SMU Data Science Review

Volume 2 | Number 3 Article 3

2019

Predict Missed Ship Date Occurrence and Determine Root Cause of Missed Ship Dates for XYZ Furniture Company

Selwyn Samuel
Southern Methodist University, selwyns@mail.smu.edu

Shon Mohsin
Southern Methodist University, smohsin@smu.edu

Rida Moustafa
Southern Methodist University, rmoustafa@smu.edu

Predict Missed Ship Date Occurrence and Determine Root Cause of Missed Ship Dates for XYZ Furniture Company

Shon Mohsin[1], Selwyn Samuel[1], Rida Mustafa[1]

[1]Master of Science in Data Science, Southern Methodist University
Dallas, Texas 75275 USA
{smohsin, selwyns, rmoustafa}@smu.edu

Abstract. In this paper we analyze sales data from 2016-2018 consisting of 5,547,066 records and 30 categorical variables for XYZ Furniture company (XYZ) and answer two questions of interest. The first is given the sales order data, can we predict if the shipment will miss the estimated delivery date that is generated from XYZ's internal shipment delivery date estimator. We run the data through Support Gradient Descent Boost, Random Forest, Shallow Neural Net, and Deep Neural Net models. The Deep Neural Net model provides the highest accuracy at 84.74% of predicting if a given record will miss the estimated ship date. This is an improvement of 17.97% from the current system performance. The second question of interest is when an estimated ship date is missed, can we determine the root causes of that missed shipment. We create clusters on the sales data and perform Association Rule Mining on the clusters. The rules point to Ship Plant 1, grouping deliveries, and selecting XYZ Company's transportation as the shipping method respectively as contributing factors for missed ship dates. Using the DNN solution, XYZ can expect a 17.97% increase in their prediction of missed ship dates. Investigating Ship Plant 1, grouping deliveries process, and using XYZ shipping services, XYZ may uncover some operational improvement opportunities.

1 Introduction

XYZ Furniture Company is one of the world's leading manufacturer of reclining chairs, upholstered sofas, love-seats, chairs and ottomans based in the US. XYZ markets its products, primarily in the United States, Canada and also internationally, to furniture retailers and also directly to consumers through stores that it owns and operates. XYZ sells their products to customers through multiple channels including their website, retail stores and also marketplaces like Amazon and QVC. XYZ products are highly customizable to customers with various choices for fabrics, leather, style, color and other options. In order to support multiple channels of business sales, XYZ leverages several custom built and commercial software products integrated together to provide the best customer experience by communicating expected product ship dates. Sales orders processed by the applications include wholesale orders from the dealers, retail customer orders from stores and online orders from the XYZ e-commerce website and other marketplaces.

Businesses of all sizes have increasing capability to collect large amounts of data from various activities of day to day business. Businesses are also beginning to show an interest in Artificial Intelligence and Big Data Analysis, and how the large amounts of continuous data that is collected can be utilized to provide business insight to make informed decisions. Moreover, the interest in predictive modeling to gain business insight is not only limited to past static data, but also extend to continuous incoming data that can be deployed into live production environments. Specifically, businesses that ship products to customers have a keen interest in communicating accurate delivery dates to those customers to enhance the customer experience, and a need to analyze root causes of why communicated estimated shipping dates are missed so remedial actions can be taken.

A sales order line is considered to have missed the estimated shipped date if the product is not shipped within the Saturday of that week. We create a categorical variable denoting the missed ship date comparing the shipped date with the estimated shipping date. In this paper, we create predictive model that can determine if a shipment will miss the XYZ-estimated ship date for a given order provided the order details. Support Gradient Descent, Random Forest, Shallow Neural Network (SNN), and Deep Neural Network (DNN) models are built and the results compared on different metrics to determine the best model.

This paper predicts; based on a specific order's associated details, if that order will miss the estimated ship date. The outcome of this prediction is binary, either yes or no. This is represented by 1 for missed or 0 for not missed in our imputed 'missed' column.

When a shipment is missed, we analyze those shipments by first running the data through a clustering algorithm, and then running Association Rule Mining (ARM) to extract relationships that may be present in the data that are causing shipments to be missed.

We conclude that the DNN yields the best result, with an accuracy score of 84.74%, an increase of 17.97% from the current shipping estimation process in place at XYZ Furniture Company. We determine that Ship Plant 1, and the use of XYZ shipping contributes to missed dates.

The remainder of this paper is organized as follows. The Overview and Methodology provides a tutorial and a description of the methods used. The Analysis provides details on the interpretation of the solution. The Ethics provide ethical considerations in respect to the solutions derived. The Conclusion provide a summary of the results and suggestions to XYZ Furniture Company. The Appendix provides data manipulation and additional context.

1.1 Overview of Methodology

1.1.1 Neural Networks using Tensorflow

This paper uses **TensorFlow**, an end-to-end open source platform for machine learning [2] using the Keras high level API to run the Shallow and Deep Neural Networks. Using open-source software instead of a closed proprietary system ensures that additional costs related to deployment can be avoided.

TensorFlow is an interface for expressing Machine Learning algorithms. the computations performed are expressed as stateful data flow graphs. Scalability is ensured using Tensorflow, as it allows parallel execution of the core model. The programming model of TensorFlow revolves around the *directed graph*. The directed graph represents the data flow computation. The directed graph itself is composed of *nodes*. Nodes maintain and update the persistent state of the data to be used. I has branching and looping controls for piping data. Each individual node has inputs and outputs, which instantiates the operation to be performed. Data fed to the nodes are called *tensors*. Tensors are the actual data values that flow along the edges of the graph. Tensors are arrays of data used by the nodes. The directed graph also includes *control dependencies*. Control dependencies point to the source node, and sets the execution order for the source and destination nodes. Users interact with the control dependencies to set the execution order of the nodes. It also allows for the control of memory usage as part of the distributed compute instructions [3].

The actual running of the model is comprised of *operations* and *kernels*. Operations are computations, like add or subtract, to be performed on the tensors, and consists of attributes that are declared during the graph construction, and instantiates the nodes. Kernels are the particular implementation of the operation that is to be run. The *TensorFlow binary* defines the set of operations and kernels available, and can be linked to additional operation or kernels as needed [4].

Finally *sessions* are how users interact with TensorFlow. Sessions are use to augment current graphs with additional nodes and edges. A *run* in the session takes the output to be computed, takes the tensors to be fed into the graph, and uses and closes nodes that need to be computed to produce the outputs. The runs also arrange the order of nodes to be executed. A typical TensorFlow use sets up a session with a graph once, and then executes the full graph or a portion via run calls [4].

Implementation of a TensorFlow system comprises of a client which uses the session interface to communicate with a *master*. A master controls the *worker process* on the devices specified. The worker processes are responsible for arbitrating access to CPUs and devices and execute graph nodes on the devices. The flexibility of TensorFlow allows either local or distributed implementation of the built model [3].

1.1.2 Random Forest

A Random Forest is an *ensemble* of decision trees, trained via the bagging method. An ensemble is an aggregate of predictors, in our case, classifiers within the Random Forest, that when grouped together, give a better prediction that a single classifier can on it's own. Bagging is when the same training algorithm is used for every predictor, but with different random subsets of the data. The data is trained with replacement. Bagging is short for bootstrap aggregating. [4][6].

