Vivekananda Journal of Research
An official Journal of **Vivekananda Institute of Professional Studies-TC**, Delhi, India

ORIGINAL RESEARCH ARTICLE
Published on 15 September, 2026
www.vjr-vips.com

Vendor Invoice Intelligence System using Machine Learning for Cost Prediction and Risk Detection

Shivam, Mayank, Nikita Malik

Department of Computer Applications, Maharaja Surajmal Institute, Delhi, India

ABSTRACT

The tracking of vendor invoices in large organizations requires the handling of thousands of invoices across multiple vendors, purchase orders, and deliveries. Manual record keeping not only is tedious but also can lead to errors, making it difficult for the finance team to detect any anomalies that pose a threat. The design, implementation, and deployment of this project include developing a Vendor Invoice Intelligence System powered by machine learning techniques for automatically predicting invoices' freight costs and detecting potentially risky invoices. In order to accomplish that, the authors implement this system by using the programming language Python along with a relational SQLite database made up of five interrelated tables with more than one million records in total. Next, exploratory data analysis is performed, features are extracted from the raw data such as the freight per unit, delay in delivery, and mismatch between invoices at the dollar level, and regression and classification models are trained on that dataset. As for the model that yields the highest accuracy for the task of predicting invoices' freight costs, after comparing three algorithms, Linear Regression emerges as the winner with an R^2 score of 96.99 percent, Mean Absolute Error of 24.11, and Root Mean Squared Error of 124.72. In terms of invoice risk detection, after comparing three different classification algorithms, Random Forest is the model that performs the best with an accuracy of 88 percent and a weighted F1 score of 0.87. Finally, both tasks are deployed on an interactive Streamlit web application in which invoices' information can be inputted and analyzed in real-time regardless of the user's technical background.

ARTICLE INFO

DOI:
10.61081/vjr/16v1i111

Correspondence
Nikita Malik
nikitamalik@msijanakpuri.com

Keywords
Vendor invoice intelligence, Freight cost prediction, Invoice risk detection, Machine Learning, Random Forest, Decision Tree, Streamlit

How to cite:
Shivam, Mayank, Malik N. (2026). Vendor Invoice Intelligence System using Machine Learning for Cost Prediction and Risk Detection. Vivekananda Journal of Research, 16(1), 97–108

INTRODUCTION

In an organizational setting characterized by frequent transactions with multiple vendors, handling invoices presents a particularly challenging task for the finance department. A number of activities take place during a procurement cycle, starting from ordering to receiving the goods and issuing a corresponding

© Authors 2026. Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) License, which allows users to download and share the article for non-commercial purposes, so long as the article is reproduced in the whole without changes, and the original authorship is acknowledged. If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. If your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit https://creativecommons.org/licenses/by-nc-sa/4.0/

invoice, followed by a payment process. In each of these processes, there is a possibility of making errors, ranging from differences between the total amount invoiced and the actual purchase amount to delayed delivery of the goods and high freight charges without any justification. Cumulatively, these errors lead to invoice cost leakage, an occurrence in which money is lost through undetected mistakes during the billing process (Abdallah *et al*., 2016)..

The data in use is sourced from the.sqlite database with five tables: begin_inventory, end_inventory, purchase_prices, purchases, and vendor_invoice. Collectively, these tables provide information about the procurement process from the perspective of a retailer that orders branded goods from multiple vendors. In this case, vendor_invoice is the focal table since it contains data on invoice quantities, invoice amount, freight charges, and payment timeline.

As stated earlier, the authors analyze the invoice dataset in two ML-based projects. First, the authors perform regression analysis to estimate the freight charge for an invoice using such features as quantity ordered and invoice total dollar amount. Second, the authors apply classification techniques to identify potentially problematic invoices with regards to their total costs and delivery timelines. An invoice is treated as requiring a further audit if there is a discrepancy between the invoice-level total and the total amount of individual items exceeding five dollars and the average delivery time exceeding ten days.

The final implementation is done via email Streamlit, a web application that enables users to upload invoice data and receive predictions on the cost and risk classification using the trained models, with no requirement to retrain them or any code writing involved. Models are saved as .pkl files for quick loading upon startup.

LITERATURE REVIEW

Applications of Machine Learning for Analyzing Financial Documents

There has been extensive research into the use of machine learning algorithms and data mining techniques to process financial documents and detect fraud in recent years (Ngai *et al*., 2011). While the earliest studies on this topic focused on rule-based algorithms and OCR to extract data fields from invoice documents, research focus has gradually moved towards identifying patterns in the now ubiquitous structured transactional data.

