International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 09, September-2019

An Exploratory look at Data Extraction and Machine Learning for Detecting Fraudulent Financial Journal Entries

Anshuman Guha
Graduate Student, Department of Computer Science
John Hopkins University
Baltimore, Maryland, USA

Eyob Gebremariam
Graduate Student, Department of Data Science
Southern Methodist University
Dallas, Texas, USA

***Abstract*— The Accounting and Auditing professions are just beginning to recognize the value of Data Mining and Analytics for detection of outliers and 100% population testing. At present, the Financial Auditing profession has relied heavily on random or judgmental sampling to validate the completeness and accuracy of financial statements. This paper will examine a proposed method of examining 100% of the past fiscal year's financial journal entries straight from the company in question's Enterprise Resource Planning (ERP) system, applying a machine learning algorithm to the data, and outputting potentially anomalous journal entries to be examined manually in more depth.**

***Index Terms*—Change detection algorithms, Classification algorithms, Detecting fraudulent financial journal**

I. STATEMENT OF PROBLEM

THE current financial auditing environment is unreliable in detecting corporate earnings manipulations, accounting errors and fraud. The materiality sample testing approach (add support) used by the Big Four accounting firms is antiquated, and has not kept pace with the changing dynamics of the business landscape. Specifically, with the increases in technology in the light of the revolutionizing work being done at Rutgers University under the Direction of Miklos Vasarhelyi, PhD [1] on the continuous auditing paradigm, the current method of sampling journal entries is not as effective as it could be. Our project employed three-machine learning driven continuous auditing approaches that can be used by both internal and external auditors to identify anomalous journal entries as they occur, instead of waiting months after the fact for a sampling tie out approach.

This method uses a scalable model that extracts data from the entity in question's production financial databases, applies a machine learning algorithm, either classifier or cluster, which can be run ad hoc or can be scheduled on any time frame. Our program is designed to require a low level of maintenance and is easily adjustable by only changing the schedule date and SQL code to adapt to different business and situations.

II. SOLUTION METHODOLOGY

Our team has developed a seven-step plan of action for addressing our research question.

1. Obtain and analyze financial accounting data to determine suitability to use in testing.
2. Write SQL queries and python code to import the chosen financial information into python for research.
3. Create a training data set and validate its effectiveness.
4. Transfer and transform the raw financial data into a scikit-learn [2] using the python programming language.
5. Experiment with Cluster and Classifier algorithms to detect journal entry exceptions hat are viable. We will test different various hyper parameters and will test our model with cross validation techniques.
6. Develop a python application that imports the data, performs the chosen algorithm, produces a graphic, and exports the exception results to a network drive.
7. Adapt the model to work for both monthly and yearly time periods.

Our solution methodology focuses on understanding how to determine anomalous potentially fraudulent journal entries, the classification and clustering machine learning algorithms, and combine these two to produce a powerful tool to overcome the problem that our research is trying to address.

III. STATEMENT OF PREVIOUS WORK RELATED TO THE PROBLEM.

There has not been a significant amount of academic research related to using machine learning to test journal entries, compared to other research areas, however a few studies have been conducted using machine learning, although not taking the exact approach of this research team [3] [4].

Since the major frauds of Enron and WorldCom, there has been a significant amount of authoritative guidance regarding the need for thorough journal entry fraud testing [5]; and implementation articles have been published regarding how to perform Journal Entry testing [6].

And how to perform testing using analytics and computer-assisted audit tools (CAATs) software, such as CaseWare, Interactive Data Extraction and Analysis (IDEA) and Audit Command Language (ACL) [7].

IV. USING CLUSTERING MACHINE LEARNING ALGORITHM TO ANALYZE FRAUDULENT FINANCIAL JOURNAL ENTRIES

The k-means cluster unsupervised machine-learning algorithm was chosen as one of the algorithms for our project.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 09, September-2019

The first step was to identify and obtain access to a financial ledger from a live ERP system. One of the team members was able to obtain access through his company, since he works in Internal Audit, and with the blessing of his boss, defined the contents of the relevant relations and wrote an SQL query to retrieve the data.