1.1.3 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is an optimization technique used in Machine Learning. A gradient is the slope of the function. It is the partial derivatives of a set of parameters with respect to its inputs. The purpose of SGD is to find the parameters of the values that minimize the cost function as much as possible[1].

$$for\ i\ in\ range\ (m) : \theta_j = \theta_j - \alpha(\hat{y}^i - y^i)x_j^i \tag{1}$$

1.1.4 Precision, Recall and $F1$ Scores

Precision takes the number of True Positives (TP), which is the number of positives that are correctly predicted, and divides them by the sum of the TP and False Positives (FP), which is the number that is wrongly classified as positive when it is not, giving us the formula:

$$precision = \frac{TP}{TP + FP} \tag{2}$$

We further calculate the Recall, which take the TP and divides it by the sum of the TP and False Negatives (FN), which is the number of positives that were classified as negative, yielding the formula:

$$recall = \frac{TP}{TP + FN} \tag{3}$$

F_1 score is the weighted average of precision and recall. It takes the scores of both false positives and false negatives. This is best used if the data has uneven class distribution, and can be derived by applying the formula:

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{4}$$

1.1.5 Southern Methodist University High Performance Compute

We use Southern Methodist University's High Performance Compute servers to run our analysis, as the volume of data provided and the layers of the neural network require very high compute capabilities not possible on personal computers.

1.1.6 Framework for Association Rule Mining

To perform the root cause analysis for the missed shipments, we use the framework described in Figure 1.

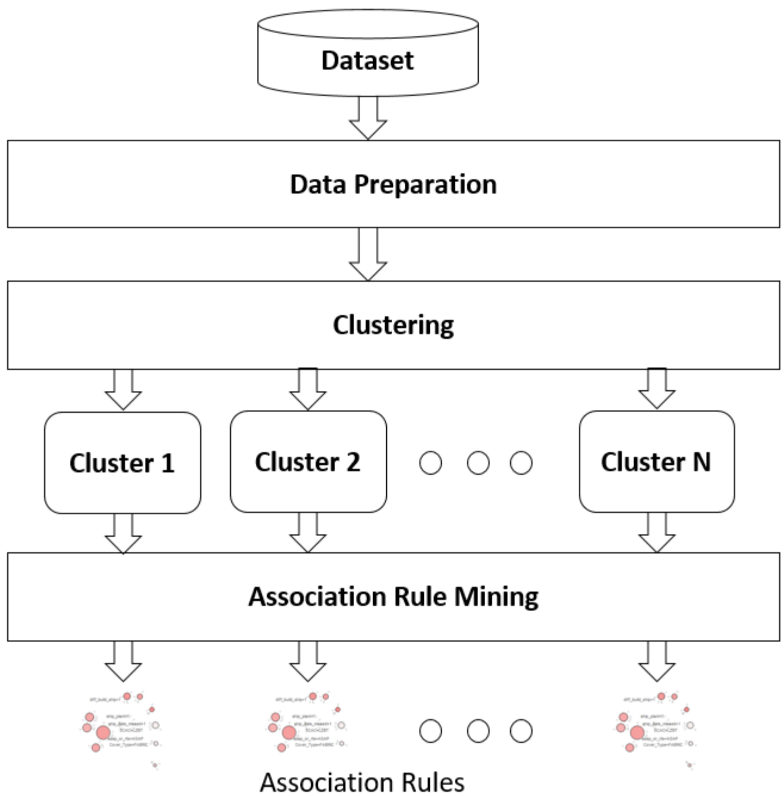

Fig. 1. Framework for Root Cause Analysis

The data set contains more than 5 million records which can be resource intensive for data mining algorithms used for analysis. We group the data into clusters using clustering algorithms and then run association rule mining algorithms for each cluster to get the association rules.

1.1.7 Clustering

Clustering is a process to partition data into groups in way that the observations in a group are similar to each other within the group than with the ones in other groups. There are many algorithms for clustering of data including K-Means, Mean-Shift, Density-Based Spatial Clustering of Application with

Noise (DBSCAN), Expectation-Maximization, Agglomerative Hierarchical and K-Modes. Since the XYZ sales order data contains only categorical attributes, we chose K-Modes clustering algorithm as it is the recommended approach for categorical attributes [8]. K-Modes is used mainly for categorical data and is similar to K-Means clustering with the use of modes instead of the means.

K-Modes clustering algorithm is a modified version of the well-known K-Means algorithm with the use of Hamming Distance to measure the distance between two observations. In a given data set of m number of categorical variables, the distance between two rows X and Y is defined as:

$$d(X,Y) = \sum_{i=1}^{m} \delta(x_i, y_i) \tag{5}$$

where,

$$\delta(x_i, y_i) = \begin{cases} 0, x_i = y_i \\ 1, x_i \neq y_i \end{cases} \tag{6}$$

In the above equations, x_i and y_i are the values of attribute i in the row X and Y. Two rows are have greater distance from each other if more attributes are distance from each other using the above formula. The optimum number of clusters are chosen by the *Elbow Method* in which the Akaike Information Criteria (AIC) is plotted against each cluster. We then observe the change of slope from steep to shallow to find the optimal cluster number.

1.1.8 Association Rule Mining

Association Rules in general describe the co-occurrence of items in a data set. It is a popular algorithm used to mine information in a large databases containing historical data. A frequently quoted example of association rules is in the domain of sales transactions. A given rule $\{Bread, Egg\}(antecedent) \rightarrow \{Milk\}$ (consequent), specifies the co-occurrence of Bread and Egg with Milk and does not imply casualty.

Association rule mining extracts the underlying relationship between various factors in a data set. The rules define the hidden patterns that are uncovered in the data set. A rule $A \Rightarrow B$ means if item set A occurs then B will also occur. We also use clustered association rules [5] to reduce the number of rule sets generated by the algorithm. Apriori algorithm is widely used for mining association rules in a data set.

The strength of the association rules generated from the algorithm are determined by their interesting measures. Threshold criteria of *Support* and *Confidence* are used to measure the strength of the association rules. *Support* for a rule $A \Rightarrow B$ denotes the percentage of all transactions in the data set that contains A and B. *Support* can be calculated using the following equation.

$$\mathbf{Support} = \mathbf{P(A \cap B)} \tag{7}$$

Confidence for a rule $A \Rightarrow B$ denotes the percentage of all transactions containing A in all transactions containing B. *Confidence* can be calculated using the following equation.

$$\mathbf{Confidence} = \mathbf{P}(\mathbf{A}|\,\mathbf{B}) = \frac{\mathbf{P}(\mathbf{A} \cap \mathbf{B})}{\mathbf{P}(\mathbf{A})} \tag{8}$$

Lift is another interesting measure calculated using the following equation. A *Lift* value of greater than 1 denotes the appearance of A and B together is more than expected while a value less than one will denote the opposite.

$$\mathbf{Lift} = \frac{\mathbf{P}(\mathbf{A} \cap \mathbf{B})}{(\mathbf{P}(\mathbf{A})\mathbf{P}(\mathbf{B}))} \tag{9}$$

Our analysis uses ARM Apriori algorithm to identify the antecedents for the consequent $\{ShipMiss = T\}$. We identified the item sets in the sales order data that occur frequently with the missed ship date within each cluster.

2 Data

The full data set provided by XYZ consists of 5,547,066 sales records for 3 years of sales data form 2016-2018 in the US. The data set consists of 30 columns of all categorical data. "ALI" is dropped as this column is a company index column and not needed for our analysis. "Order_Ship_to_Country" is dropped since the area of interest is only the United States. "fulfillment_method" is dropped as the value is 'US Plant' for all orders in the United States.