Zheng *et al*.'s (2018) analysis of fraud detection problems and the application of ensemble machine learning models reveal that ensemble approaches such as Random Forest perform reliably better than any single classifier when faced with financial fraud issues. It aligns with the authors' experience, during which random forest proves superior to logistic regression and decision tree for flagging high-risk invoices. analysis of fraud detection problems and the application of ensemble machine learning models reveals that ensemble approaches such as Random Forest perform reliably better than any single classifier when faced with financial fraud issues. It aligns with the authors' experience, during which Random Forest proves superior to Logistic Regression and Decision Tree for flagging high-risk invoices.

In their thorough review of methods used for fraud detection, Abdallah *et al.* (2016) emphasize that class imbalance is a constant challenge for researchers in this field, noting that in nearly all data sets, there are substantially fewer transactions of a fraudulent/risky nature than the number of legitimate transactions. This is certainly relevant to this problem, although the current dataset is somewhat less imbalanced, with about one in every three invoices being flagged. Their proposed solutions include the use of ensembles and setting different threshold values for increasing sensitivity.

Linear models prove extremely useful for extracting signals in structured procurement data, according to Carbonneau *et al.* (2008) work on demand prediction in supply chains. The same pattern can be observed in this project with regard to the regression model, which performs significantly better compared to both of the tree-based classifiers, mainly because the connection between the freight and the invoice is inherently linear.

SMOTE is another contribution by Chawla *et al.* (2002), which aims to address class imbalance by generating artificial instances of minority classes. Since SMOTE can be used to generate additional

minority classes, it would effectively increase the value of recall in predicting invoices that should be flagged – something that will be implemented in future iterations of this algorithm. Raschka & Mirjalili (2019) and Geron (2019) agree that making trained models easily accessible to non-technical users is vital agree that making trained models easily accessible to non-technical users is vital.

Problem Statement

Finance departments in organizations where procurement is prevalent handle vast numbers of invoices from vendors that require careful evaluation and verification to determine their legitimacy for payment. In most companies, the evaluation of invoices requires a lot of manual intervention, creating three primary concerns.

Firstly, overpayments in regard to transportation expenses usually go unnoticed since the cost incurred by each transport service is relatively small compared to the total bill and the person evaluating the invoice lacks knowledge about the expected transportation cost based on the nature and amount of the goods. Secondly, invoices containing discrepancies such as total amounts not being consistent with the total amount of itemized charges or delivery time being significantly delayed might be approved despite the issues due to the high number of transactions to assess. Thirdly, since there is no way of assessing financial risks associated with each invoice, the auditing process cannot be effectively prioritized. These challenges are overcome by the creation of a machine learning model that estimates the expected transport costs and classifies the invoice in terms of risk levels.

RESEARCH OBJECTIVES

The goals of this research are as follows. First, the study aims to conduct exploratory data analysis on a five-table relational invoice dataset to gain insights into vendor behavior, freight cost patterns, and delivery timelines. Next, the goal is to create features by combining raw data from different tables, including information about the ratio of freight as well as other measures relating to delivery delays. In the third step, several regression models are created and tested for forecasting the cost of freight and the one that exhibits better performance is selected. In the fourth step, several classification models are created for identifying invoice risk with a distinct binary risk label that can be created through differences in dollars and delivery delays. Finally, the trained models are deployed as an interactive Streamlit app.

Dataset Description

This data set is extracted from the project initiated by Mishra (2026), and it is provided in a database structure called SQLite (.db) format which consists of five relational tables covering the entire life cycle of the purchase and storage of branded items by a retail company. These tables are imported in Python using the sqlite3 package and later transformed to a Pandas DataFrame (Pedregosa *et al*., 2011). The dataset is publicly available on Kaggle (Mishra, n.d.).

Data Analysis and Feature Engineering

1. Regression Preprocessing — Loading the Data

For the regression task, the vendor_invoice table is loaded directly from the SQLite database using a single SQL query and convert it straight into a Pandas DataFrame:

vendor_df = pd.read_sql_query("SELECT * FROM vendor_invoice", con)

This table contains the required variables – Quantity, Dollars, and Freight – therefore, joining other tables is unnecessary. In addition, after loading the data into a Pandas dataframe, .head(), .info(), and .describe() functions are called to check the data for missing values within these three columns.

2. Correlation Analysis for Regression

To determine the relationship between Quantity, Dollars, and Freight, their pairwise correlations are calculated, and a heatmap is plotted. The precise correlations are given in figure 1 below.