A. Fetching Data from SQL Database

Through the python library, pyodbc [8], the team was able to import the general ledger transactions from Microsoft Dynamics AX ERP [9] system via the ODBC API (Online Database Connectivity Application Programming Interface) to python for analysis.
Below is the SQL (Structured Query Language) query used for importing financial ledger transaction data. In order to use date as a numerical variable in the model, it was converted to an integer format. This query is fetching the journal entries account number, transaction date, posting codes, voucher, who created the journal entry, and the amount of the journal entry. A "WHERE" statement was used to include only journal entries that were posted in the past 60 days, which will allow for model repeatability without having to edit the source code. Reversed journal entries as well as voided entries are outliers that would skew the result. This data was removed since it is not relevant to our test since because voided entries are generally not fraudulent.

```
SELECT
ledgertrans.Accountnum,
cast(ledgertrans.transdate AS INT) AS Date,
ledgertrans.Amountcur,
ledgertrans.Txt,
ledgertrans.Posting,
ledgertrans.LedgerPostingJournalID,
ledgertrans.Voucher,
ledgertrans.Transtype,
ledgertrans.Crediting,
ledgertrans.Createdby,
Userinfo.Name
FROM ledgertrans
LEFT JOIN Ledgertable
ON Ledgertable.AccountNum =
ledgertrans.ACCOUNTNUM
LEFT JOIN UserInfo
ON UserInfo.ID = ledgertrans.createdby
WHERE ledgertrans.transdate > (GetDate () -60) and
ledgertrans.txt NOT LIKE '%Reverse%'
and txt NOT LIKE '%Void%' AND Ledgertable.Dataareaid =
'tel'
ORDER BY
ledgertrans.transdate DESC;
```

B. Using K-means Clustering Algorithm to Analyze Data

K-means clustering is a method of grouping data into K predetermined group based on minimum sum of squared distance between data point and cluster centroid. For this algorithm the base code was structured off of a tutorial found on the Get Data Science course [10]. The non-numeric features that were used in our analysis, createdby, accountnum, and positing, were converted into a matrix in order to be used in the Scikit Learn K-means clustering algorithm. The DictVectorizer feature extraction was used from the sklearn library for this conversion [11]. The numeric features, amountcur, crediting and date, were converted into a matrix using the hstack command from the scipy (parse) library [12]. These two feature matrixes were then combined and scaled using the sklearn scaling command. Trial and error experimentation was conducted to determine the most effective cluster number and the number of times the model will run with different seeds. 25 clusters and 10 runs were decided on as being the most effective. The model was then fitted and predicted.

C. Testing K-means Clustering Algorithm Output Data for Accuracy

The resulting clustered data was put into pandas [13] dataframe, and matched up to the original data set, which included additional information such as voucher numbers, in order to interpret the results in a useful matter on a tuple by tuple basis. The data was then grouped by clusters and counted; and the clusters containing fewer than 20 tuples were extracted for analysis[1]. The results were then exported to an excel spreadsheet on a shared network drive for analysis.
After model testing had been conducted on sample data sets, the full model was run on live production data, and multiple anomalous entries were identified that were deemed worthwhile to investigate. A couple of these anomalous entries were posting by IT Staff, who are not normally authorized to post journal entries. Although a legitimate reason for their entries was found, these sorts of entries are the kind auditors need to efficiently identify to investigate further.