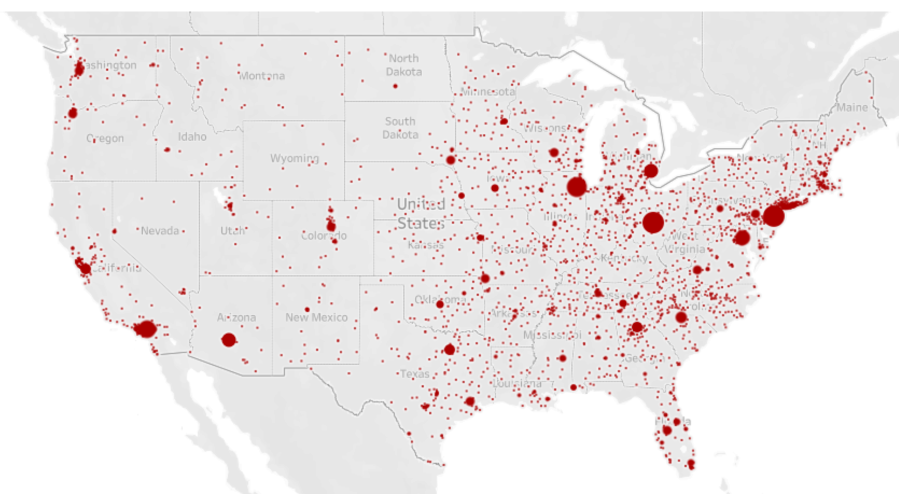

Fig. 2. Orders by Zip Code

A map of orders plotted from the data on the United States shows the order hot spots in Figure 2.

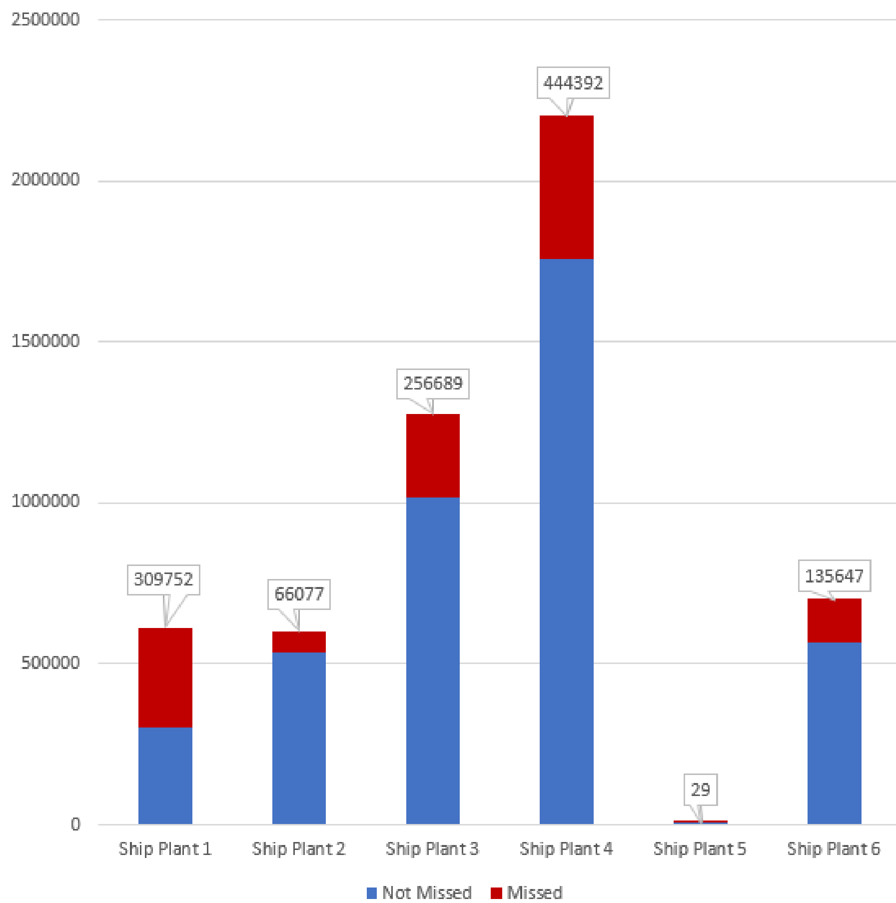

Fig. 3. Missed Orders by Ship Plant

Plotting the instances of missed shipments by plant in Figure 3, we observe that the missed shipments occur roughly 25% of the time. The exception is Ship Plant 1, where nearly half the shipments are missed.

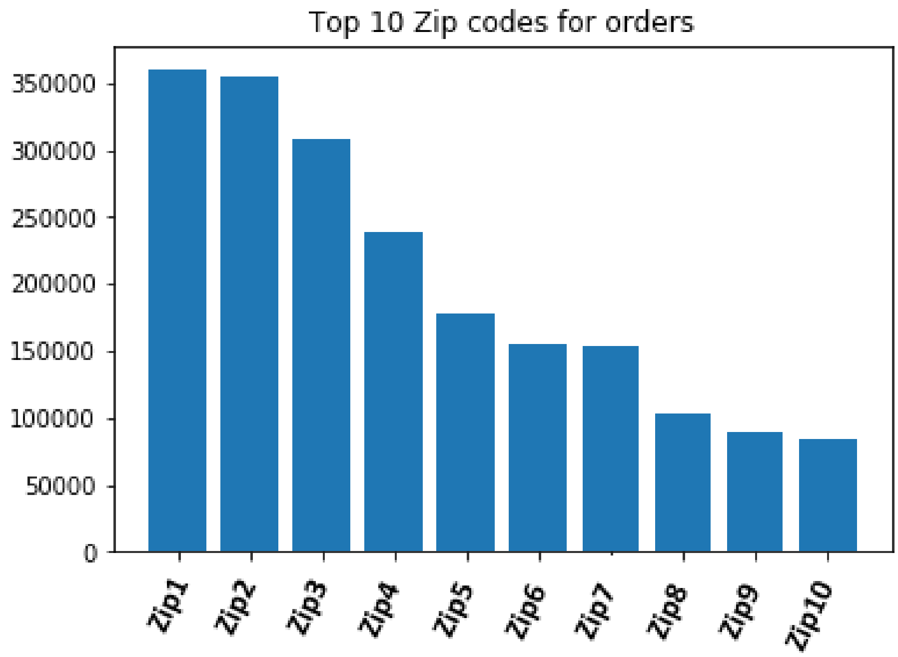

Fig. 4. Top Zip Codes

The Top Zip Codes for delivery show where potential challenges may be encountered during the delivery process in Figure 4.

We begin with the columns provided in Table 1.

Column	**Description**	**Unique**	**NaN**
asap_o_rts	Ship ASAP or at requested date	2	0
build_cell	Order build cell	349	87554
build_plant	Order build plant	6	0
current_ets_date	Updated estimated time to ship	196	6
customer_sold	Sold to customer or Dealer	2	0
order_plant	Order received plant	6	0
original_ets_date	Initial estimated time to ship	190	0
rts_date	Requested ship date	359	4758455
set_source_code	Internal code	9	23
ship_group	Group shipping code	952470	2860603
ship_plant	Plant that shipped the order	6	65203
shipped_date	Actual ship date of order	914	0
Primary_AE_NO	Primary Account Executive ID	88	0
Responsible_AE_NO	Responsible Account Exec ID	70	0
Regional_Sales_Territory	Sales territory in US	13	0
cover	Type of product cover	2422	0
Grade	Grade code for the product	35	0
Pattern	Pattern associated with the product	1032	0
Color	Color of the product	184	0
Cover_Type	Type of product cover	2	0
Finish	Type of finish for the product	16	0
Prefix	Prefix code for the product	180	0
Suffix	Suffix code for the product	306	0
SCAC	Shipping Std. Carrier Alpha Code	53	90527
Pool_Code	Code related to pooling shipments	31	4826319
Pool_Zip_code	Zip code for pooling shipments	33	4789113
Order_Ship_to_Zip_Code	Zip code of order destination	2862	0
Option_String	String containing additional options	125818	3601277

Table 1. Data Columns with Unique Counts and NaNs

For the entire data set, 11834 rows are dropped that have invalid Zip codes, as we assume this is a data error, and not a significant amount of records given the size of the data set. All missing data is then filled with zeros.