From this heatmap, one can see that the correlations between any two variables are extremely high and positive. For instance, the correlation between dollars and freight is 0.99; hence, dollars are nearly a perfect predictor of the freight of an invoice. The correlation between quantity and freight is 0.95, while the correlation between quantity and dollars is 0.96. These numbers confirm the



Figure 1: Correlation Heatmap — Quantity, Dollars, and Freight (vendor_ invoice table).

linear association between any two variables in this dataset, thereby explaining the superior performance of the linear regression model in this problem. Multicollinearity between quantity and dollars is expected and should be expected in most cases when dealing with monetary values. However, since the primary concern is the accuracy of predictions, using both features is justified.

3. Scatter Plot Analysis

For visual verification, a double scatter plot is created for both quantity and dollars as a function of freight, as shown below in figure 2.

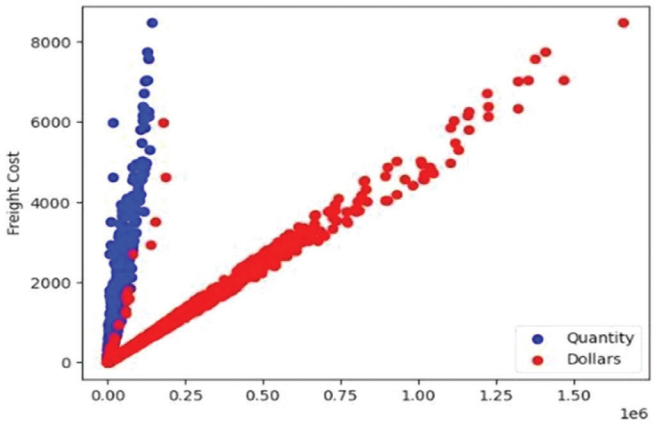


Figure 2: Scatter plot — quantity vs. freight (blue) and dollars vs freight (red).

The two plots show completely different trends. Looking at the blue points (quantity vs. freight), you can see a vertical cluster on the left. This means most orders have low quantities, but their freight costs vary wildly, from near zero up to about $8,000. This pattern exists because most invoices are for small quantities, yet the actual freight charges change drastically depending on the supplier and product type.

In the second plot (dollars vs. freight), there is a clear diagonal line of data points stretching from the bottom-left to the top-right, covering invoice values up to roughly $1.5 million. The takeaway here is the extremely strong correlation between freight cost and the invoice's total value—they move almost perfectly together along that diagonal line (James *et al*., 2021).

4. Regression line visualisation

Figure 3 shows the linear fit between dollars and freight directly, with a red regression line overlaid on the scatter of individual data points.

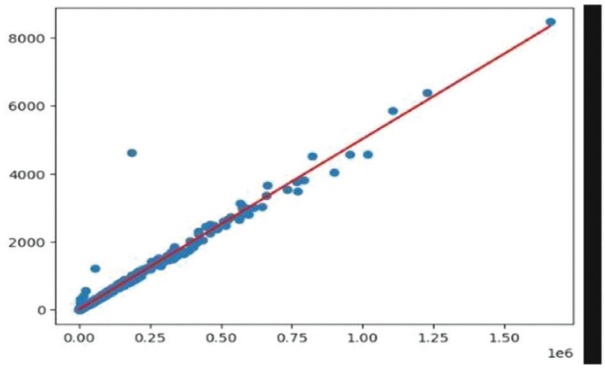


Figure 3: Linear regression fit — Dollars vs. freight with fitted regression line.

The fitted regression line—the solid red line—demonstrates an excellent linear fit, perfectly tracking the data from the origin up to the highest values (around $1.5 million in total dollars and $8,500 in freight). While the fit is strong, scattered outliers are observed, particularly in the middle range of invoice values, deviating noticeably above or below the trend line. These points represent invoices where the actual freight cost was abnormally high or low relative to the expected amount, often suggesting special arrangements, urgent needs, or data entry errors. The presence of these larger deviations explains why the Root Mean Squared Error (RMSE of 124.72) is significantly higher than the Mean Absolute Error (MAE of 24.11), as RMSE disproportionately penalizes these larger, influential errors.

5. Feature engineering – Freight per unit

Once the relationships are verified to be linear visually, the study proceeds with feature engineering by creating the following important feature:

vendor_df['freight_per_unit'] = vendor_df['Freight'] / vendor_df['Quantity']

The freight_per_unit metric normalizes the total shipping charge by the number of units in an order, giving us a standardized per-item cost. This makes it much easier to compare shipping efficiency across orders of all different sizes. For instance, a large order with a high total freight cost might actually be very efficient per unit, whereas a small order with a low total cost could end up being disproportionately expensive per item. By including this normalized perspective, the regression model gains a more stable feature that better captures the actual efficiency and economics of freight pricing.

