rotating equipment, oil and water separators, and heat exchangers (Telford, Mazhar, & Howard, 2011). The unexpected breakdown of these assets leads to unplanned downtime and higher cost structures.

2.2 Unplanned downtime in the oil and gas industry

Downtime in the oil and gas industry is extremely costly. The four most used maintenance models used across the industry are total productive maintenance (TPM), condition-based maintenance (CBM), reliability-centered maintenance (RCM) and condition monitoring (CM). TPM and RCM rely on planned maintenance schedules, resulting in expensive over-maintenance. (Fraser, 2014).

Condition-based maintenance tends to be the most suitable model for capital-intensive organizations such as oil and gas producers (Fraser, 2014). CBM is based on deterministic and probabilistic models. It consists of monitoring different parameters of a system[2] for potential failures. These parameters can be temperature, humidity, vibration levels, noise, or corrosion. This monitoring approach avoids making unnecessary replacements and maintenance, thus reducing the risk of over-maintenance costs (Al-Najjar & Alsyouf, 2003).

Figure 2 shows that unplanned downtime is 36% lower for companies in the oil and gas industry that use predictive methods and CBM. This results in 60% less downtime costs, or an average of $34 million of savings each year (Baker Hughes Company, 2016). The x-axis shows three maintenance approaches versus the left y-axis which is the cost of downtime in millions of dollars. The top gray line matches the scale of the right y-axis and displays the unplanned downtime rate for each maintenance approach. The bottom gray line depicts the annual unplanned downtime days for each maintenance approach.

[2] Could be equipment, machinery or even an entire facility.

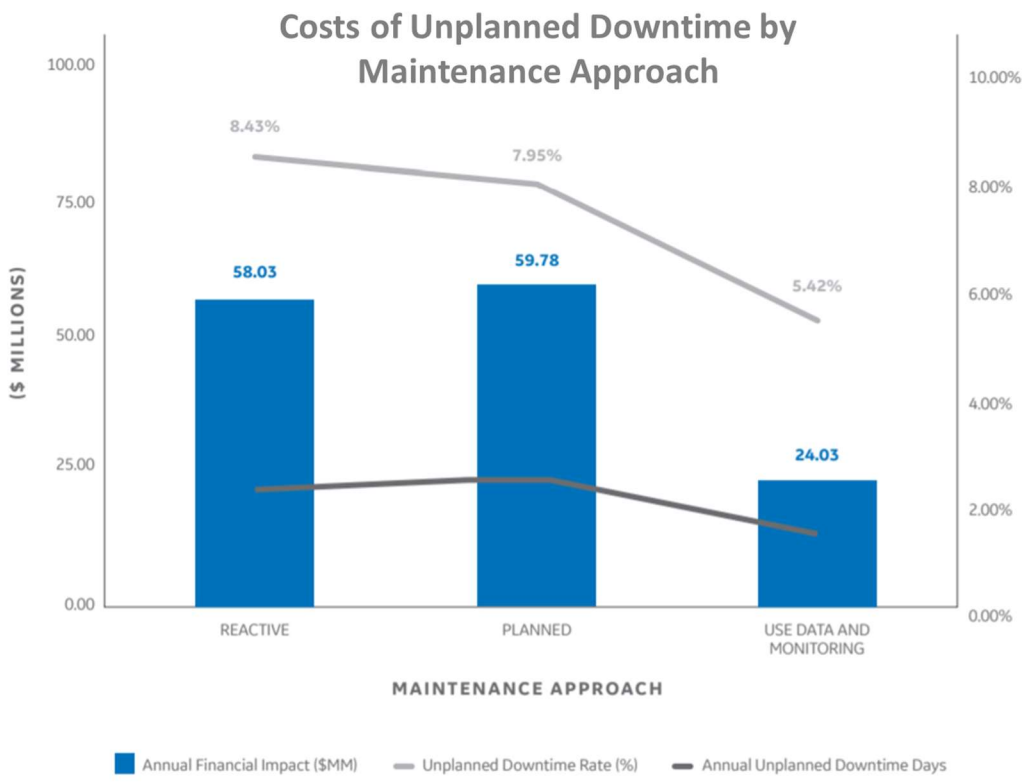

Figure 2. Costs of Unplanned Downtime by Maintenance Approach. Source: The Impact of Digital on Unplanned Downtime. Retrieved from bhge.com: https://www.bhge.com/sites/default/files/2017-12/impact-of-digital-on-unplanned-downtime-study.pdf, 2016.

Advanced analytics and the collection, management and visualization of data are changing the industry by building powerful predictive solutions for reducing unplanned downtime and its related costs. In this context, artificial intelligence and machine learning models are gaining more traction in the oil and gas industry.

2.3 Artificial intelligence and machine learning

In computer science, artificial intelligence (AI) is the ability of machines to do tasks that are normally attributed to human intelligence, such as pattern recognition, decision making, learning and problem solving (Russel & Norvig, 2016).

One of the main objectives of AI research is to solve complex problems by involving the ability of machines to gain knowledge from experience. These problems have been addressed by what is called machine learning (ML), which is considered “the most successful aspect of AI” (Dunjko & Briegel, 2018).

Machine learning gives computers the ability to “learn” without receiving explicit programming instructions. Through different methods and algorithms, computers can imitate the human learning process, thus identifying patterns in the data, making decisions or predicting behaviors (Aghabozorgi & Reza Khayyambashi, 2018).

When working with complex datasets with multiple features, a traditional programming model will require many rules to complete tasks such as pattern recognition, summarization and giving recommendations. For example, it is difficult to create the code of a program that allows a computer to detect spam e-mails, detect frauds in credit card transactions or classify images based on their visual characteristics. A traditional programming model could address the challenges described in these examples; however, machine learning algorithms provide simpler, more generalizable, and accurate solutions.

Before describing the machine learning model we used for this research, it is important to make the distinction between *unsupervised* and *supervised learning*.

In unsupervised learning, the machine uses data that is neither labeled nor classified. The machine groups the information and identifies similarities or patterns without any sort of previous guidance. Unsupervised learning uses clustering or association algorithms to group data with similar characteristics. These algorithms include k-means clustering and principal component analysis (PCA) (Shalev-Shwartz & Shai, 2014). Supervised learning consists of training a model with an existing dataset that already contains the labels for the pattern the model is aiming to identify. Then the model is given a new set of previously

unseen data, and with the use of supervised learning algorithms, it predicts the correct output (Barber, 2012). Supervised learning algorithms can be divided in two categories (Singh, Thakur, & Sharma, 2016):

- Classification: separates or classifies the data into discrete categories; these include support vector machine (SVM), k-nearest neighbors and naïve Bayes algorithm
- Regression: fits the data to a model and is mostly used for continuous values; these include linear regression, polynomial regression, and logistic regression

Most recommender systems fall in the supervised learning category. They use past purchase data (labels) to make product and services recommendations. Predicting whether a customer would purchase a given product is a binary classification problem since the predicted variable has only two possible outcomes.

2.4 Recommender systems and collaborative filtering

Recommender systems suggest the most valuable products or services to customers based on their previous purchase history or preferences (Adomavicius & Tuzhilin, 2005). These systems do so by applying discovery techniques to make a personalized recommendation. These techniques may involve deciding what products to buy, which movies to watch, or what websites to visit. Recommender systems have had a lot of success in the e-commerce industry, with Amazon being a prime example. Amazon uses item-to-item collaborative filtering that instead of matching a customer to other similar customers, matches a user's purchased and rated items to similar items and recommends them (Linden, Smith, York, 2003). Krawiec (2016) states that part of the company's sales increase from $9.9 billion to $12.83 billion (29% increase) from its second fiscal quarter in 2011 to a year later in 2012, is largely due to there being recommendations for additional products at nearly every step of the purchasing process. E-commerce companies often succeed in implementing recommender systems because they have massive amounts of customer and purchase data that can be used to build these models.

Recommender systems are a powerful technology that can extract additional value for a business from its user databases, creating win-win situations for customers and businesses (Sarwar, Karypis, Konstan, & Riedl, 2001).

Adomavicius & Tuzhilin (2005) defined three main categories for recommender systems based on the approach used to make recommendations:

- Content-based: Recommendations are made based on similarities between items and content, without using information related to the users. These are used when there is known data about the items (name, location, description, and other features), but not on the user
- Collaborative filtering: Recommendations are made based on similarities between users' profiles. The main idea behind them is that similar users will like similar items
- Hybrid recommendations: Combines content-based and collaborative filtering to make recommendations

Cremonesi, Koren, & Turrin (2010) suggest that collaborative filtering algorithms produce high quality recommendations with better accuracy than baseline algorithms.

User-based and item-based recommender systems are the two most popular techniques in collaborative filtering (Park, Park, Jung, & Lee, 2015). User-based models predict recommendations based on similarities between users' preferences for items, while item-based models predict user preferences of items based on the preference level of similar items for the same user. Both techniques rely on the use of the k-nearest neighbors algorithm.