For machine learning preparation, the entire data set is one-hot-encoded (OHC). This is done to optimize the machine learning algorithms. For example, a 'ship_plant' value of 6 is in no way greater than 1. Without OHC, the algorithm may assign a higher weight to 6 than to one. Placing each level in a column in it's own column of binary ones and zeros mitigates this incorrect weighting. Additionally, some columns have an extreme number of levels, with each level having no statistical significance to the next level. Creating these 'dummy' variables allows each level to be evaluated on it's own merit and evaluated in the model. A dummy variable is an independent variable that either takes the value of 0 or 1 for presence or absence of the level for the record [10].

The resultant data set after OHC consists of 1833 columns and 5,535,203 rows. With the data set prepared for machine learning algorithms, we proceed with partitioning the data for our machine learning algorithms. The data is split into two. 20% of this data is set aside as a validation data set for later use to test the performance of our models using a random sampling method. The rest 80% of the data is further split into train data of 70% and 30% is used to test using a random sampling.

2.1 Methods and Experiments

2.1.1 Stochastic Gradient Descent

We take a small subset of 15% of the test and train data and run it through a simple Stochastic Gradient Descent (SGD) classifier. We use a classifier because the 'missed' column that we are trying to predict consists of either 1 or 0, a simple classification. The purpose of this step is to get a baseline that we can compare against as we iterate over different models.

We assume that the data does not have any strong linear relationships within itself.

2.1.2 Random Forest Classifier

The random forest classifier is run with 15% of the data set with the hyper-parameters as described in Table 2.

Hyper-parameter	**Value**
bootstrap	True
criterion	gini
max depth	none
max features	auto
max leaf nodes	none
min impurity decrease	0.0
min impurity split	None
min samples leaf	1
min samples split	2
min weight fraction leaf	0.0
n estimators	500
n jobs	-1
oob score	False
random state	None
verbose	0
warm start	false

Table 2. Random Forest Hyper-parameters

2.1.3 Initial Neural Network Model with Unbalanced Data

The data set target is unbalanced, with 'missed' =1 occurring 21.98% within the data set. This imbalance needs to be addressed increase the F_1 score which is done for the Deep Neural Network.

For comparison, we run a simple neural network with total trainable parameters of 121,537 while ignoring the balance issue. The model is run with the configuration in Table 3.

Layer	Output Shape	Parameter
input layer	(none, 1332)	0
dense_1 (Dense)	(none, 64)	117312
dense_2 (Dense	(none, 64)	4160
predictions (Dense)	(none, 1)	65

Table 3. Shallow Neural Net Layers

This is run over 10 epochs with a batch size of 256. The model is trained over 3,099,713 records. For each epoch, the model is then evaluated over the test data set.

2.1.4 Final Deep Neural Network Model with Bias-corrected Data

On training our final Neural Network Model, we take into account the unbalanced nature of the data set. We offset this by setting an initial bias term to the model by taking the log of the positive and negative counts.

$$BiasTerm = log(\frac{PositiveCount}{NegativeCount}) = -1.2668 \tag{10}$$

We configure the layers in as displayed in Table 4 for a total of 666,881 trainable parameters.

Layer	Output Shape	Parameter
dense_21 (dense)	(none, 256)	469248
dense_22 (dense)	(none, 256)	65792
dense_23 (dense)	(none, 256)	65792
dense_24 (dense)	(none, 256)	65792
dropout_7 (dropout)	(none, 256)	0
dense_25 (Dense)	(none, 1)	257

Table 4. Neural Net Layers

To increase the F_1 score of our model, we first increase the number of dense hidden layers. We increase the batch size to 2048 so the model evaluates more records per batch. We increase the epochs to 20 to allow the model more training iterations. We implement an early stopping criterion on the model where we set the parameter 'restore_best_weights' to TRUE to iterate over the best weights per epoch and stop at epoch 19.

2.1.5 Apriori Algorithm

The Apriori algorithm requires the data to be formatted as transactions which is a sparse matrix of itemsets. The data is transformed into a sparse matrix of itemsets and the algorithm is run.

2.1.6 Clustering

We use the Akaike Information Criteria (AIC) and Bayesian Information Criteria (BIC) to determine the optimal number of clusters for the sales order data. We generate models for 1 to 10 clusters using the K-Modes algorithm.

2.1.7 Association Rule Mining

Unsupervised Machine Learning and data mining techniques are used extensively in retail industry for use cases including customer segmentation and category-based clustering. We leverage association rule mining to determine the rules associated with a missed estimated shipping date. Association rules are run for the data.

The volume of the sales data over the last 3 years along with the cutomizability of the products provide challenges to the association rule mining algorithms and the results we can derive from them.

3 Results

3.1 Prediction Results

Running the Gradient Descent Classifier on a subset of data, we get an F_1 score $= 41.775\%$.

The results of the random classifier are slightly better, with an $F_1 = 56.185\%$ value. This indicates that traditional machine learning algorithms would need to be heavily tuned to achieve higher prediction scores.

For the Shallow Neural Network, we print out the TP, TN, FP, and FN from each epoch, and then calculate the F_1 score as seen in Table 5.

Epoch	Loss	TN	FP	FN	TP	Precision	Recall	F_1
1	0.487392	2286097	132620	392663	288333	0.684953	0.423399	52.33%
2	0.525523	2288722	129995	394721	286275	0.687715	0.420377	52.18%
3	0.574018	2290447	128270	396395	284601	0.689322	0.417919	52.04%
4	0.615192	2290713	128004	397401	283595	0.689008	0.416442	51.91%
5	0.643045	2293303	125414	398862	282134	0.692272	0.414296	51.84%
6	0.667724	2293661	125056	399954	281042	0.692055	0.412693	51.71%
7	0.681175	2294019	124698	401626	279370	0.691394	0.410237	51.49%
8	0.721668	2292435	126282	400321	280675	0.689692	0.412154	51.60%
9	0.764936	2294020	124697	402657	278339	0.690606	0.408723	51.35%
10	0.793685	2295921	122796	404272	276724	0.692641	0.406352	51.22%

Table 5. Shallow Neural Net Training Epoch Results

The trained model is then evaluated for each epoch against the validation data set with results in table 6.

Epoch	Loss	TN	FP	FN	TP	Precision	Recall	F_1
1	0.527607	976139	60003	166102	126205	0.677764	0.431755	52.75%
2	0.564738	978809	57333	168190	124117	0.684029	0.424612	52.40%
3	0.621976	979722	56420	168788	123519	0.686449	0.422566	52.31%
4	0.599447	988009	48133	178164	114143	0.703388	0.39049	50.22%
5	0.65262	976909	59233	165823	126484	0.681058	0.432709	52.92%
6	0.658729	978772	57370	167910	124397	0.684376	0.42557	52.48%
7	0.713291	970165	65977	159504	132803	0.66809	0.454327	54.09%
8	0.750409	988162	47980	178619	113688	0.703219	0.388934	50.09%
9	0.742018	979592	56550	169913	122394	0.683979	0.418717	51.94%
10	0.796909	971060	65082	160809	131498	0.668929	0.449863	53.79%

Table 6. Shallow Neural Net Validation Epoch Results

We observe that the baseline neural network actually performs worse than the random forest, with a net F_1 score of 53.78%.