6. Outlier Detection Using Interquartile Range

To understand how freight efficiency changes with order size, the first and third quartile thresholds for quantity have been computed:

low_quantity = vendor_df['Quantity'].quantile(0.25) # Result: 83.0

high_quantity = vendor_df['Quantity'].quantile (0.75) # Result: 5100.5

The mean freight_per_unit is then computed separately for orders below Q1 and above

Q3: vendor_df.loc[vendor_df['Quantity'] < low_ quantity, 'freight_per_unit'].mean() # 0.0949

vendor_df.loc[vendor_df['Quantity'] > high_ quantity, 'freight_per_unit'].mean() # 0.0491

Table 2: Freight per unit by order size group.

Order Size Group	Quantity Boundary	Mean Freight Per Unit ($)
Small orders (bottom 25%)	Less than 83 units	0.0949
Large orders (top 25%)	More than 5,100.5 units	0.0491

According to the findings, small orders are characterised by an average_freight cost of approximately 9.49 cents per unit, compared to 4.91 cents per unit for larger ones. As one may note, small orders are almost two times more expensive to deliver because of the flat fee part of the shipping price. It happens due to the fact that handling, documentation, and other fees involved in the freight process cannot be adjusted to the size of an order, making them more burdensome for smaller deliveries. Therefore, from a business perspective, this insight suggests that consolidation of small orders may help reduce the_freight_per_unit cost.

7. Classification Preprocessing — SQL-Based Feature Construction

For the classification task, a much richer feature set is needed that combines invoice-level data with purchase-level aggregations. Rather than loading and merging tables separately in Python, the entire integration step is performed inside a single SQL query using a Common Table Expression that aggregates the purchases table by PONumber before joining it to vendor_invoice:

```
df = pd.read_sql_query("""
WITH purchase_agg AS (
SELECT p.PONumber,
count(distinct Brand)  as total_brands,
sum(p.Quantity)     as total_item_quantity,
sum(p.Dollars)      as total_item_dollars,
avg(julianday(p.ReceivingDate)-julianday(p.PODate)) as avg_receving_delay
FROM purchases p GROUP BY p.PONumber
)
SELECT vi.Quantity as invoice_quanity, vi.Dollars as invoice_dollars,
vi.Freight,
(julianday(vi.InvoiceDate)-julianday(vi.PODate))  as days_to_POInvoice,
(julianday(vi.PayDate)-julianday(vi.InvoiceDate))  as days_to_pay,
pa.total_item_quantity, pa.total_brands,
pa.total_item_dollars, pa.avg_receving_delay
FROM vendor_invoice vi
LEFT JOIN purchase_agg pa ON vi.PONumber = pa.PONumber""", con)
```

SQLite's julianday() function converts date strings into continuous day numbers, making

day-difference calculations straightforward and accurate. The LEFT JOIN preserves all invoice records even if a matching purchase order is not found in the aggregation.

8. Risk Label Creation

The binary classification target called flag_invoice is created using a function that applies two business-grounded conditions:

```
def create_invoice_risk_label(row):
if abs(row['invoice_dollars'] - row['total_item_dollars']) > 5:
return 1  # Dollar mismatch detected
if row['avg_receving_delay'] > 10:
return 1  # Abnormal receiving delay
return 0
df['flag_invoice'] = df.apply(create_invoice_risk_label, axis=1)
```

An invoice is marked as risky (1) if the absolute difference between the invoice-level and item-level dollar totals exceeds five dollars, indicating a possible billing discrepancy, or if the average number of days taken to receive goods in that purchase order exceeds ten days, indicating a logistical irregularity. Invoices that pass both checks are labeled as normal (0).

9. Risk Label Distribution

After applying the labeling function, the distribution of the target variable is visualised, shown in figure 4.

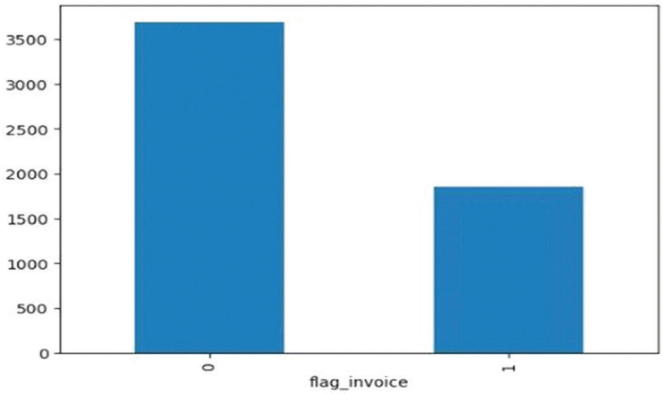


Figure 4: Distribution of flag invoice — Class 0 (Normal) vs Class 1 (Flagged/Risky).