2.5 K-nearest neighbors

The recommender systems that are achieving the most widespread success are those based on the k-nearest neighbors algorithm (k-NN). The k-NN algorithm computes similarities among neighbors and then

makes recommendations based on those similar neighbors. These algorithms are advantageous because they can rapidly incorporate new information; however, the search for neighbors can be a challenge when dealing with large or sparse datasets, since it requires a great deal of computational time (Sarwar et al., 2001). As an example of how user-based collaborative filtering works, see Figure 3 below. If Customer 1 and Customer 2 are identified as similar customers, they could be used to suggest new products or services to one another. If Customer 1 purchases Services A, B, and C first, and then Customer 2 purchases only Services B and C, a recommender system would suggest that Customer 2 might also like to purchase Service A.

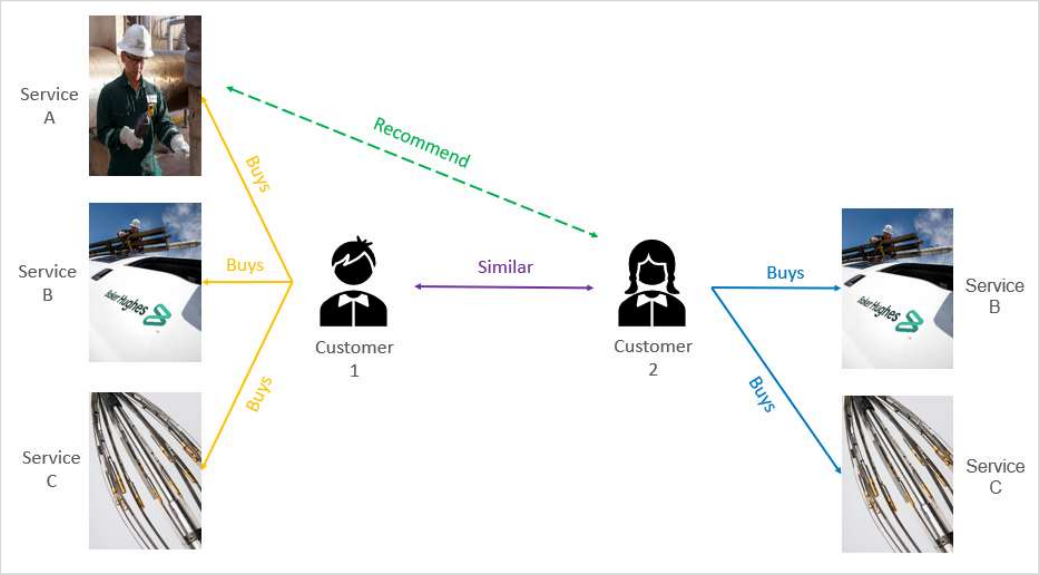

Figure 3. User-based collaborative filtering visual example

In the k-nearest neighbors algorithm, the similarity among users or customers is measured as a distance function. The most frequently used distance measures are the Euclidean distance and cosine similarity, but other methods like the Mahalanobis and city block distance are also found in the literature. Chomboon, Chujai, Teerarassamee, Kerdprasop, & Kerdprasop (2015) conducted an experimental comparison between different distance measures that use the k-nearest neighbors algorithm. Their results indicate that Euclidean distance is the distance measure that yields the best results.

2.5.1 Euclidean distance

The Euclidean distance is the straight-line distance between two points in a Euclidean space (Dangeti, 2017). Each point can be represented as a vector in an *n*-dimensional space. Given two vectors *p* and *q* with *n* dimensions each, the general formula to calculate the Euclidean distance between them is:

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{N} (q_i - p_i)^2}$$

2.5.2 Cosine similarity

Cosine similarity is a distance measure equal to the cosine of the angle between two *n*-dimensional vectors (Dangeti, 2017). The smaller the angle, the closer the vectors are to one another.

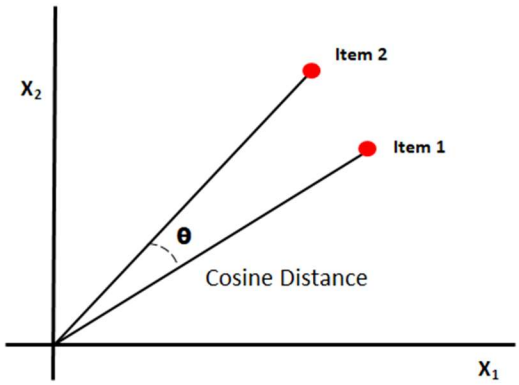

Figure 4. Cosine similarity illustration. Source: Dangeti, P. (2017). Statistics for machine learning. Packt Publishing Ltd.

Figure 4 represents how the similarity between users or items is measured by the cosine of the θ angle instead of the straight line between the two points. The general equation to calculate the cosine similarity between two *n*-dimensional vectors *p* and *q* is:

$$\cos(\theta) = \frac{p.q}{\|p\|\|q\|} = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} {p_i}^2}\sqrt{\sum_{i=1}^{n} {q_i}^2}}$$

Collaborative filtering user-based models that use the k-nearest neighbors algorithm assume that neighbors with smaller distances will likely have similar purchase behaviors, and model predictions will be a function of the distance measure to the closest neighbors. As with any machine learning predictive model, the accuracy of these predictions needs to be validated. The model must be accurate enough to make predictions not only for the data used to build the model, but also for new customers fed into the system.

2.6 Machine learning validation metrics for recommender systems

Recommender systems' predictions are often made through binary classifiers. This means the model predicts whether a customer would purchase (a positive prediction) a product or not (a negative prediction). This classification produces two types of correct predictions: true positives (TP) and true negatives (TN); and two possible incorrect labels: false positives (FP) and false negatives (FN). The goal of any classification model is to obtain the highest number of true values and the lowest number of false values. These four possible outcomes are typically represented in a confusion matrix (Saito & Rehmsmeier, 2015).

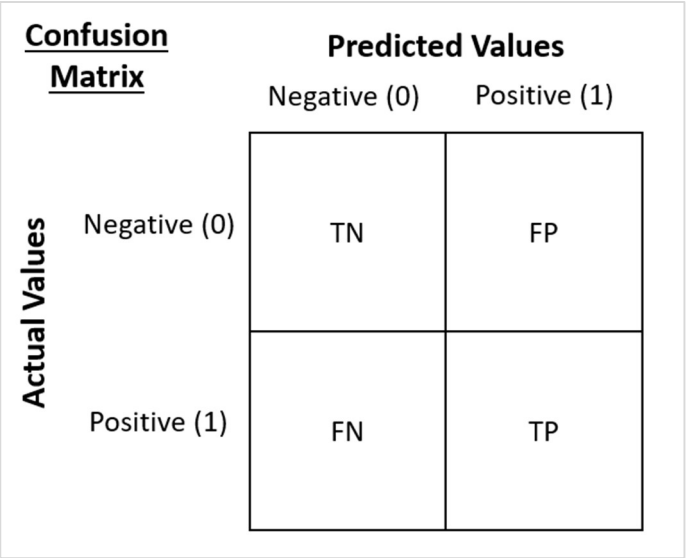

Figure 5. Schematic representation of a confusion matrix

Figure 5 shows a schematic representation of a confusion matrix, true negatives (TN in top left) occur when the true label was a negative value (0), and the model predicted it as such. True positives (TP in bottom right) occur when both the prediction and the actual value correspond to the positive class (1). False positives (FP in top right) occur when the predicted value is the positive class, but the actual value was negative (0). False Negatives (FN in left bottom) occur when the model predicts the negative class, but the actual value was positive.

According to Saito & Rehmsmeier (2015), the most widely used metrics for measuring the performance of a classification model are accuracy (ACC), error rate (ERR), sensitivity (SN) and specificity (SP). Figure 6 shows the most popular validation metrics and their formulas, derived from the values in the confusion matrix. The authors have concluded that these basic metrics are well suited for datasets where the class-labels are balanced, which means the actual labels have roughly the same amount of positive (1) and negative (0) values. Nevertheless, for highly imbalanced datasets, these basic metrics fail to evaluate the real performance of the model.

Measure	Formula
ACC	(TP + TN) / (TP + TN + FN + FP)
ERR	(FP + FN) / (TP + TN + FN + FP)
SN, TPR, REC	TP / (TP + FN)
SP	TN / (TN + FP)
FPR	FP / (TN + FP)
PREC, PPV	TP / (TP + FP)
MCC	$(TP * TN—FP * FN) / ((TP + FP)(TP + FN)(TN + FP)(TN + FN))^{1/2}$
$F_{0.5}$	1.5 * PREC * REC / (0.25 * PREC + REC)
F_1	2 * PREC * REC / (PREC + REC)
F_2	5 * PREC * REC / (4 * PREC + REC)

Figure 6. Basic evaluation matrix derived from the confusion matrix. Source: Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PloS one, 10(3).