	AccountNum	Date	LedgerPostingJournalID	Name
18656	14100	10/29/15	DAY	Assistant Controller
18657	14100	10/29/15	DAY	Assistant Controller
18692	20250	10/29/15	DAY	Assistant Controller
18693	20250	10/29/15	DAY	Assistant Controller
43131	12202	10/24/15	Prod_78	shopuser
43144	12310	10/24/15	Prod_78	shopuser
49357	50190	10/18/15	AR_24	IT Manager
49608	12190	10/18/15	AR_24	IT Manager
61744	50190	10/7/15	AR_24	Systems Administrator
61745	50112	10/7/15	AR_26	Systems Administrator
61746	50190	10/7/15	AR_26	Systems Administrator
61755	40112	10/7/15	AR_26	Systems Administrator
61757	12190	10/7/15	AR_24	Systems Administrator
61758	12190	10/7/15	AR_26	Systems Administrator
61767	12202	10/7/15	AR_26	Systems Administrator
61768	11120	10/7/15	AR_26	Systems Administrator
67972	12202	10/5/15	Prod_78	shopuser
68462	12310	10/5/15	Prod_78	shopuser
87952	20250	9/28/15	DAY	Assistant Controller
87953	20250	9/28/15	DAY	Assistant Controller
90907	14100	9/28/15	DAY	Assistant Controller
90908	14100	9/28/15	DAY	Assistant Controller

Table 1K-Means Cluster Output File

Posting	Voucher	amountcur	cluster	createdby	crediting
14	023905_GJ	$ 6,486,795.97	22	9873	0
14	023905_GJ	-$ 6,322,089.99	20	9873	1
14	023905_GJ	$ 6,322,089.99	22	9873	0
14	023905_GJ	-$ 6,486,795.97	20	9873	1
112	01312021_078	-$ 1,740.03	19	shop3	1
111	01312021_078	$ 1,740.03	19	shop3	0
60	PAK-074858	-$ 3,625.67	16	9844	1
59	PAK-074858	$ 3,625.67	16	9844	0
60	PAK-074672	-$ 7,445.80	14	9099	1
52	CM-001620	-$ 7,445.80	14	9099	1
60	CM-001620	$ 7,445.80	14	9099	0
51	CM-001620	$ 11,624.80	14	9099	0
59	PAK-074672	$ 7,445.80	14	9099	0
59	CM-001620	-$ 7,445.80	14	9099	1
61	CM-001620	$ 7,445.80	14	9099	0
31	CM-001620	-$ 11,624.80	14	9099	1
112	01306143_078	-$ 0.01	19	shop3	1
111	01306143_078	$ 0.01	19	shop3	0
14	023727_GJ	$ 6,200,208.74	22	9873	0
14	023727_GJ	-$ 4,218,405.29	20	9873	1
14	023727_GJ	$ 4,218,405.29	22	9873	0
14	023727_GJ	-$ 6,200,208.74	20	9873	1

Table 2 K-Means Cluster Output File

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 09, September-2019

V. USING CLASSIFICATION MACHINE LEARNING ALGORITHM TO ANALYZE FRAUDULENT FINANCIAL JOURNAL ENTRIES

For this study, we decided to use two methods: random forest decision trees [14] and support vector machines [15]. These algorithms were chosen due to the difficulty of the classification task: the classes for this task are extraordinarily unbalanced. Random forests have been shown to be effective at dealing with class imbalance and support vector machines were chosen due to their ability to implement multiple kernels (both linear and non-linear).

Using the data source created by one of our team members, a dataset was created which we split into training and test validation sets for the classification task. The original ratio of occurrence of the classes in this dataset is 99.97% not fraud vs. 0.03% fraudulent journal entries (17 out of 58,843 journal entries).

To prepare the data, we normalized the data with sci-kit-learn [16] and removed non-numeric data features. For additional accuracy, we extracted one feature: the number of transactions for each account grouped by credits and debits. This means our remaining features were the amount, whether the journal entry was a credit or a debit, and the transaction count for that account credit/debit combination. After experimenting with different feature options, we realized that the transaction count became very important to prediction:

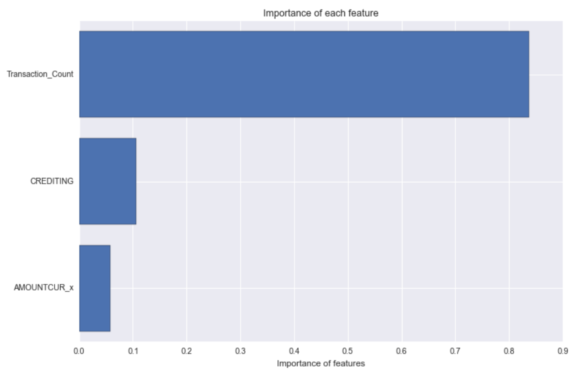

Figure 1 Importance of Features

Initially, with both classification models, accuracy was used to judge the success of our models. It was quickly recognized that this metric was insufficient to meet our needs as the models showed accuracy in Table 3:

A. Comparison of Classification Model Accuracy

Model	Testing Accuracy	Training Accuracy
Random Forest	99.97%	99.93%
SVM Linear Kernel	99.96%	99.99%
SVM Polynomial Kernel	99.96%	99.99%
SVM RBF Kernel	99.96%	99.99%
SVM Sigmoid Kernel	99.96%	99.99%

Table 3 Classification Model Accuracy Table

From the accuracy metric, the model appeared to be performing very well. But to confirm this, a confusion matrix was created for each model type (please note, there was no significant difference among the SVM kernels, so they have been combined into one confusion matrix): Table 4 and 5

Random Forest Confusion Matrix	Non Fraudulent (Predicted)	Fraudulent (Predicted)
Non Fraudulent (Actual)	17638	7
Fraudulent (Actual)	6	2

Table 4 Confusion Matrix Table for Random Forest

SVM Confusion Matrix	Non Fraudulent (Predicted)	Fraudulent (Predicted)
Non Fraudulent (Actual)	17651	0
Fraudulent (Actual)	2	0

Table 5 Confusion Matrix Table for SVM

As can be seen in the confusion matrices, the accuracy for true negatives (i.e. the fraudulent invoices) was not sufficient to be of use in an enterprise setting: 11.76% for random forests and 0% for the SVM. After doing additional research [4,5,6] it was decided to attempt to rebalance the classes. It was theorized by balancing the incidence of each class, it would enable the predictor to more accurate predict which class each journal entry belonged to.

We also discovered that it was possible to weight each class in the random forest models and gave the classes a 0.001 weight to non-fraudulent class to a 1000 weight to the fraudulent case.

The dataset was sampled to a 50% occurrence rate for each class and this dramatically increased the accuracy for the random forest, but lowered it for the SVM:

B. Comparison of Classification Model Accuracy after Balancing Incidence of Each Class

Model	Testing Accuracy	Training Accuracy
Random Forest	95.83%	100.00%
SVM Linear Kernel	91.67%	81.82%
SVM Polynomial Kernel	100.00%	81.82%
SVM RBF Kernel	91.67%	72.73%
SVM Sigmoid Kernel	75.00%	81.82%

Table 6 Comparison of Classification Model Accuracy after Balancing

However, there was an increase for both models in terms of predicting fraudulent invoices accurately: Table 7 and 8

Random Forest Confusion Matrix	Non Fraudulent (Predicted)	Fraudulent (Predicted)
Non Fraudulent (Actual)	5	0
Fraudulent (Actual)	0	6

Table 7 Confusion Matrix Table for Random Forest after Balancing

SVM Confusion Matrix	Non Fraudulent (Predicted)	Fraudulent (Predicted)
Non Fraudulent (Actual)	**6**	**1**
Fraudulent (Actual)	**1**	**3**

Table 8 Confusion Matrix Table for SVM after Balancing

The Random Forest Model increased to 100%, which suggests possible over-fitting, and the SVM reached 75% false negative accuracy.

In conclusion, we found that the ability to weight the classes made a significant difference for the random forest algorithm and rebalancing the classes increased the accuracy for the support vector machines. It is our recommendation that the random forest algorithm be used in an enterprise setting, as it had much improved accuracy over each kernel of the SVM implementation.