The chart shows approximately 3,700 normal invoices (Class 0) and approximately 1,900 flagged invoices (Class 1), giving a class ratio of roughly 66 percent normal to 34 percent risky. This is a moderate imbalance that is workable for standard classification algorithms. The fact that about one in three invoices is flagged under these conditions is itself an important business insight: a significant proportion of vendor invoices in this dataset contain either a dollar discrepancy or a delivery delay that warrants investigation.

10. Full Feature Correlation Heatmap

The full correlation matrix across all nine features and the target variable is computed and visualized, as shown in figure 5.



Figure 5: Full Correlation Heatmap — All Classification Features and Target Variable.

The heatmap supports several crucial correlations. Dollars and Freight correlate to 0.99, as seen in the regression analysis performed earlier. Of all the variables in the data set, avg_receving_delay turns out to be the strongest predictor for flag_invoice with a linear correlation value of 0.31. There exists a moderate correlation between days_to_POInvoice and flag_invoice at a value of 0.18 while days_to_pay does not correlate with flag_invoice at all with a linear correlation value close to zero of -0.005. There also exists a positive correlation of 0.52 between days_to_POInvoice and avg_receving_delay, which makes sense because when goods reach late, the invoices will take longer time as well.

11. Statistical Feature Significance t-test

Before finalising the features for the classification model, Welch's independent-samples' t-test was performed from scipy.stats to determine which features show a statistically meaningful difference in their mean values between flagged and normal invoices:

```
from scipy.stats import ttest_ind

t_stat, p_value = ttest_ind(flagged[metric].dropna(),

normal[metric].dropna(), equal_var=False)
```

Welch's formulation has been used, which does not assume equal variances between groups, making it more appropriate for financial data where variance often differs between risk categories. The results are shown in table 3.

Table 3: Welch's t-test results — Feature Significance by Invoice Risk Class.

Feature	Flagged Mean	Normal Mean	*p-value*	Significant?
invoice_ quantity	6728.28	5723.55	0.021	Yes
invoice_dollars	65600.61	54302.64	0.008	Yes
Freight	334.02	276.89	0.008	Yes
days_to_ POInvoice	17.23	16.02	0.000	Yes
total_item_ quantity	6728.28	5723.55	0.021	Yes
total_item_ dollars	65600.61	54302.64	0.008	Yes
avg_receving_ delay	8.47	7.27	0.000	Yes
days_to_pay	35.42	35.49	0.692	No
total_brands	42.29	40.82	0.508	No

Seven of the nine features pass the $p < 0.05$ significance threshold. The two that do not are days_to_pay and total_brands. The days_to_pay means are 35.42 and 35.49 for flagged and normal invoices respectively, a difference of less than one day that is clearly random variation. The total_ brands means differ by just 1.47, and the *p-value* of 0.508 confirms no meaningful signal. These two features are identified as non-significant and are treated as weak predictors for the classification model. The seven significant features form the discriminating input space, with avg_receving_ delay and days_to_POInvoice being the strongest temporal signals of invoice risk.

MACHINE LEARNING METHODOLOGY

Data Splitting

For both the regression and classification tasks, the data is split into training and testing sets using an 80/20 ratio through train_test_split from Scikit-learn with a fixed random state to make results reproducible, following standard data science practices (Kotu & Deshpande, 2019). This means 80 percent of the records are used to train each model, and the remaining 20 percent are held out entirely for evaluation.

Regression Task

Freight is the dependent variable used in the regression model. The independent variables include Quantity, Dollars, and freight_per_unit. In total, three regression models are created using the same training data set and evaluated using the same test data set using MAE, RMSE, and R-squared (James *et al*., 2021).

Classification Task

The classification is done with respect to the flag_ invoice variable. There are seven statistically significant predictor variables selected by t-tests used in the training data set, including invoice_quantity, invoice_ dollars, Freight, days_to_POInvoice, total_item_ quantity, total_item_dollars, and avg_receiving_delay. Three classifiers are built, and their performance is evaluated based on accuracy, precision, recall, and F-score. The performance measures include those of all classes, macro average, and weighted average (James *et al*., 2021).

ALGORITHM EXPLANATION

Linear Regression

Linear regression fits a straight line (or a hyperplane when multiple predictors are involved) to model the dependency of a continuous outcome on one or more input variables by minimizing the total squared error between observed and fitted values. Each input feature receives a weight that quantifies how the predicted outcome shifts for every unit to change in that feature. The method is computationally lightweight and straightforward

to interpret, though it presupposes a linear dependency structure between the inputs and the target. In this experiment, the heatmap and scatter plot both reveals pronounced linear associations in the data, which explains why the Linear regression model outperforms the tree-based alternatives on freight cost prediction (James *et al*., 2021).