For imbalanced datasets, measures like recall (REC) and precision (PREC) (see Figure 6) focus on measuring how accurate the model is at predicting the positive class (1), with much more importance than the

negative class (0). Precision is the fraction of true positive (TP) values among all positive predictions (TP+FP) and recall is the fraction of true positive (TP) values among all positive labels in the real data (TP+FN). Recall indicates how many of the actual positive values are correctly predicted by the model, while precision indicates how many of the predicted positive values are truly positive.

F-1 and F-2 scores are the harmonic mean (the reciprocal of the arithmetic mean of the reciprocals) between recall and precision, which provides a better measure of classification models with imbalanced data.

The precision-recall curve (PRC) shows the tradeoff between precision and recall and provides a model-wide evaluation instead of the F-score for a single threshold. The use of this curve provides the best way to compare different models with highly imbalanced binary class-labels (Saito & Rehmsmeier, 2015).

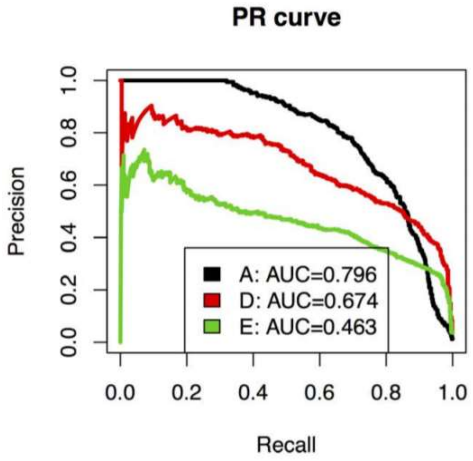

Figure 7. Example of a Precision-Recall Curve. Source: Keilwagen, J., Grosse, I., & Grau, J. (2014). Area under precision-recall curves for weighted and unweighted data. PloS one, 9(3), e92209.

Figure 7 shows the plot for a typical precision-recall curve (PRC). Recall is represented on the x-axis versus precision on the y-axis. The higher the area under the curve (AUC), the better the model is at predicting the positive class. An accurate model is one that can maintain precision as high as possible as recall also increases.

F-scores and the area under the curve of the PRC are particularly important for user-based collaborative filtering recommender systems, where normally a single customer may only have purchased a few items in the past, so datasets tend to contain a lot of “zero” (not purchased) values and only a few “one” (purchased) values.

2.7 Literature Review conclusion

Prices have fallen in the oil and gas industry in the last five years and fluctuations in the market have forced oil and gas companies to invest in new technologies to reduce costs and maintain profits. These new technologies are being driven by advanced analytics and data-driven decisions to reduce unplanned downtime and its related costs. With vast amounts of data, oil and gas companies can benefit from the use of machine learning to anticipate customers’ needs and make recommendations for products and services to avoid future downtime in their facilities. User-based collaborative filtering can be a powerful method to build such recommender systems. Systems using the k-nearest neighbors algorithm can be easy to implement, and they often provide accurate purchase predictions.

Datasets used for recommender system often have sparse matrices of combinations between customers and products. The use of F-scores and the precision-recall curve are the best metrics to validate the performance of systems with imbalanced datasets.

These conclusions serve as a good basis to justify the methodology of our project. We built a user-based collaborative filtering recommender system using the k-nearest neighbors algorithm and validated it with the F2-score and precision-recall AUC.

3. METHODOLOGY

The objective of this project is to develop a model using machine learning techniques to reduce unplanned downtime of customers of the sponsoring company. This section describes the steps followed to build a user-based collaborative filtering model that makes product recommendations based on customers' similarity. Our experimental design followed the typical structure of a supervised learning problem. This structure is described in Figure 8.

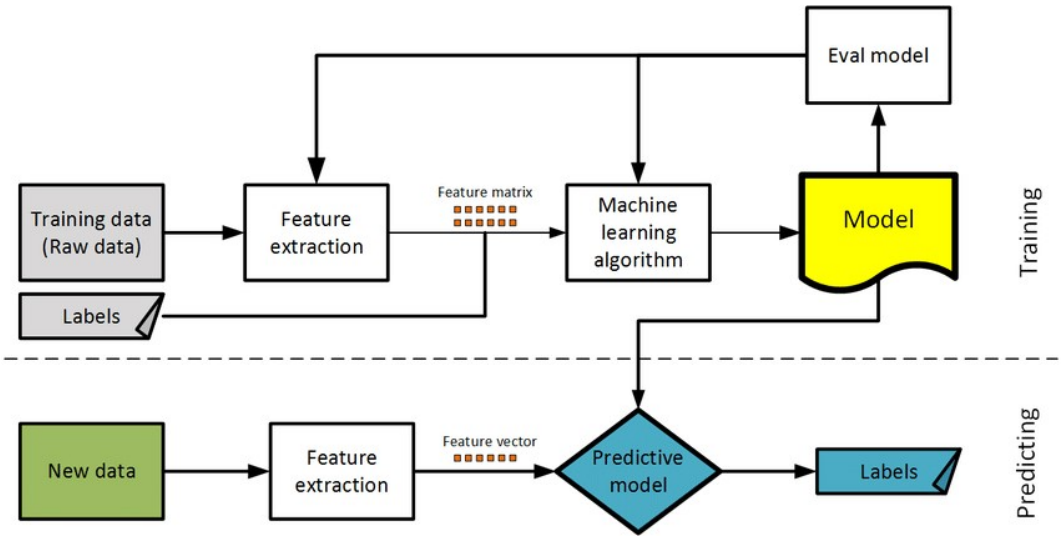

Figure 8. Flowchart of a supervised machine learning model. source: Nguyen, et al. (2017). Joint network coding and machine learning for error-prone wireless broadcast. In 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 1-7). IEEE.

First, we analyzed the sales data provided by the sponsoring company, then we preprocessed it and separated it into "training" (raw data with removed labels) and "testing" (new data) datasets. Next, we built the customer-product purchases matrix (feature matrix) and determined customer similarities using k-nearest neighbors (k-NN) as a machine learning algorithm fitted to the training data. Customer similarities were used to build a model to predict whether a customer would purchase (or not) a given product. With this function, we computed the purchase predictions for all possible combinations of products and customers, then evaluated the model's accuracy by comparing the predictions with the

actual purchase (labels) history. Then we proceeded to test the model with new data and compare the testing accuracy metrics with those obtained from the training of the model (eval model). Finally, we used the resulting product recommendations to estimate unplanned downtime reductions and cost savings per customer. A detailed explanation of each of these steps is provided below.[3]

3.1 Data overview and pre-processing

The data used to build the model consisted of sales records from the last three years for 180 customers and 74 different products of the sponsoring company. We anonymized the data in compliance with the rules of the intellectual property of the sponsoring company. Customer and product features were removed, as well as any financial information related to the sales in the dataset.

	customerId	customer	productId	product	purchased
0	0	Customer_0	0	Product0	1
1	1	Customer_1	0	Product0	0
2	2	Customer_2	0	Product0	0
3	3	Customer_3	0	Product0	0
4	4	Customer_4	0	Product0	0

Figure 9. First five rows of the original sales dataset

Figure 9 shows the first five rows of the original dataset. The entire dataset contains 13,320 rows and five columns describing the customer and product names and ids. Each row represents a combination of one of the 180 customers with one of the 74 products. The "purchased" column indicates whether the product was acquired by the customer. A "1" value indicates the customer purchased the product. Alternatively, a "0" value indicates that customer has not acquired that product in the last three years.[4]

[3] Note: The model described in this methodology section was built using a Python 3.7.4 programming environment.

[4] This column does not correspond to the number of units sold, it merely indicates whether the product was purchased or not.

3.1.1 Train/Test split of the dataset

Every supervised learning model requires the data to be proportionally separated into two datasets, normally identified as *training* and *testing* datasets. The training dataset is used to build and train the model to obtain initial predictions. The testing dataset is used to measure the model's accuracy against new unseen data. By splitting the data, this avoids creating an overfitted model. Typically, the training dataset represents two-thirds of the original dataset and the testing dataset contains the remaining one-third of the data. This distribution tends to produce the best results in high-dimensional classifiers (Dobbin & Simon, 2011).