In the Bias-corrected DNN, the scores in the 19 training epochs shows an improvement in the F_1 score with the best output at 70.14% as seen in Table 7.

Epoch	Loss	TN	FP	FN	TP	Precision	Recall	F_1
1	0.371602	2283778	134464	394759	286712	0.680742	0.420725	52.00%
2	0.364339	2277653	140589	364359	317112	0.692837	0.465335	55.67%
3	0.353626	2277261	140981	348689	332782	0.702423	0.488329	57.61%
4	0.344653	2277931	140311	336056	345415	0.711131	0.506867	59.19%
5	0.336934	2279020	139222	325822	355649	0.71867	0.521884	60.47%
6	0.329979	2280315	137927	317210	364261	0.725348	0.534522	61.55%
7	0.323667	2281393	136849	308669	372802	0.731485	0.547055	62.60%
8	0.317855	2282360	135882	300401	381070	0.737148	0.559187	63.60%
9	0.312349	2282943	135299	293213	388258	0.741577	0.569735	64.44%
10	0.307315	2284142	134100	286857	394614	0.746366	0.579062	65.22%
11	0.302556	2285543	132699	281239	400232	0.751002	0.587306	65.91%
12	0.29799	2286927	131315	275519	405952	0.755587	0.5957	66.62%
13	0.293709	2288742	129500	270649	410822	0.760328	0.602846	67.25%
14	0.289487	2290729	127513	266284	415187	0.76504	0.609251	67.83%
15	0.285705	2291655	126587	261553	419918	0.76837	0.616193	68.39%
16	0.282187	2294276	123966	257646	423825	0.773698	0.621927	68.96%
17	0.278773	2295778	122464	254519	426952	0.777102	0.626515	69.37%
18	0.275441	2297883	120359	251898	429573	0.781138	0.630361	69.77%
19	0.272428	2299539	118703	249297	432174	0.78452	0.634178	70.14%

Table 7. Deep Neural Net Training Results

During the validation phase of the training epochs, the F_1 score goes up as well, with the highest score in epoch 13 at 60.07% in table 8, although not as much as during the training phase.

Epoch	Loss	TN	FP	FN	TP	Precision	Recall	F_1
1	0.371602	983503	53114	167587	124245	0.700528	0.425742	52.96%
2	0.367707	960208	76409	141469	150363	0.663058	0.515238	57.99%
3	0.359492	971279	65338	147483	144349	0.688402	0.49463	57.56%
4	0.358267	971966	64651	147392	144440	0.6908	0.494942	57.67%
5	0.357163	976408	60209	150566	141266	0.701159	0.484066	57.27%
6	0.357103	972012	64605	145411	146421	0.693853	0.50173	58.24%
7	0.357772	974511	62106	147256	144576	0.699509	0.495408	58.00%
8	0.359057	967385	69232	140955	150877	0.685465	0.516999	58.94%
9	0.359942	966855	69762	140212	151620	0.68488	0.519545	59.09%
10	0.361454	963678	72939	137172	154660	0.679528	0.529962	59.55%
11	0.363758	970454	66163	143574	148258	0.691434	0.508025	58.57%
12	0.365576	965496	71121	138982	152850	0.682454	0.52376	59.27%
13	0.368882	955092	81525	131553	160279	0.662847	0.549217	60.07%
14	0.370827	959109	77508	133596	158236	0.67122	0.542216	59.99%
15	0.372422	957308	79309	132507	159325	0.667654	0.545948	60.07%
16	0.374927	955639	80978	132976	158856	0.662358	0.544341	59.76%
17	0.375941	963720	72897	137945	153887	0.678562	0.527314	59.35%
18	0.379038	969572	67045	143002	148830	0.689427	0.509985	58.63%
19	0.383426	958117	78500	134303	157529	0.667414	0.539793	59.69%

Table 8. Deep Neural Net Validation Results

The final F_1 score of the biased neural network based on validation set yields a score of 59.69%.

3.2 Root Cause Analysis Results

3.2.1 Apriori Algorithm

The sparse matrix for the sales order attributes is shown in *Figure 5*.

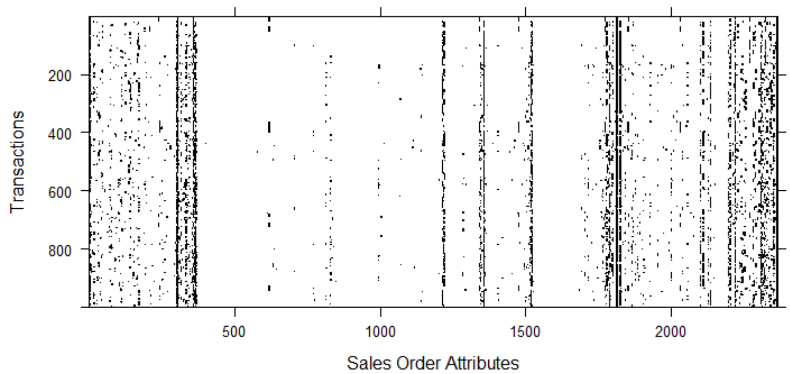

Fig. 5. Sparse Matrix of Transaction Data

Frequent itemsets are the ones that occur above a specified threshold in the transactions. The frequent itemsets in the sales order transactions data are shown in *Figure 6*.

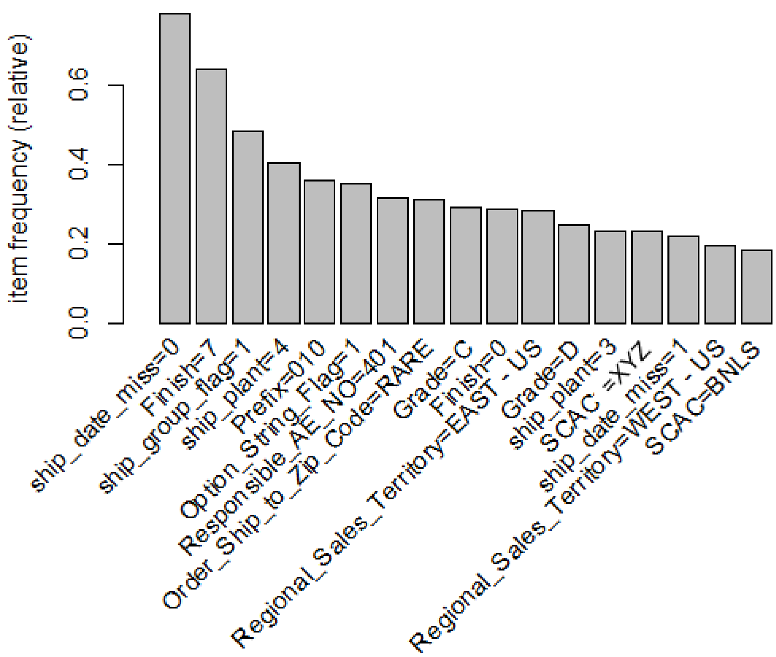

Fig. 6. Frequent Items

The frequent itemsets include product attributes, ship plants, transportation companies and regional sales territories.

3.2.2 Clustering

Figure 7 shows the number of clusters and the corresponding AIC/BIC values. Using the *Elbow* method we have an option to choose either 5 or 9 clusters. We chose smaller of the two, 5 as the optimal number for our analysis.