Decision Tree

The decision tree divides the dataset based on the thresholds of the inputs, stepwise, which is a foundational technique in data mining (Han *et al*., 2006). In regression tasks, the decision tree aims to decrease the variance of the target variable. For classification tasks, decision trees can split based on either Gini impurity or entropy measures. The splitting continues until specific conditions (maximum depth, for instance) are met. As a result, a comprehensible hierarchy is built. Without any regularizations, however, there is a risk of overfitting (James *et al*., 2021).

Random Forest

Random Forest is an ensemble technique that constructs a large collection of decision trees, each trained on a bootstrapped sample of the data and restricted to considering only a random subset of features at every split. Final predictions are produced by averaging the outputs of all individual trees in regression settings, or by taking the class with the most votes in classification settings. Introducing randomness at both the sample and feature levels substantially lowers the variance that a single tree would exhibit, resulting in stronger out-of-sample generalization (Chen & Guestrin, 2016). As an additional benefit, the algorithm exposes a feature importance ranking derived from the cumulative contribution of each variable across all trees (Breiman, 2001).

Logistic Regression

Logistic regression is a linear classification method that models the probability of a binary outcome by passing a weighted combination of input features through the logistic sigmoid function, which maps any real-valued input to a value between zero and one. That probability is then thresholded to produce a class prediction. Although its name mirrors that of Linear Regression, the two serve entirely different purposes: one estimates a continuous quantity while the other decides between categories. In this study, Logistic regression is included as a linear benchmark whose results can be contrasted with those of the nonlinear, tree-based methods to gauge how much the decision boundary complexity matters for invoice risk detection (James *et al*., 2021).

MODEL TRAINING AND HYPE-RPARAMETER TUNING

Regression Training

All three regression models are trained using Scikit-learn's default implementations on the 80 percent training split. For the Random Forest Regressor, 100 estimators are used as the default setting. After comparing evaluation metrics across all three models on the held-out test set, Linear regression is identified as the best performer and serialised as a .pkl file using joblib for use in the Streamlit application.

Classification Training and Hyper-parameter Optimisation

For the classification task, Logistic regression, Decision Tree Classifier, and Random Forest Classifier are trained using Scikit-learn defaults in the first round. After the initial comparison confirms that Random Forest performs best, GridSearchCV is applied to search over a grid of hyperparameter combinations including the number of estimators, maximum tree depth, and the minimum number of samples required to split a node or populate a leaf. 5-fold cross-validation is used within the training set throughout the grid search to ensure that no test data leaks into the tuning process. The best-performing Random Forest configuration identified by the grid search is then serialised as a .pkl file for deployment.

EXPERIMENTAL RESULTS AND PERFORMANCE COMPARISON

Regression Results

Table 4 presents the evaluation of metrics for all three regression models on the held-out test set.

Table 4: Regression Model Performance Comparison.

Model	MAE	RMSE	R^2 Score (%)
Linear Regression	24.11	124.72	96.99%
Decision Tree Regressor	38.12	138.25	96.30%
Random Forest Regressor	26.13	134.79	96.48%

All three models achieve very high R^2 scores, confirming that the selected features are strongly predictive of freight cost. Linear Regression achieves the best result across (Carbonneau *et al*., 2008) the lowest MAE of 24.11 means its predictions are on average within approximately 24 dollars of the actual freight charge, and its RMSE of 124.72 is the lowest among the three models, indicating that it also handles the outlier invoices with high freight values more consistently than the tree-based approaches. The fact that a simple linear model outperforms both tree-based alternatives confirms that the relationship between invoice attributes and freight cost is fundamentally linear in this dataset, consistent with the 0.99 correlation between Dollars and Freight observed in the exploratory analysis.

Classification Results

Table 5 presents the evaluation of metrics for all three classification models on the held-out test set, which contains 1,109 records in total.

Table 5: Classification Model Performance Comparison.

Model	Accuracy	Precision (Class 1)	Recall (Class 1)	F1 (Class 1)	Weighted F1
Logistic Regression	66%	0.56	0.06	0.11	0.55
Decision Tree Classifier	81%	0.74	0.71	0.72	0.81
Random Forest Classifier	88%	0.94	0.68	0.79	0.87

The classification results show a clear and consistent improvement as model complexity increases. Logistic Regression achieves only 66 percent accuracy and essentially fails to detect risky invoices, with a Class 1 recall of just 0.06. This means it correctly identifies fewer than 1 in 10 flagged invoices, which would be practically useless for a risk detection system. The linear decision boundary Logistic Regression relies on is simply not expressive enough to capture the patterns that distinguish risky invoices from normal ones in this feature space.

The Decision Tree Classifier improves substantially to 81 percent accuracy and a Class 1 F1 score of 0.72, correctly identifying 71 percent of risky invoices while maintaining a precision of 74 percent. This is already a workable classifier for practical use, though the Random Forest does better on every metric.