Our initial model used an 80–20% split. This means only 80% of the customers were used to build and train the model. The split was done randomly to avoid the effect of any possible bias in the way customers were originally displayed in the data.[5] When validating the model, we used the testing dataset size as a hyperparameter in order to find the split that would return the best model validation metrics. Hyperparameters are settings that can be tuned to control the before of a machine learning model.

```
1 df_train.head()
```

	customerId	customer	productId	product	purchased
2142	162	Customer_162	11	Product11	0
1167	87	Customer_87	6	Product6	0
4214	74	Customer_74	23	Product23	0
6073	133	Customer_133	33	Product33	0
3213	153	Customer_153	17	Product17	0

```
1 df_test.head()
```

	customerId	customer	productId	product	purchased
4259	119	Customer_119	23	Product23	0
5195	155	Customer_155	28	Product28	0
1813	13	Customer_13	10	Product10	0
10693	73	Customer_73	59	Product59	0
6162	42	Customer_42	34	Product34	0

```
shape of the training set:  (10656, 5)
shape of the testing set:  (2664, 5)
```

Figure 10. View of the first five rows of the training and testing datasets

[5] For model evaluation purposes, a random seed was used in order to always obtain the same resulting training and testing datasets.

Figure 10 shows a view of the first five rows and the shape of each dataset. The training dataset contains 10,656 out of the 13,320 customer purchase records, whereas the testing dataset contains the remaining 2,664 records.

3.1.2 Creating the user-item matrix

In order to build a traditional user-based collaborative filtering model we had to transform the data into a user-item matrix, commonly referred to as the "R-matrix." The structure of this matrix is shown in Figure 11. Customers are represented in rows and products are represented by columns. The values of the matrix represent the purchase value.

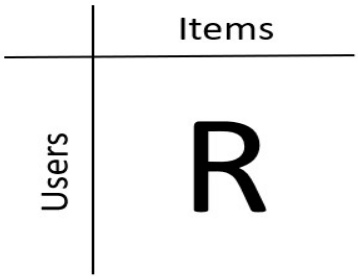

Figure 11. Schematic representation of the user-item matrix

Figure 12 shows the R-matrix for the dataset used in our model. The matrix was built from the training data. Each row is a customer ID and each column represents a product ID. The values in the matrix show the purchase history for any given combination of product and customer. Since the train/test split removed several values of the matrix, we replaced them using a random generator following the distribution of zero and one values in the actual dataset. The resulting matrix has the shape 180x74, which corresponds to the correct number of customers and products in the model.

```
#display the train customer-product matrix
train_matrix
```

	0	1	2	3	4	5	6	7	8	9	...	64	65	66	67	68	69	70	71	72	73
0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	NaN	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	NaN	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	NaN	0.0	0.0	NaN	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	NaN	...	0.0	0.0	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
175	0.0	NaN	NaN	0.0	0.0	NaN	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	NaN	NaN	0.0	0.0	0.0	0.0
176	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	NaN	0.0	0.0	1.0
177	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	NaN	0.0	...	0.0	0.0	0.0	0.0	NaN	0.0	0.0	0.0	0.0	0.0
178	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	NaN	0.0	...	0.0	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
179	0.0	0.0	0.0	1.0	0.0	0.0	NaN	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	NaN

180 rows × 74 columns

Figure 12. Customer-product matrix built from the training dataset. NaN values were replaced with random values following the zero-one proportion in the original sales data.

3.2 Finding the k-nearest neighbors for each customer

Product recommendations are dictated by similar purchase behavior among customers. To illustrate this idea, consider an example where we have 3 customer purchase records for 5 different products (Figure 13).

	Product 1	Product 2	Product 3	Product 4	Product 5
Customer A	1	1	1	0	0
Customer B	1	1	?	0	0
Customer C	0	1	0	1	1

Figure 13. Customer similarity example

We would like to determine whether it is a good idea to recommend Product 3 to Customer B. We see that both Customer A and Customer C share some common product purchases with Customer B. However, Customer A appears to be much more “similar” to Customer B, in the sense that they both purchased Products 1 and 2, and alternatively did not purchased Products 4 and 5. It is very likely that Customer B will require Product 3, and would likely be a good recommendation.

This notion of similarity between users can be quantified using a measure of "distance" between customers. For our model, we considered two different distance measurements mentioned in our Literature Review (Section 2.5): Euclidean distance and cosine similarity.

Using both distance measures as hyperparameters of our model, we built a neighbors dictionary from the training data to serve as an input in our prediction function.

3.3 Normalizing distance measures

Euclidean distance and cosine similarity are measured on different scales. To use distance as a similarity score in our prediction function, we needed to measure the distances on a scale from zero to one. While cosine similarity is already measured from a zero to one scale, Euclidean distances required normalization.

To normalize the Euclidean distances, we used the min-max scaler method, which consists of subtracting the minimum calculated distance from each neighbor's distance and dividing the result by the range between the minimum and maximum distances.

$$Normalized\ distance_i = \frac{d_i - \min(d)}{Max(d) - Min(d)}$$

Both distance measures were considered separately to make purchase predictions in our user-based collaborative filtering model. Distance measures were calculated for every pair of customers in the training dataset.

```
1 neighbors_dict_train[7]
```

```
SortedList([(1.0, 31), (1.0, 66), (1.4142135623730951, 16), (1.4142135623730951, 45)])
```

```
1 neighbors_normalized_train[7]
```

```
SortedList([(0.35355339059327373, 31), (0.35355339059327373, 66), (0.5, 16), (0.5, 45)])
```

Figure 14. Normalized Euclidean distance and nearest neighbors for Customer 7

Figure 14 shows the 4-nearest neighbors to Customer 7. The first number in each pair is the Euclidean distance and the second number is the customer ID of the neighbor. Customers 31 and 66 are the most similar to Customer 7, and they are equally distanced. Customers 16 and 45 are less similar to Customer 7, but they rank as the third and fourth closest neighbors. The first list shows the distance measures before being normalized. The list in the bottom shows the distances after being normalized.

3.4 Calculating purchase predictions

The central objective of our recommender system is to predict whether a certain product would be purchased by a given customer. We defined purchase predictions as a function of the similarity measures between each user and its k-nearest neighbors, and whether these neighbors acquired a given product or not. With this in mind, we created the following prediction function:

$$\hat{p}_{i,j} = \begin{cases} 1, & \forall\, S_{i,j} > roundup\ threshold \\ 0, & otherwise \end{cases}$$

$$S_{i,j} = \frac{\sum_{i' \in \varphi_k}(1 - d_{ii'}).\, p_{i'j}}{\sum_{i' \in \varphi_k}(1 - d_{ii'})}$$

where:

i : customer for which the prediction is made

j : product for which the prediction is made

$\hat{p}_{i,j}$: predicted purchase of product j by customer i

$S_{i,j}$: predicted purchase score of product j for customer i

k: the number of k-nearest neighbors to customer i to be considered in the prediction

φ_k : set of k-nearest neighbors of customer i. The size of the set is equal to k

$d_{ii'}$: the normalized distance between customer i and neighbor i'

$p_{i'j}$: purchase value of product j by neighbor i'

This function uses customer similarities to predict whether a given customer i would buy product j. This prediction is notated as $\hat{p}_{i,j}$ and can be either a value of "1" for a predicted purchase, or "0" to indicate the customer would not likely acquire this product.

To arrive at these predictions, we first calculate the "purchase score" ($S_{i,j}$) for the same customer i and product j. This score uses the distance between each of the k-nearest neighbors of customer i (neighbors are identified as i') as a "weight" to the prediction. This means that if the closest neighbor i' purchased product j (thus making $p_{i'j}$=1), then it will have a greater influence on the purchase decision than those neighbors that are less similar to customer i.

After calculating the purchase score, we compare the result to the *roundup threshold* value. The roundup threshold is defined as a limit value that will force the prediction to take a binary value of zero or one. If the purchase score is greater than the roundup threshold, then the prediction is rounded to "1"; alternatively, if the score is smaller than the roundup threshold, then the prediction is rounded to "0".

Consider an example using the actual data where we want to predict whether Customer 4 would purchase Product 3, meaning we want to arrive at the predicted value for $\hat{p}_{4,3}$.

1) First, we set the number of k-nearest neighbors to consider in the prediction. For example, $k = 2$
2) Find the 2-nearest neighbors to Customer 4 and get their normalized distances

```
1 neighbors_normalized_train[4]
```

```
SortedList([(0.1339745962155613, 3), (0.42264973081037416, 12)])
```

Figure 15. Nearest 2-neighbors to Customer 4 and their normalized distances

Using cosine similarity as a distance measure we determine Customer 3 is the closest neighbor to Customer 4, followed by Customer 12 as the second neighbor. Figure 15 shows the resulting distances between each neighbor and Customer 4 (notated in the score function as $d_{ii'}$)

$$d_{4;\,3} = 0.13397, \quad d_{4;12} = 0.42265$$

3) Find whether Neighbors 3 and 12 purchased (or not) the product we are making the prediction for (these are notated in the score function as $p_{i'j}$).