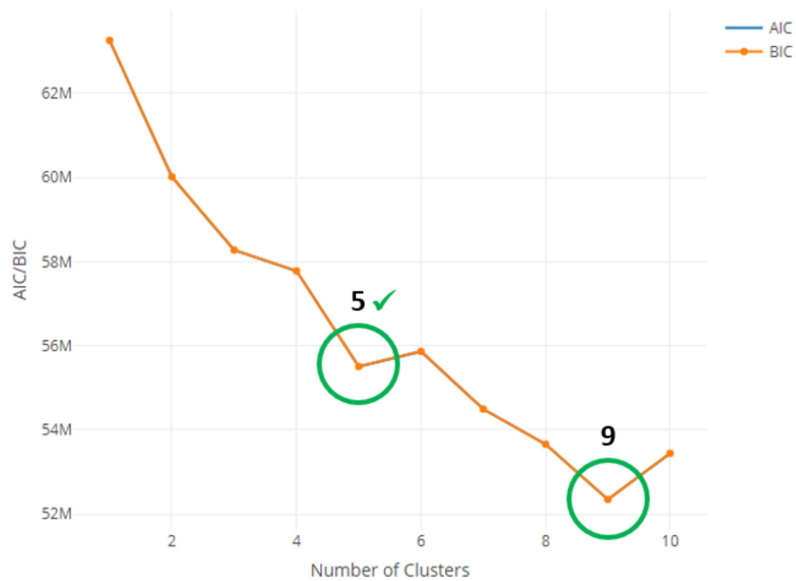

Fig. 7. Optimal Number of Clusters

Figure 8 illustrates the geographical spread of the clusters in the United States map. The clusters are grouped close to some of the shipping plants and the magnitude of the clusters may provide us with clues on where to look for association rules.

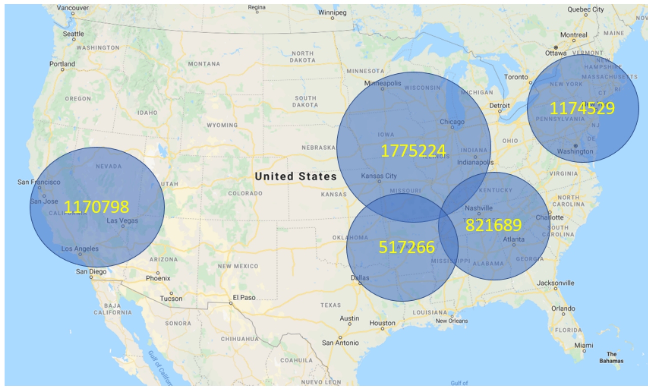

Fig. 8. Geographical distribution of Clusters

Figure 9 describes the properties of each of the 5 clusters identified by the K-Modes algorithm, the count of rows in each of the cluster and the percentage of the size.

Cluster #	Cluster Description	Count	Size (%)
1	Products to US West Coast, XYZ Company transportation and Shipping plant #6	1170798	21.44
2	Products to US Mid West, Shipping plant #3, and Fabric Cover	1775224	32.52
3	Eastern Regional Sales Territory, Shipping plant #4 and Manufactured in a different plant	1174529	21.51
4	Manufactured and Shipped from plant #4, Eastern Regional Sales and additional options	821689	15.05
5	Regional Accounts Managers Sales, shipped to the East Coast, Leather cover and Shipping plant #1	517266	9.47

Fig. 9. Clusters Description

We proceed to perform association rule mining for one of the clusters. We chose *Cluster 1* as the candidate cluster for association rules analysis and observe the output in Figure 10.

3.2.3 Association Rule Mining

The network graph in *Figure 10* shows the visualization for the top 10 extracted association rules in the sales order transactions for *Cluster 1*. The itemsets are represented as vertices and the association rules are the directed edges between itemsets.

Each rule is represented by a circle, the arrows coming into the circle are the antecedents for the rule and the arrow coming out of the circle is the consequent for the rule. We have the same consequent $\{ShipMiss = T\}$ for all the rules. Larger circles denote higher support for the rule while red circles denote higher lift for the rule.

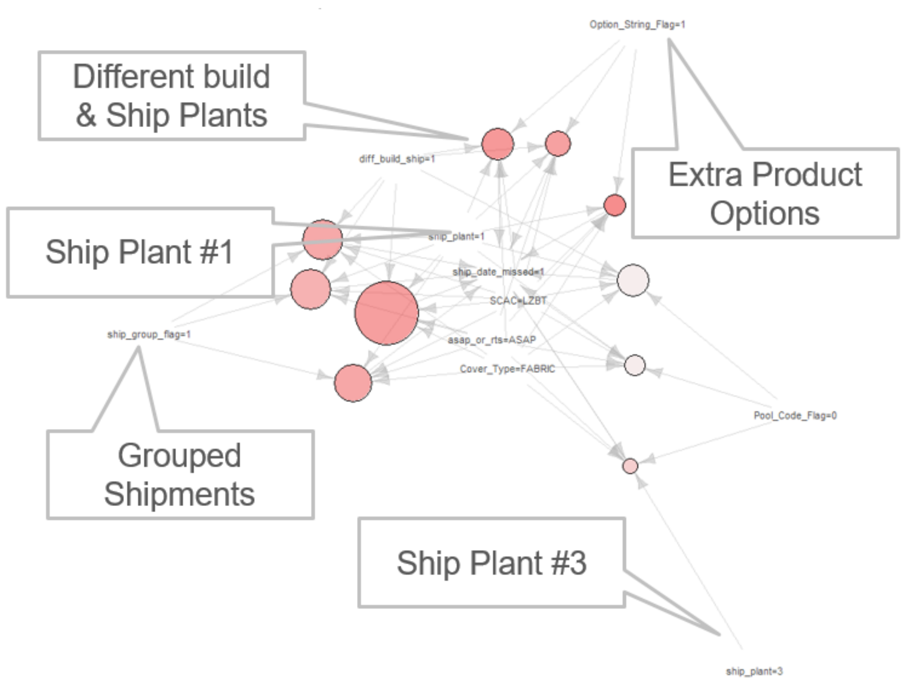

Fig. 10. Association Rules Graph

4 Analysis

4.1 Prediction Analysis

Comparing the 4 classification methods, the $F1$ score of the DNN with bias correction yields the highest result.

Model	F_1
SGD Classifier	41.78%
Random Forest Classifier	56.19%
Shallow Neural Network	53.78%
Deep Neural Network with Bias	59.69%

Table 9. F1 score comparison for all models

4.2 Root Cause Analysis

Clustering the sales order data into the optimal number of clusters provided a logical grouping of the sales orders based on the regions and some of the product attributes. The number of missed shipments for each cluster is shown in *Figure 11*. We identify *Cluster 3* has higher missed shipments ratio than the other clusters. *Cluster 3* includes shipping plant #4, Eastern Regional Sales Territory and the products were manufactured at a different plant than the shipped plant.