The Random Forest Classifier achieves 88 percent accuracy and a weighted F1 score of 0.87. Its Class 1 precision of 0.94 is particularly strong: when it predicts that an invoice is risky, it is correct for 94 percent of the time. Its Class 1 recall of 0.68 means it misses about 32 percent of genuinely risky invoices, but this precision-recall profile is well-suited to an audit support context where false alarms are costly and the goal is to give reviewers a short, reliable list of high-priority cases.

Table 6: Detailed Classification Report — Random Forest Classifier.

Class	Precision	Recall	F1 Score	Support
0 (Normal)	0.85	0.98	0.91	725
1 (Flagged/ Risky)	0.94	0.68	0.79	384
Macro Average	0.90	0.83	0.85	1109
Weighted Average	0.88	0.88	0.87	1109

System Architecture

Vendor Invoice Intelligence Pipeline is made up of four layers. The first layer comprises the data layer, which consists of five tables of SQLite database containing all the raw data regarding procurement and stock. The second layer includes preprocessing and feature engineering, which involves the use of python scripts for data loading, dates parsing, merging of tables using SQL, handling of missing values, and generating new features. The third layer includes the model layer, which contains

the trained regression and classification models as .pkl files. The fourth layer entails the application layer, which uses the Streamlit application as an interface for interaction.

Data flow starts from the database, followed by processing and feeding into the notebook used for training models. After training of models and saving in the computer, the Streamlit application loads the saved models from the .pkl files and makes predictions based on them. This separation between training pipeline and application guarantees promptness of application without any need for further training.

Table 7: Technology Stack Summary.

Component	Technology Used	Purpose
Programming Language	Python 3.x	Data processing, ML, and deployment
Database	SQLite (.db)	Relational storage of five tables
Data Manipulation	Pandas, NumPy	DataFrames, feature engineering
Visualisation	Matplotlib, Seaborn	EDA charts, heatmaps, scatter plots
Machine Learning	Scikit-learn	Model training, evaluation, GridSearchCV
Statistical Testing	SciPy (ttest_ind)	Welch t-test for feature significance
Model Serialisation	Joblib (.pkl files)	Saving and loading trained models
Web Application	Streamlit	Interactive browser-based deployment
Development Env.	Jupyter Notebook	EDA, experimentation, analysis

Streamlit Application Deployment

The Streamlit application provides an interface accessible through the browser that allows for deploying the machine learning models without requiring coding skills. At the moment of execution, the application loads the files with the regression and the classification models using joblib.load() function.

The sequence of actions on the part of the user consists of the following steps. Firstly, the application launches, and users access it through their web browsers. Secondly, users fill out an invoice form. This can be accomplished by uploading a CSV file with invoice data or providing data themselves by filling in the quantity field, entering the monetary value, and specifying the relevant dates. Thirdly, after clicking on the 'Run' button, the application uses the same Python functions to calculate the freight_per_unit, days_to_pay, avg_receiving_delay, and any other derived features just as was done during training.

After this, two results are provided by the application. First of all, the estimated freight value is displayed together with a description of how typical this value is compared with historical data. This makes interpretation easier. In addition, the second result is the classification result (Normal or High Risk) together with the probability of belonging to the specified class.

Finally, the application includes a graphical panel with bar charts and scatter plots that visualize the patterns in the uploaded data set. In particular, the distribution of freight charges and proportion of invoices classified as risky are shown in this panel. As all output changes in response to the inputs, the application is truly interactive and useful for a review session.

DISCUSSION

Findings

In terms of technical considerations, the main conclusion drawn is that Linear Regression outperforms both tree-based regressors on prediction of freight cost, despite the fact that this model is relatively simple when compared to two other regression algorithms (Carbonneau *et al*., 2008). Such behavior is supported by exploratory findings: correlation value of 0.99 between variables of Dollars and Freight as well as presence of clear diagonal band in a scatter plot clearly indicates the dominance of linear dependency in this problem domain. Hence, introduction of flexibility provided by a Decision Tree as well as ensemble averaging used in case of a Random Forest neither improves predictive power nor lowers variance of such predictors. Instead, this may even add slight instability around the level of explainable variance.

As far as classification results are concerned, a major difference between logistic regression models

and those based on tree algorithms provides additional insights. Specifically, it should be noted that while Logistic Regression shows recall rate of just 0.06 for Class 1 with overall accuracy of 66 percent, the latter metric alone might be rather misleading for moderately unbalanced datasets in question that tend to output mostly majority class predictions. Tree-based algorithms, being able to find nonlinear interactions within features, perform significantly better in this scenario, with Random Forest obtaining Class 1 precision of 0.94. As a result, this can be translated into meaningful business applications: audit flags produced by this algorithm can be trusted with a high level of certainty.