   In the actual dataset, Customer 3 did buy product 4, and Customer 12 did not buy it. Therefore:

$$p_{3;\,3} = 1, \quad p_{12;3} = 0$$

4) Calculate the "purchase score" of Product 3 for Customer 4 ($S_{4,3}$). For this we use the score formula indicated above, considering Customers 3 and 12 as the i' neighbors:

$$S_{i,j} = \frac{\sum_{i' \in \varphi_k}(1-d_{ii'}).p_{i'j}}{\sum_{i' \in \varphi_k}(1-d_{ii'})} \qquad S_{4;3} = \frac{(1-d_{4;\,3})*p_{3;\,3} + (1-d_{4;12})*p_{12;3}}{(1-d_{4;\,3}) + (1-d_{4;12})}$$

   This results in:

$$S_{4;3} = \frac{(1-0.13397)*1 \;+\; (1-0.42265)*0}{(1-0.13397) +\; (1-0.42265)} = \frac{0.87*1+0.58*0}{0.87+0.58} = \mathbf{0.6}$$

   Since Customer 3 has a smaller distance to Customer 4, its purchase behavior has a larger weight (0.87) in the final score. The weight of the purchase behavior of Customer 12 (0.58) still impacts the purchase score, but with less influence.

5) Compare the purchase score with the roundup threshold to make the final prediction. Now that we have the purchase score $S_{4;3} = 0.6$, we must decide whether this score is large enough to make a purchase prediction. If the roundup threshold is set to 0.5 (i.e.), since the score is higher than the roundup threshold, the prediction would be equal to "1" and we would predict the purchase of Product 3 by Customer 4:

$$\hat{p}_{i,j} = 1 \quad : \forall\, S_{i,j} > roundup\ threshold$$

$$\hat{p}_{i,j} = 0 \quad : otherwise$$

$$S_{4;3} = \ 0.6 > 0.5 \ \rightarrow \ \hat{p}_{4,3} = 1$$

Purchase predictions $\hat{p}_{i,j}$ were calculated for every possible combination of customers and products, both in the training and testing datasets, resulting in a total of 13,320 predictions.

3.5 Validating the model and finding the best values for hyperparameters

The next step of a supervised learning problem is to compare the predicted value($\hat{p}_{i,j}$) against the actual purchase value ($p_{i,j}$). As we concluded in Section 2.6 of the Literature Review, the F-2 score and area under the curve of the precision-recall curve are the best measures to validate the performance of our model.

So far, we have described the model using arbitrary values for different hyperparameters. These hyperparameters of our model are:

- Split: size of the test dataset (possible values evaluated: 0.1, 0.2, 0.3 and 0.4)
- K: the number of k-neighbors to each customer considered in the prediction function (possible values evaluated: 1, 2,3 4, 5, 10, 15, 20, 25)
- Distance measure: whether to use Euclidean distance or cosine similarity as the distance measure (0 for Euclidian distance and 1 for cosine similarity)
- Roundup threshold: the limit value to compare to the prediction score to obtain the classified label (possible values: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9)

We ran the model a total of 648 times for every possible combination of these hyperparameters to find the combination that yields the highest validation metrics.

3.6 Estimating unplanned downtime savings and additional revenues for the sponsoring company

The main objective of our research is to arrive at estimates of downtime savings and potential additional revenues for the sponsoring company as a result of the implementation of the recommender system we developed. To arrive at such estimates, we used the following numbers provided by the sponsoring company and our literature review:

1. The average downtime per service performed or product installed by the sponsoring company is 36 hours
2. The average product/service sales price is $15,000
3. The total estimated cost of unplanned downtime per day is $1.41 million (Baker Hughes Company, 2016)

We considered a recommendation to be "accurate" when our model predicted a purchase for a product or service for a given customer, and the actual historical sales data used to build the model also indicated the customer acquired that product or service. Accurate recommendations are equal to the true positive (TP) values of the model results (confusion matrix).

Since we only considered recommendations from the testing dataset, we extrapolated the test results to the original sales data to estimate the number of accurate recommendations we could achieve. To make this extrapolation we used the true positive rate (Recall) as the link between the testing dataset and the entire sales data.

$$Recall = \frac{TP}{TP + FN}$$

The testing dataset recall represents how many actual purchases in the sales data that were predicted correctly by our model. The estimated number of accurate recommendations from our model can then be calculated as follows:

$$Accurate\ recommendations = Testing\ dataset\ recall * Total\ purchases\ in\ the\ original\ data$$

Using the total estimated number of accurate recommendations, we can estimate the savings that would have been obtained if our model had been implemented before the historical sales occurred. Based on the number of accurate recommendations, downtime savings and additional revenues were calculated as follows:

Estimating unplanned downtime reduction:

Annual downtime reduction was obtained by multiplying the number of accurate recommendations by the average downtime required to execute a service or install a product by the sponsoring company:

$$Annual\ downtime\ reduction\ = \left(\frac{Average\ downtime\ per\ service}{24\ h/day}\right) * Accurate\ recommendations$$

$$= \left(\frac{36\ hours}{24\frac{h}{day}}\right) * Accurate\ recommendations \quad (in\ days\ of\ downtime/year\ )$$

We obtained the reduction per customer by dividing the total downtime reduction by the number of customers for which an accurate recommendation was made:

$$Annual\ downtime\ reduction\ per\ customer$$

$$= \frac{Annual\ downtime\ reduction}{Number\ of\ recommended\ customers} \quad (in\ days\ of\ downtime/customer - year)$$

Estimating unplanned downtime cost savings:

Cost savings were estimated using the same logic as above, this time multiplying the number of accurate recommendations by the average downtime cost per day of $1.41 million:

$$Total\ annual\ cost\ savings$$

$$= Annual\ downtime\ reduction\ per\ customer * Average\ downtime\ cost\ per\ day$$

$$= Annual\ downtime\ reduction\ per\ customer * \$1.41\ MM/day \quad (in\ \$MM\ per\ year)$$

$$Annual\ cost\ savings\ per\ customer$$

$$= \frac{Annual\ cost\ savings}{Number\ of\ recommended\ customers} \quad (\ in\ \$MM/customer - year)$$

Estimating additional revenues for the sponsoring company:

Additional revenues for the sponsoring company were estimated by multiplying the number of accurate recommendations by the average service price of the products and services delivered by the sponsoring company:

$$Annual\ additional\ revenues$$

$$= Average\ service\ price * Number\ of\ accurate\ recommendations\ (in\ \$\ per\ year)$$

$$Annual\ additional\ revenues\ per\ serviced\ customer$$

$$= \frac{Total\ annual\ additional\ revenues}{Number\ of\ recommended\ customers} \quad (in\ \$/customer - year)$$

4. RESULTS AND DISCUSSION

This section includes the results and analysis obtained from the machine learning recommender system we developed. We present our initial findings of sales insights, best model hyperparameters, and validation of the model's performance. Finally, we present the estimated downtime savings for customers and the potential additional revenues for the sponsoring company.

4.1 Initial data discovery

To get a better understanding of the sales dataset, the top 10 purchased products were plotted as seen in the top-left graph of Figure 16.