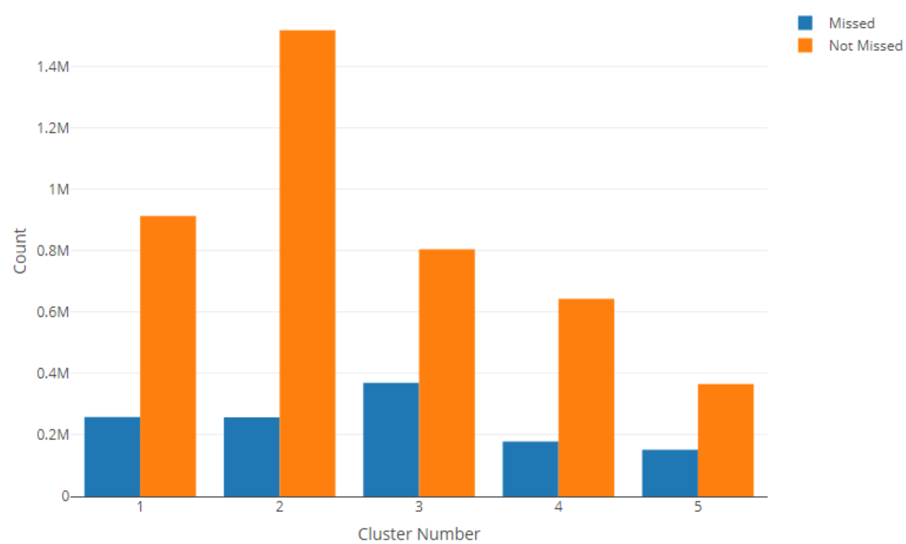

Fig. 11. Shipment miss count across clusters

Figure 12 shows the parallel coordinates plot for the top association rules. Each line represents a rule with the head pointing to the consequent item of the rule $\{ShipMiss = T\}$. The width of the arrows denotes the support and the intensity of the color denotes the confidence for the rule.

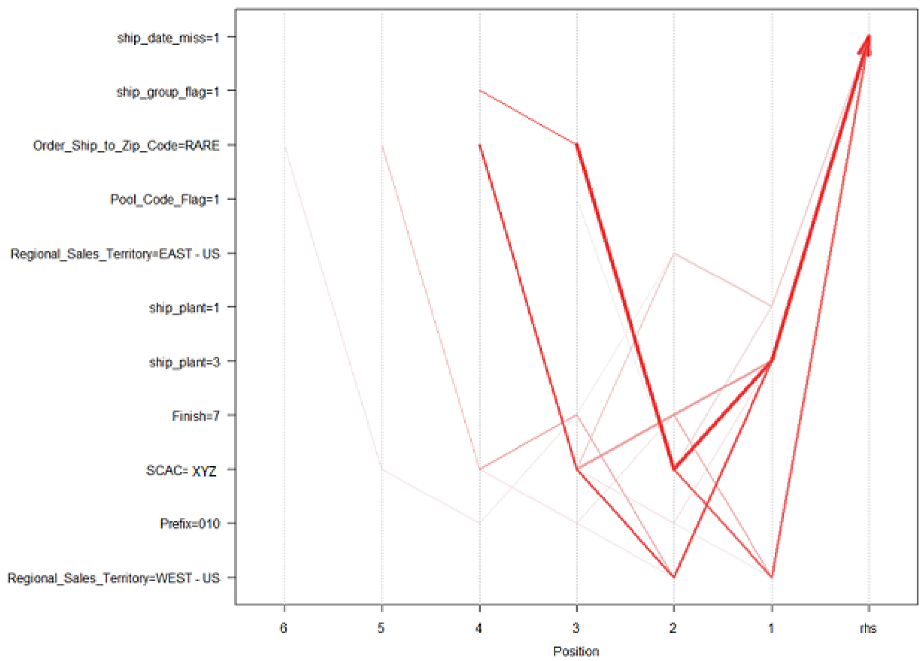

Fig. 12. Parallel Coordinates Plot

Rules with higher confidence point to Ship Plant 3, deliveries to rare zip codes and XYZ transportation as the root cause for the missed shipments.

5 Ethics

The project was sponsored by XYZ to provide analysis of the challenges in meeting the estimated shipping dates provided to the customers and to develop a machine learning model to predict the possibility of missing the estimated ship dates. We were provided with 3 years of sales order data from their Order Management System. There are other systems including their Enterprise Resource Planning (ERP) system that have supply chain related data that could have helped us provide more accurate analysis of estimated shipping dates. We provided a thorough analysis of the sales order data but also highlighted the potential risks in interpreting the results without the inclusion of other relevant data [ACM 2.5].

Confidentiality [ACM 1.7] of the data provided to us was maintained throughout the various phases of the project. We masked sensitive data like the plant identity from the data set using maps to numeric codes. We utilized local and SMU computing resources for our computation instead of public cloud resources to perform our analysis.

Results of prediction algorithms when used for decision making could introduce bias towards an individual, group or organization. Bias can be introduced into Machine Learning (ML) pipeline through the data, classifiers and also the predictions themselves. The impact of bias is more pronounced when black box algorithms like Neural Networks are used in predictions compared to explainable statistical models. Many companies including IBM have created tools that eliminate bias automatically from various steps in the ML pipeline. "An automated process trained to look for historical patterns of success and to suggest that whatever led to success in the past will lead once again to success in the future." [7]

Feature importance of prediction algorithms when combined with the actual predictions has the potential to introduce bias against certain entities. In cases where a specific plant or zip code has high importance in the prediction result, we have highlighted the risk in taking actions based on that information. Since we did not include the entire supply chain data, even though a plant may contribute to a missed shipping date, it could be because of their suppliers delay, transportation delay and not the performance of plant itself [ACM 1.4].

We used association rule mining algorithm to perform the analysis on the historical data. The scope of the analysis was limited to the sales order data provided and the algorithms used for the analysis. We did not perform time series analysis that could have highlighted additional seasonal factors contributing to the missed ship dates. Similar to the prediction results, the association rules need to be interpreted within the context of the data set provided.

Another important note with association rules is that they highlight the co-occurrence of the items and they do not imply causality. The conclusion for the analysis included the limitations of the results and also the potential problems that could arise in decisions made without further analysis [ACM 1.3].

6 Conclusions

To understand the powers of predictions and analysis, imagine an opaque bag of words. The sum of those words when strung together tell a coherent story. Now imagine taking out a word one at a time, and trying each time to tell the whole story based on the word you've picked out. As you pick out more words, the likelihood that you will be able to accurately tell the story increases. It helps to think of the data columns that you receive as the words you receive, and the prediction algorithm as the process of you trying to tell the story of the bag of words. In this way, the data set received is vitally important in trying to tell the story asked.

In this paper, we have discussed prediction models and analysis models for missed shipments using the XYZ sales order data. Missed shipments can be caused because of various other factors in the supply chain that are not present in the sales order data such as supplier delay, inventory availability, manufacturing delay and transportation delay. In the absence of the rest of the supply chain data, the results of this study can at best be used as a starting point for further analysis.

6.1 Prediction Conclusions

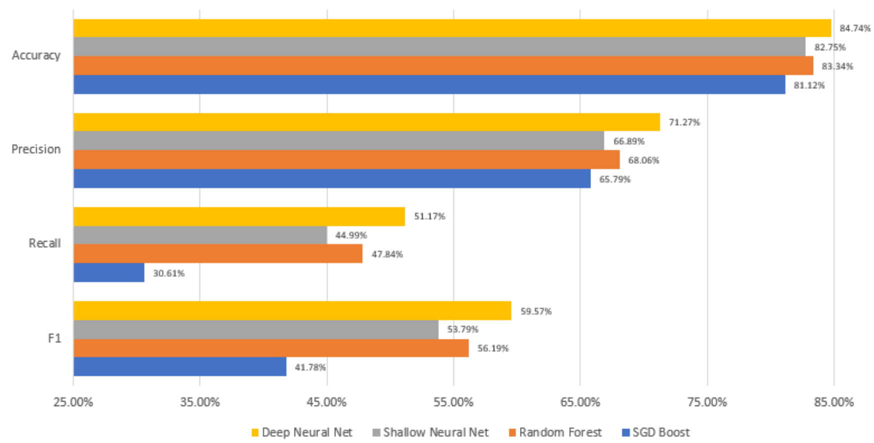

Fig. 13. Final Results Comparison by Model

To tell the story of the prediction task, training a data set with the columns provided yields an F_1 score of 59.6%. The story we try to tell is given an order record, can we predict if that order will miss its estimated ship date. If you were to use this neural network model to answer this question, the model would be accurate 84.74% of the time. Currently, XYZ's inbuilt prediction accuracy is at 71.83%. If XYZ were to use the DNN solution built in this paper, they would see an increase of **17.97%** in their shipment accuracy.