VI. RECOMMENDATION

The Clustering and Classification algorithms tested provided encouraging results towards implementing machine learning for journal entry fraud testing. Based strictly on the results performed, we found the unsupervised K-means algorithm as being superior due to ability to pick out anomalous entries without requiring significant computing power. The SVM algorithm showed potential however, and we believe it should be explored more fully in future research.

VII. CONCLUSION

The use of machine learning for journal entry fraud testing is a scarcely explored area that produces high quality results for anomalous entries. The ability to succinctly download financial journal entries from an ERP system's underlying relational database is fairly straightforward and provides unlimited opportunities for further analysis. By utilizing the diverse and powerful Python libraries, we were able to create three separate machines learning models that were performed on journal entries downloaded via an ODBC connection from a live production database. The analysis produced high quality results that warrant further research into applying machine learning, specifically K-means clustering, to journal entry fraud testing.

REFERENCES

[1] Miklos A.Vasarhelyi (2004), Michael G.Alles (2004), Alexander Kogan (2004). Principle of Analytic Monitoring for Continuous Assurance. Journal of Emerging Technologies in Accounting [Online]. Available: http://raw.rutgers.edu/docs/research/M31.%20principles%20of%20analytic%20mont.pdf

[2] Scikit-Learn Developers (2012 June). Scikit-Learn User Guide [Online]:http://www.math.unipd.it/~aiolli/corsi/1213/aa/user_guide-0.12-git.pdf

[3] Anuj Sharma (2012 Feb.), Prabin Kumar Panigrahi (2012 Feb.). A Review of Financial Accounting Fraud Detection based on Data Mining Techniques. International Journal of Computer Applications (0975 – 8887) [Online]. Available: http://arxiv.org/pdf/1309.3944.pd

[4] Argyrou, Argyris. Auditing Journal Entries Using Extreme Value Theory. Presented at Proceedings of the 21st European Conference on Information Systems [Online]. Available: http://www.staff.science.uu.nl/~vlaan107/ecis/files/ECIS2013-0848-paper.pdf

[5] SAS No. 99; SAS No. 113(2002 Feb.). Consideration of Fraud in a Financial Statement Audit. [Online]: http://www.aicpa.org/Research/Standards/AuditAttest/DownloadableDocuments/AU-00316.pdf

[6] Center for Audit Quality (2008 Dec.). Practice Aid for Testing Journal Entries and Other Adjustments Pursuant to AU Section 316 [Online]: http://www.thecaq.org/docs/practice-aid/caqjetestingpracticeaid12082008.pdf?sfvrsn=2

[7] Richard B. Lanza (2015 Oct.), Scott Gilbert (2015 Oct.). A Risk-Based Approach to Journal Entry Testing [Online]: http://www.journalofaccountancy.com/issues/2007/jul/ariskbasedapproachtojournalentrytesting.html

[8] Python Development Tools (2015 Nov). Pydoc User Guide [Online]: https://docs.python.org/2/library/pydoc.html

[9] Microsoft's ERP solution for enterprises (2015). [Online]: https://www.microsoft.com/en-us/dynamics/erp-ax-overview.aspx

[10] Microsoft's ERP solution for enterprises (2015). [Online]: http://www.gadatascience.com/modeling/kmeans.html

[11] Scikit-Learn Developer (2012 June) Developers (2012 June). Scikit-Learn User Guide [Online]: http:// http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.DictVectorizer.html

[12] SciPy V0.16.1 Reference Guide (2015 Oct) Developers (2012 June). Scikit-Learn User Guide [Online]:http://docs.scipy.org/doc/scipy-0.16.0/reference/generated/scipy.sparse.hstack.html

[13] Python Data Analysis Library V 0.17.1 (2015 Nov) [Online]: http://pandas.pydata.org/

[14] 3.2.4.3.1.sklearn.ensemble.RandomForestClassifier V 0.17 [Online]:http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier

[15] 1.4. Support Vector Machines V0.17 [Online]:http://scikit-learn.org/stable/modules/svm.html

[16] Sklearn preprocessing normalize V0.17 [Online]:http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html