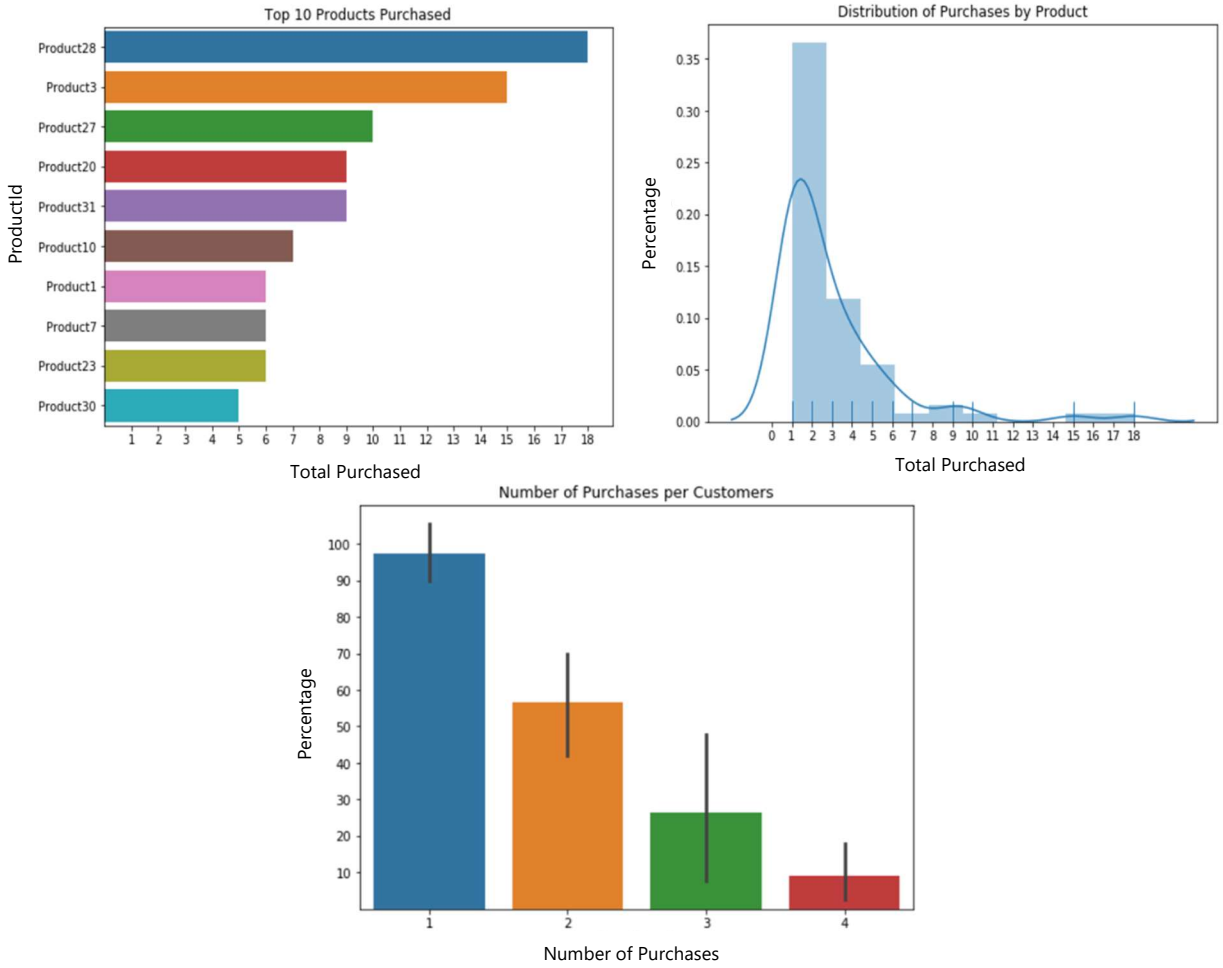

Figure 16. Sales data insights for customer and products

The most purchased product was Product 28, with a total of 18 purchases. Although the top 10 products had total purchases of 5 or greater, most of the products were only purchased fewer than 4 times total by all customers (top-right of Figure 16). To find out how many purchases each customer made, we plotted the distribution of the number of purchases per customer. The bottom-middle graph of Figure 16 shows that customers only bought a maximum of 4 products and most of the time bought 1 or 2 products.

4.2 Initial model performance results

Using initial arbitrary (but reasonable) hyperparameters we ran the model to make the 13,320 predictions for all the possible combinations of products and customers. The model's results are presented in Figure 17. Initial results show that the model performs well at making predictions in the training (in-sample) dataset but has low performance in the testing (out-of-sample) dataset. Precision and recall values from the initial run suggest the model is very good at predicting negative (zero) values but is not effective when predicting positive (one) values, leading to the low F2-score of 10.6% for the testing dataset.

```
validation metrics results
all values printed in the format: train, test

TN values: 11278 1247
TP values 162 3
FN values: 40 15
FP values: 508 67

precision values: 0.2418 0.0429
recall values: 0.802 0.1667

confusion matrix for training dataset
[[11278   508]
 [   40   162]]

confusion matrix for testing dataset
[[1247   67]
 [  15    3]]

f1-score values: 0.3716 0.0682
f2-score values: 0.548 0.1056
```

Figure 17. Initial model validation results. For test size = 0.1, distance measure = Euclidean distance, k = 5 and roundup threshold = 0.1.

Alternatively, the precision-recall curves in Figure 18 validate a high performance on the training data with an AUC = 78.9%; but the curve for the testing dataset is slightly is over the "No-skill" line with an AUC = 3.2%. The "No-skill" line represents a baseline model with no classification ability.

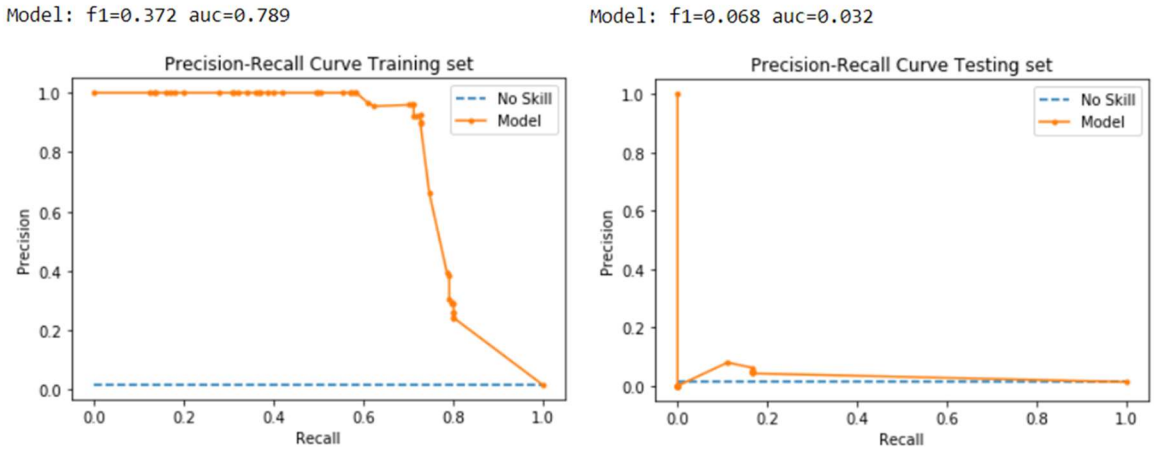

Figure 18. Precision-recall curves for the training and testing dataset

4.3 Best model hyperparameters

As indicated in the Methodology (Section 3.5), the model was executed 648 times to cross-validate all hyperparameters (testing dataset size, distance measures, number of k-nearest neighbors and roundup threshold). Figure 19 shows the combination of hyperparameters that yielded the highest F-2 score for the testing dataset. The results indicate that with a train/test split of 80–20%, using Euclidean distance, considering 10-nearest neighbors and a roundup threshold of 0.1 for making predictions, the result is an F2-score of 49.19% for the training dataset, and 13.23% for the out-of-sample testing dataset.

```
--------------------------------------
Best Parameters for the testing dataset:
test size                    0.2000
K-neighbors                 10.0000
distance measure             0.0000
roundup threshold            0.1000
f2-score Training dataset    0.4919
f2-score Testing dataset     0.1323
```

Figure 19. Results for the best combination of the model hyperparameters

To understand the model's sensitivity to the values of the hyperparameters, we used the results of the 648 possible models and plotted every hyperparameter individually against the F2-score values.

The top-left boxplot in Figure 20 suggests there is not enough statistical evidence to conclude that the F2-score is independent of the test size, at least for values from 0.1 to 0.4. Nevertheless, for test sizes of 0.1 and 0, the outliers provide higher F2-score results. The same occurs with the distance measure, with the Euclidean distance (notated as "0") providing higher scores for the outliers, but the boxplots still overlap.