The implementation of the DNN shows an improvement in the F_1 score over all of the other models tried, showing the strength of the methodology and it's application. There is potential to further improve the score by iterating over the hyper-parameter configurations, but without additional data from other sources, mainly new columns that are not in the current data set, the prediction task will not be any orders of magnitude better than what is currently achieved.

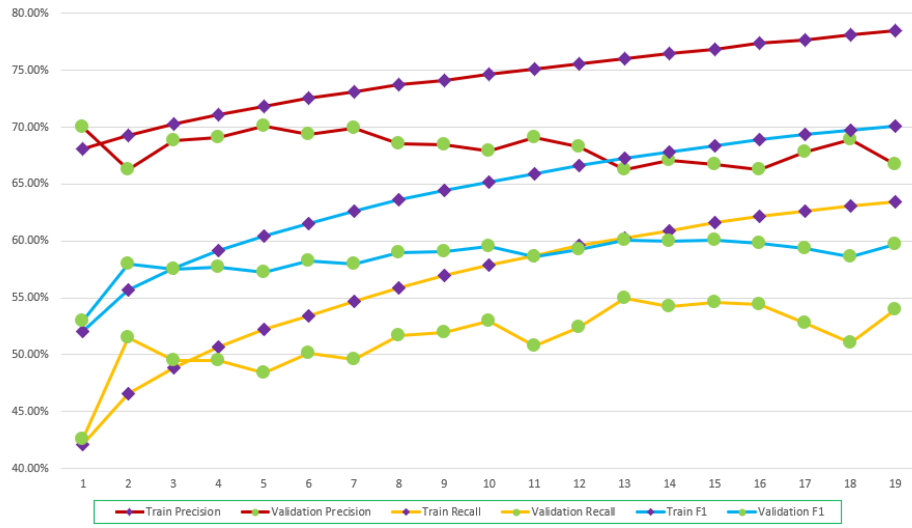

Fig. 14. Precision, Recall, F1 Scores of Deep Neural Network through 19 epochs

6.2 Root Cause Analysis Conclusions

Association Rules highlight the items that co-occur along with the missed shipments. The co-occurrence of the items include customer choosing extra product options, difference in the build and ship plants, customer requesting grouped shipments, ship plants #1 and #3. These could be starting point for further analysis.

7 Recommendations

Based on the Gartner 2018 report [9], we propose an Enterprise AI/ML Platform for XYZ that integrates their applications to the machine learning pipeline for data acquisition, feature engineering, model engineering, model training and deployment. We recommend the prediction and analysis be done using a broader set of supply chain data for better performance. An Application Programming Interface (API) deployed in the cloud can be used to predict ship date misses using ML algorithms. The AI/ML Platform can be the foundation for various other prediction and analysis models within XYZ. We also recommend the investigation of Ship Plant 1 for a higher number of missed shipments compared to other plants, along with investigations of the efficiency of using XYZ internal shipment methods to deliver to customers.

References

1. Optimization: Stochastic gradient descent, http://deeplearning.stanford.edu/

2. Tensorflow, https://www.tensorflow.org/
3. Abadi, M.: Tensorflow: learning functions at scale. ACM SIGPLAN Notices **51**(9), 11 (2016). https://doi.org/10.1145/3022670.2976746
4. Aurelien, G.: Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. OReilly Media (2017)
5. Brain Lent, Arun Swami, J.W.: Clustering association rules Report No. GIT-CC-95-04
6. Liaw, A., Wiener, M., et al.: Classification and regression by randomforest
7. O'Neil, C.: Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy. Crown Publishing Group, New York, NY, USA (2016)
8. Sachin Kumar, D.T.: A data mining framework to analyze road accident data. Journal of Big Data (2015)
9. Sapp, C.: Preparing and architecting for machine learning: 2018 update. Gartner (2018), id G00365935
10. Susan Garavaglia, A.S.: A smart guide to dummy variables: Four applications and a macro. Dun Bradstreet

8 Appendix

Success of Machine Learning algorithms depend heavily on the quality and the quantity of data available for supervised and unsupervised learning and predictions. In organizations such as XYZ, data is stored in various application databases and data marts. We had discussions with the IT leadership to come up with the candidate use cases for the project. We met with the data mart, wholesale and retail order management teams to educate them about the machine learning concepts and the benefits of leveraging the algorithms for predictions.

Working with the key members of the teams, we identified the database tables and the date range for the data we need for the analysis. We also identified the columns that need to be anonymized to protect customer information, proprietary and sensitive data. After going through the internal approval process, we received an initial batch of 2 weeks of sales order data for our exploratory data analysis. The initial data set helped us to get a better understanding of the attributes and also to identify the type of Machine Learning algorithms we can use for our analysis. The following changes were made to the initial dataset to preserve anonymity and to facilitate processing within our algorithms.

1. 'order_plant' is given a sequential value of 1-6 to anonymize the plant names as part of the Non-Disclosure Agreement (NDA) with XYZ.
2. 'build_plant' is given a sequential value of 1-6 to anonymize the plant names as part of the NDA with XYZ.
3. 'ship_plant' is given a sequential value of 1-6 to anonymize the plant names as part of the NDA with XYZ.
4. 'set_source_code' is coded from 1-9 for each of the unique values.
5. 'ship_group' comprises of 52% missing values. If ship group is present, it is a unique value. To reduce the 952,470 levels, any values present are converted to 1, and missing values converted to zero.

6. 'ship_plant' is given a sequential value of 1-6 to anonymize the plant names as part of the NDA with XYZ.
7. A new column 'missed' is derived from subtracting the 'shipped_date' from 'current_ets_date' and then labelling 1 if that difference is greater than 6 days, else zero. This is done to conform to the XYZ's definition of a 'missed' shipment if the actual ship date is 6 days greater the current estimated ship date. This new column is our prediction target.
8. 'original_ets_date' is dropped after 'missed' is calculated.
9. 'current_ets_date' is dropped after 'missed' is calculated.
10. 'Regional_Sales_Territory' is encoded from 1-12 to prepare the columns for one-hot-encoding.
11. 'cover' has 2422 levels, and to reduce the levels it is grouped so that all levels that aggregate to less than 30% of the orders, which is 4000, are converted to the label 'rare'.
12. 'pattern' has 1032 levels, and to reduce the levels it is grouped so that all levels that aggregate to less than 30% of the orders, which is 19000, are converted to the label 'rare'.
13. 'Pool_Code' has 87% missing values. Values present are converted to 1, and missing values converted to zero.
14. 'Pool_Zip_code' has 86% missing values. Values present are converted to 1, and missing values converted to zero.
15. 'Order_Ship_to_Zip_Code' has 2830 levels. The first transformation is converting zip codes that are 9 digit zip codes to 5 digit zip codes. Next, to reduce the levels it is grouped so that all levels that aggregate to less than 30% of the orders, which is 13000, are converted to the label 'rare'.