Finally, the use of the dual condition approach to risk label generation proves to be effective and easily interpretable. With no need for experts' input and historical audit database in place, t-test shows the effectiveness of such a labeling scheme, which is generalized by model with excellent results throughout the entire dataset. It should be noted that, once more, this method is independent of any machine learning algorithms applied in this research.

In terms of business value, the finding concerning IQR-based freight efficiency is especially interesting, providing the idea for action that doesn't rely on machine learning results at all. Namely, small orders have almost twice the shipping cost per unit than large orders.

Limitations

Several limitations of this study should be mentioned. The first limitation concerns the creation of risk labels using a function based on certain rules rather than audit results. While the risk labeling process is based on sound reasoning related to financial activities, there might be cases when some flagged risky invoices are actually legitimate ones, and vice versa – some invoices labeled as normal have errors that do not fall under the listed criteria.

The second limitation associated with the current model refers to the source of the dataset that can be considered industry specific as far as its characteristics are concerned. Thus, models trained with this dataset are unlikely to work well with other types of industries and their peculiarities without training on a new dataset.

A third limitation concerns class imbalance since about 34 percent of all invoices are considered risky. As a consequence, the recall of the model is limited in terms of the minority class. There are no efforts to balance classes via such approaches as, for example, SMOTE.

Finally, the Streamlit app is not developed in order to use it in practice and is used to present the solution created. Thus, no authentication and encryption are included in the prototype.

CONCLUSION

This research provides an introduction to the Vendor Invoice Intelligence System, which represents a full-cycle machine learning project for predicting the cost of goods shipped and detecting potentially risky invoices. Starting from a raw SQLite database containing five relational tables with more than one million rows of data altogether, thorough exploratory data analysis is performed, domain-oriented features are designed by aggregating and transforming data with SQL scripts and Python packages, and statistical significance testing is conducted to select the best set of predictive variables. Various linear and tree-based regression and classification models are then used to predict the total shipment price and the level of invoice risk based on the chosen features. The results show that Linear Regression outperforms other regression methods (R^2 = 96.99%, MAE = 24.11), and Random Forest outperforms others (88% accuracy, high-risk class precision of 94%). The developed software product is implemented as a Streamlit application capable of providing end users with real-time predictions of freight cost and invoice risk levels.

There are several possible directions that can be pursued to extend the current study. First, the most urgent task is to increase recall for the invoice category of interest through resampling techniques, such as SMOTE, or by tweaking the class weight parameter for the Random Forest model. Anomaly detection techniques such as Isolation Forest or Autoencoders could be included to identify invoices which do not conform to the rules defined in this study. The change to online

machine learning will allow the model to learn new data as it comes in. The addition of a RESTful API layer will make it easier to integrate with enterprise applications such as SAP or Oracle Financials. Lastly, NLP can segment the feature space.

REFERENCES

1. Abdallah, A., Maarof, M. A., & Zainal, A. Fraud detection system: A survey. Journal of Network and Computer Applications, 68, 90-113.
2. Carbonneau, R., Laframboise, K., & Vahidov, R. Application of machine learning techniques for supply chain demand forecasting. European journal of operational research, 184(3), 1140-1154.
3. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.
4. Géron, A. . Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. " O'Reilly Media, Inc.".
5. Mishra, A. (n.d.). Vendor invoice intelligence system [Video tutorial series]. YouTube. Available online: https://www.youtube.com/@AyushiMishra
6. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. Scikit-learn: Machine learning in Python. The Journal of machine Learning research, 12, 2825-2830.
7. Raschka, S., & Mirjalili, V. Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2. Packt publishing ltd. Packt publishing ltd.
8. Zheng, L., Liu, G., Yan, C., & Jiang, C. Transaction fraud detection based on total order relation and behavior diversity. IEEE Transactions on Computational Social Systems, 5(3), 796-806.
9. Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.
10. Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).
11. Han, J., Kamber, M., & Mining, D. (2006). *Concepts and techniques*. Morgan Kaufmann, *340*(1), 94104-103205.
12. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning with Applications in R* (2nd ed.), 978-1-0716-1418-1, Springer.
13. Kotu, V., & Deshpande, B. . Data science: concepts and practice. Morgan Kaufmann.
14. Ngai, E. W., Hu, Y., Wong, Y. H., Chen, Y., & Sun, X. . The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. Decision support systems, 50(3), 559-569.
15. Mishra, A. (n.d.). Vendor invoice intelligence system Dataset Kaggle Available online: https://www.kaggle.com/ayushimishra2809