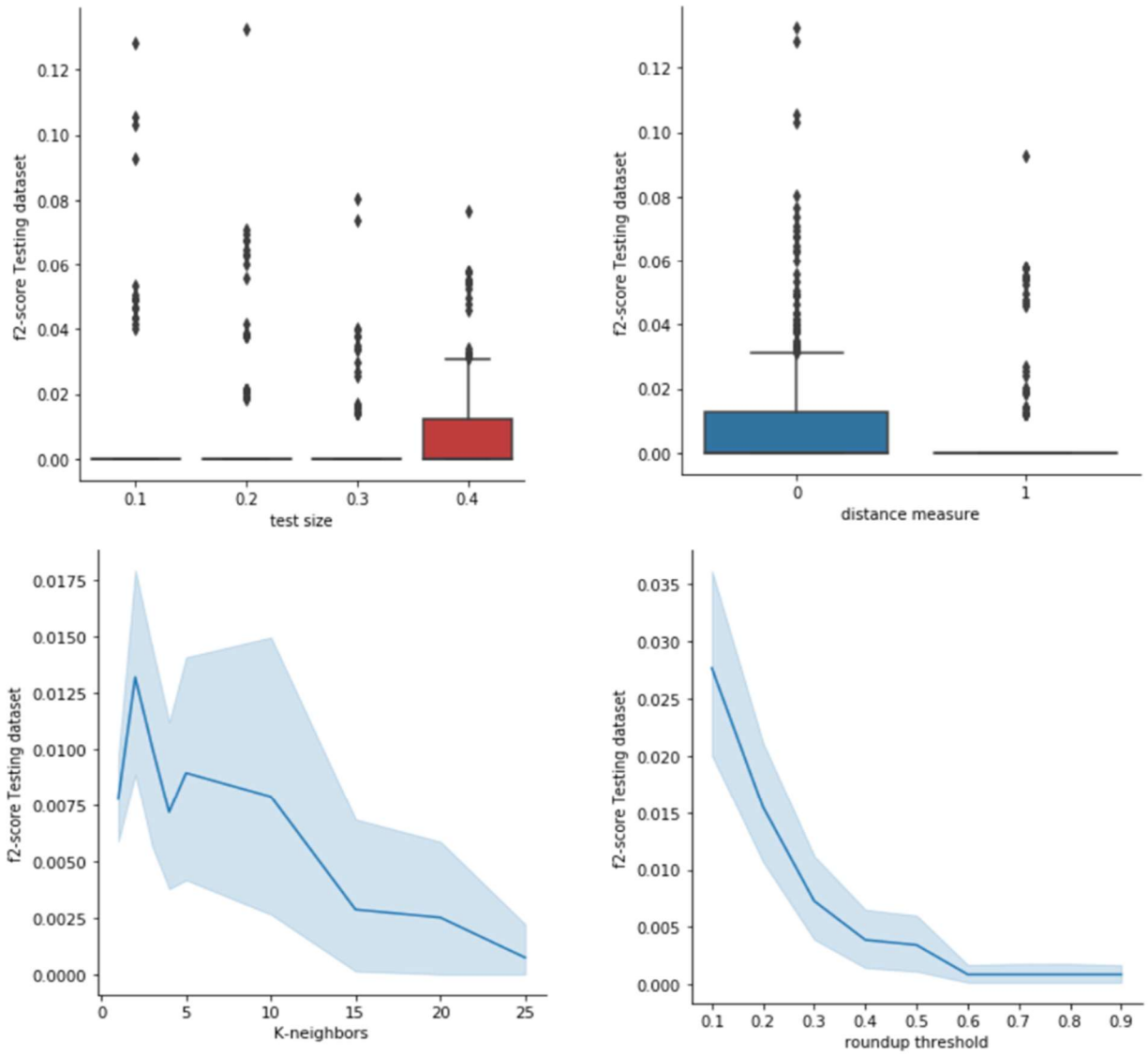

Figure 20. Plots for all possible values of the model hyperparameters against the F2-score of the test dataset

The bottom-left graph of Figure 20 indicates that the number of neighbors considered in the prediction formula has a clear influence on the model's performance. This intuitively makes sense because of the high sparsity in the sales data. For higher numbers of k, the model begins to incorporate neighbors that likely are not very similar[6].

As for the roundup threshold, the higher its value, the harder it is for the model to predict a positive value. This occurs again because of the high sparsity in the sales data. With high sparsity, negative (zero) values will have more weight in the prediction calculation.

4.4 Calculating the number of accurate recommendations

To calculate the total number of accurate recommendations our model made, the recall value from the testing dataset was applied to the total number of purchases from the initial dataset. To calculate recall, we take the true positive (TP) and false negative (FN) values from the confusion matrix of the testing dataset, as seen in Figure 21.

$$Recall = \frac{TP}{TP + FN} = \frac{10}{10 + 40} = \frac{10}{50} = \mathbf{20.0}\%$$

Using this recall value, we applied this to the total number of purchases (220) from the original dataset to estimate how our model would have performed.

$$Accurate\ Recommendations = 220 * 20.0\% = \mathbf{44}$$

[6] The shaded around the line plot of the bottom graphs indicate the error bands of the confidence intervals of the y-axis variable.

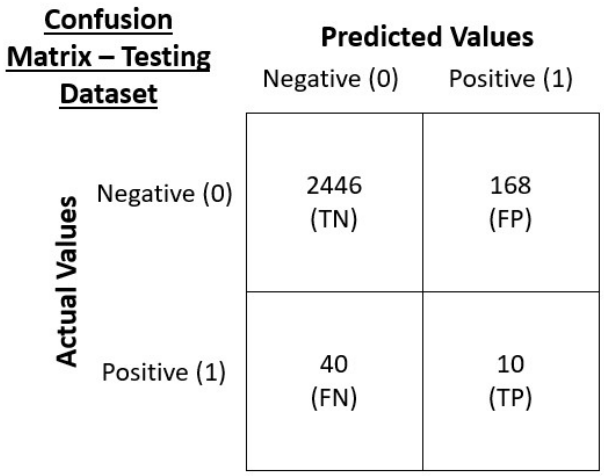

Figure 21. Confusion Matrix - testing dataset using the best model hyperparameters

The total amount of downtime saved and additional revenue for the sponsoring company can be calculated from the number of accurate recommendations.

4.5 Downtime reduction and additional revenues

The model accurately predicted 44 purchases for 38 different customers. Based on the estimates from the Methodology (Section 3.6), we calculated the annual downtime reduction and additional revenue for the sponsoring company.

Estimated unplanned downtime reduction:

$$Annual\ downtime\ reduction\ = \left(\frac{Average\ downtime\ per\ service}{24\ h/day}\right) * Total\ accurate\ recommendations$$

$$= \left(\frac{36\ hours}{24\frac{h}{day}}\right) * 44 = \mathbf{66}\ (days\ of\ downtime/year)$$

$$Annual\ downtime\ reduction\ per\ customer\ = \frac{Annual\ downtime\ reduction}{Number\ of\ recommended\ customers} = \frac{66\ days}{38\ customers}$$

$$= \mathbf{1.74}\ (days\ of\ downtime/customer - year)$$

The results show the recommender system can reduce unplanned downtime by 1.74 days per customer per year. With an average of 27 days of downtime for oil and gas organizations, this represents a 6.44% reduction in unplanned downtime.

Estimated unplanned downtime cost savings:

$$Total\ annual\ cost\ savings\ =\ Anual\ downtime\ reduction * Average\ downtime\ cost\ per\ day$$

$$= 66\ days * \$1.41\frac{MM}{day} = \$\mathbf{93.06}\ (\$MM\ per\ year)$$

$$Total\ annual\ cost\ savings\ per\ customer\ =\ \frac{Total\ annual\ cost\ savings}{Number\ of\ recommended\ customers} = \frac{\$93.06MM}{38\ Customer}$$

$$= \$\mathbf{2.45}\ (\ \$MM/customer - year)$$

Oil and gas organizations experience an average of $38 million in financial impacts from unplanned downtime every year. A $2.45 million cost reduction would be equivalent to 6.44% of savings per customer every year.

Estimated additional revenues for the sponsoring company:

$$Total\ annual\ additional\ revenues$$

$$= Average\ service\ price *\ Number\ of\ accurate\ recommendations\ (\$\ per\ year)$$

$$= \$15{,}000 * 44 = \$\mathbf{660}\boldsymbol{k}\ (per\ year)$$

$$Total\ annual\ additional\ revenues\ per\ serviced\ customer = \frac{Total\ annual\ additional\ revenues}{Number\ of\ recommended\ customers}$$

$$= \frac{\$660k}{38\ customers} = \$\mathbf{17}\boldsymbol{k}\ (in\ \$/customer - year)$$

With 44 accurate recommendations and an average sales prices of $15,000, the sponsoring company could obtain additional revenues of $660 thousand (or $17K per customer) by implementing the developed recommender system within this specific product line.

Figure 22 shows a summary of the downtime savings and additional revenue produced by our machine learning recommender system.

```
Total number of accurate recommendations: 44 recommendations

Estimated downtime reduction for all customers: 66.0 days/year
Estimated downtime reduction per customer: 1.74 days/customer-year

Estimated downtime costs savings for all customers: $93.06 MM/year
Estimated downtime costs savings per customer: $2.45 MM/customer-year

Estimated additonal revenue for Baker Hughes: $660.0 K/year
Estimated additonal revenue for Baker Hughes per customer: $17.37 K/customer-year
```

Figure 22. Results for number of recommendations, estimated downtime reduction and cost savings

4.6 Limitations of sparse data on the model's performance

As described in Section 3.1 of the Methodology, the dataset provided by the sponsoring company included the purchase history of 180 customers and 74 products. The dataset contained 220 purchases out of 13,320 possible purchases which equates to 1.65%. Having such a sparse customer-product matrix limited not only the process of finding the k-nearest neighbors, but also the number of accurate recommendations, resulting in an out-of-sample F2-score of 13.2%.

We tested our model with a denser sample dataset provided by the sponsoring company comprised of 2,596 customers and 24 products. This sample dataset contained 27,306 purchases out of 62,304 possible purchases, which equated to 43%. Using the best model hyperparameters presented in Section 4.3, the model yielded an F2-score of 86.7% for the testing dataset. The F2-score is 73.5% greater than the score obtained with the initial sparse data. In respect to the AUC, Figure 23 shows the denser sales data yielded

an AUC of 98.8% for the training dataset and 83.5% for the testing dataset. This is a 19.9% and 80.3% improvement compared to the sparse dataset AUC for the training and testing datasets, respectively. These results represent a significant improvement for the testing dataset validation metrics and provide strong support to the performance of our model with the use of larger and denser sales data.

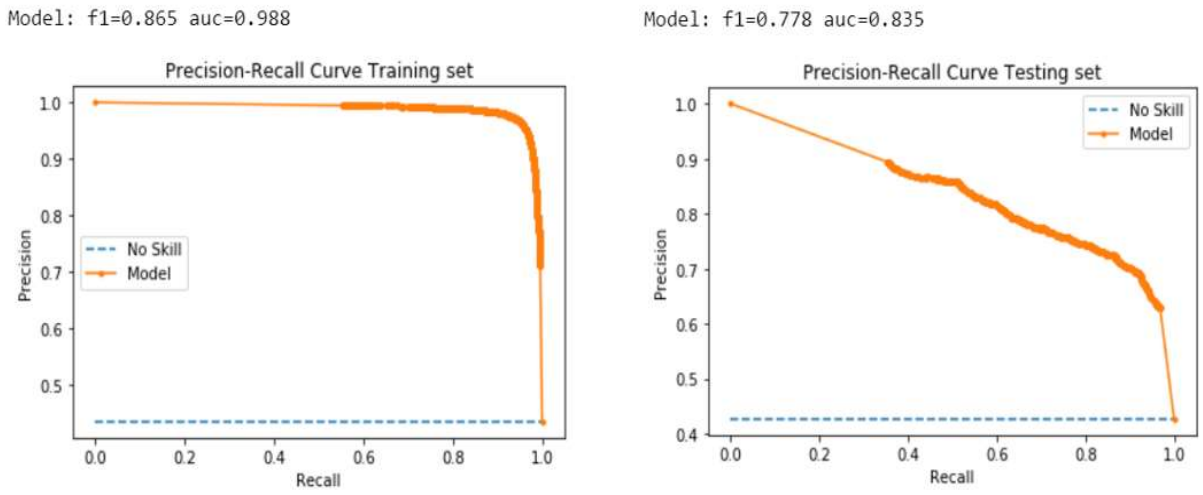

Figure 23. Precision-recall curves obtained from the denser sample sales dataset

4.7 Scalability

The machine learning recommender system we developed is a robust model that can easily be scaled across the sponsoring company. The only data required to train the model is historical sales data containing the customer and previous product purchases. However, as noted in the above Section 4.6, the model performs better with larger datasets. As the number of sales records increases, the model will take more time to run since it requires building large arrays and Python dictionaries. Computational complexity is particularly high for finding the nearest neighbors for all customers in the data. Additionally, historical sales data must be in a specific format to run the current model; therefore, the data or model may need to be modified as it is scaled to other product lines within the sponsoring company.

5. CONCLUSION & RECOMMENDATIONS

As the price of oil has fallen in recent years and companies face challenges reducing unplanned downtime of their critical equipment. With 27 days of unplanned downtime per year, companies in the oil and gas industry experience an average $38 million per year, representing 7.95% of their cost structure. To tackle this problem, companies are turning to a predictive maintenance approach based on advanced analytics and machine learning models that have proven to reduce maintenance costs and unexpected equipment failure.

This capstone project presents the benefits of a user-based collaborative filtering recommender system on reducing downtime and increasing revenues for Baker Hughes and other companies in the oil and gas industry. For our dataset, we discovered that the model yields the highest out-of-sample F2-score when using an 80–20% or 90–10% train/test split and with 10 or less k-nearest neighbors. There was not sufficient evidence to prove differences in the use Euclidian distance or cosine similarity when determining similar customers based off their previous purchase history.

Our results show that our model can achieve a 6.44% reduction of unplanned downtime and cost savings for the sponsoring company. Furthermore, we proved that these results can significantly be improved by incorporating more sales data.

5.1 Insights and Management Recommendations

Our recommendation to Baker Hughes' management team is that, although our model has a low F2-score due to the sparse dataset, the recommender system still led to a possible downtime reduction of 6.44% per customer and an additional $660k in annual revenue for Baker Hughes.

Our model was developed using data from one product line within Baker Hughes. We recommend that the company begin to use our model within this product line to reduce downtime for its customers and

increase its revenues. Once the model has been successfully implemented within the first product line, we recommend that members of the data science team scale and adapt it to the other product lines within Baker Hughes. Incorporating more customer and product data will improve the model's performance in making more accurate recommendations.

5.2 Future Research

There is more opportunity to research machine learning recommender systems within the oil and gas industry. As more sales history data becomes readily available, machine learning recommender systems will only continue to improve. Incorporating useful customer features, product features, and product data can lead to better and more accurate recommendations. Our model was limited to a sparse dataset, a dependent binary variable, and a user-based collaborative filtering technique. Future research can evaluate the use of other collaborative filtering techniques such as item based and hybrid models. Additionally, the use of more complex machine learning algorithms such as matrix factorization and deep learning could result in more robust recommender systems for companies in the oil and gas industry.

REFERENCES

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering, 17(6)*, pp. 734-749.

Aghabozorgi, F., & Reza Khayyambashi, M. (2018). A new study of using temporality and weights to improve similarity measures for link prediction of social networks. *Journal of Intelligent & Fuzzy Systems, 34(4)*, pp. 2667-2678.

Al-Najjar, B., & Alsyouf, I. (2003). Selecting the most efficient maintenance approach using. *International Journal of Production Economics, 84*(1), pp. 85-100.

Baker Hughes Company. (2016). *The Impact of Digital on Unplanned Downtime.* Retrieved from bhge.com: https://www.bhge.com/sites/default/files/2017-12/impact-of-digital-on-unplanned-downtime-study.pdf

Barber, D. (2012). *Bayesian reasoning and machine learning.* Cambridge University Press.

Chima, C. M. (2007). Supply-Chain Management Issues In The Oil And Gas Industry. *Journal of Business & Economic Research (JBER), 5*(6), pp. 27-36.

Chomboon, K., Chujai, P., Teerarassamee, P., Kerdprasop, K., & Kerdprasop, N. (2015). An empirical study of distance metrics for k-nearest neighbor algorithm. *In Proceedings of the 3rd international conference on industrial application engineering*, pp. 1-6.

Christensen, P. J., Graf, W. H., & Yeung, T. W. (2013). Refinery power failures: causes, costs and solutions. *18*(4), pp. 103-112. Retrieved from Digitalrefining.com.

Cremonesi, P., Koren, Y., & Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. *In Proceedings of the fourth ACM conference on Recommender systems*, pp. 39-46.

Dangeti, P. (2017). *Statistics for Machine Learning .* Packt Publishing Ltd.

Dobbin, K. K., & Simon, R. M. (2011). Optimally splitting cases for training and testing high dimensional classifiers. *BMC medical genomics 4(1)*, p. 31.

Dunjko, V., & Briegel, H. J. (2018). Machine learning & artificial intelligence in the quantum domain: a review of recent progress. *Reports on Progress in Physics, 81(7)*, 074001.

Fraser, K. (2014). Facilities management: The strategic selection of a maintenance system. *Journal of Facilities Management, Vol 12 No. 1*, pp.18-37.

Keilwagen, J., Grosse, I., & Grau, J. (2014). Area under precision-recall curves for weighted and unweighted data. *PloS one, 9(3), e92209*.

Krawiec, T. (2018). *The Amazon Recommendations Secret to Selling More Online*. Retrieved from Rejoiner.com: http://rejoiner. com/resources/amazon-recommendations-secret-selling-online

Leach, N. (2019). *Global Oil & Gas Exploration & Production.* IBISWorld Industry Report.

Linden, G., Smith, B., & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing 7(1)*, pp. 76-80.

Nguyen, D., Nguyen, C., Duong-Ba, T., Nguyen, H., Nguyen, A., & Tran, T. (2017). Joint network coding and machine learning for error-prone wireless broadcast. *IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 1-7. IEEE.

Park, Y., Park, S., Jung, W., & Lee, S. G. (2015). Reversed CF: A fast collaborative filtering algorithm using a k-nearest neighbor graph. *Expert Systems with Applications, 42(8)*, pp. 4022-4028.

Regnier, E. (2007). Oil and energy price volatility. *Energy Economics, 29*(3), pp. 405-427.

Rose, J., Berndtsson, M., Mathiason, G., & Larsson, P. (2017). The advanced analytics Jumpstart: definition, process model, best practices. *JISTEM-Journal of Information Systems and Technology Management, 14(3)*, pp. 339-360.

Russel, S. J., & Norvig, P. (2016). *Artificial Intelligence: a modern approach .* Malaysia : Pearson Education Limited .

Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS one, 10(3)*.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. *Proceedings of the 10th international conference on World Wide Web*, pp. 285-295.

Shalev-Shwartz, S., & Shai, B.-D. (2014). *Understanding Machine Learning: From Theory to Algorithms.* Cambirdge University Press.

Singh, A., Thakur, N., & Sharma, A. (2016). A review of supervised machine learning algorithms. *In 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom) Ieee*, pp. 1310-1315.

Telford, S., Mazhar, M. I., & Howard, I. (2011). Condition based maintenance (CBM) in the oil and gas industry: an overview of methods and techniques. *International Conference on Industrial Engineering and Operations Management, Kuala Lumpur, Malaysia*, (pp. 22-24).