ABSTRACT

An invoice management system is an application or platform that tracks the dues,
dues clearance, previous invoices, and fulfillment as well as enables the user to process the invoices in a logical order such that they are cleared timely.
In B2B real world, recognizing the due clearing dates is an important aspect of the smooth running of the company. This application can be used for the same purpose.

The AI-Enabled FinTech B2B Invoice Management Application has been built by using a machine learning model to predict the status of the invoice and when it will be cleared. This system helps businesses to process their pending invoices in a logical order such that they are cleared on time. The application has React JS for frontend and JAVA for the backend whereas the middleware works are done using JAVA Servlets. The model is deployed into production using Flask.

Keywords: Machine learning, Web Application, B2B, Invoice prediction, Regression, Artificial intelligence, invoice management, ReactJS, Java Backend

Contents

List of Figures

Chapter 1

Introduction

An Invoice Management System is a computer web application program that allows businesses to manage their invoices. Invoice management systems help businesses to track their pending payments and dues through the invoices. It helps the client to identify potential late payments with the help of machine learning algorithms.

An invoice management system can also automate the account receivable, beginning with the payment due date, invoice date, baseline date, invoice amount, etc. The B2B business network works on credit systems. When a buyer business orders goods or services, the seller business issues an invoice against it which is generally cleared later. This is where the invoice management system jumps in.

Invoice management software is also shareable, from the customer service team to the accounting team, the warehouse staff, and you, the business owner. The best invoice management systems also have a built-in prediction system to give you insights into the clearing of the dues and enable smooth clearing and tracking of the pending and upcoming invoices. The invoice management system is an integral part of any B2B business as it helps in smooth workflow and clear insights on the dues and invoices which improves the working efficiency of the company or business.

Effective invoice management improves the business workflow and increases the likelihood of timely clearing dues. It also is an effective tool to track who to work with for a smooth workflow of the company.

Chapter 2

Basic Concepts

Most invoice management software programs follow a 6-step system, relying on automation to help businesses accurately process and view customer invoices. When used properly, invoice management software produces a seamless flow from entering invoices through after-sales follow-up. The smoother the software runs, the faster one can process, view, and add invoices, and the less likely to call a person that would not clear the invoice at that time or give more products to the customer with excess pending dues and uncleared invoices.

2.1 Entry of invoice details

The customer places an order through your third-party sales site, your own website, or over the phone with a live representative. Online, your customers will enter their details on a standardized form, with an option to have a securely saved preferred payment method. To improve the data collection process, make all fields of your online form mandatory so that you have all the necessary contact information for the customer up-front. This creates an invoice record and allows your invoice management system to track their dues history, the volume of orders, and payment and delivery preferences. It also gives you their phone number and email address, should you need to follow up with due invoice recovery. The company representative can then easily track the companies to call for the recovery of the due invoices.

2.2 Dataset

For the smooth functioning of this project, the basic structure on which the application runs is data. It is a data-intensive project and to store such a huge amount of data we need to have a properly designed database with properly designated methods to handle the data. For this project, MySQL database and SQL WorkBench have been used to ensure the smooth handling of data to and from the database.

2.3 Data Preparation

There are various stages of processing that the data has to go through before it can be used for prediction and model training processes. These processes are known as data preprocessing and data cleaning. For the same, python has been used as it eases the task with so many built-in libraries and additional package support for data cleaning as well as machine learning algorithms. At last, after data cleaning, we get a uniform collection of data which can now be easily used for model training and also for producing analytical insights from them. The analytical insights are made using Matplotlib and Seaborn libraries.

2.4 Data Prediction

The machine learning part has been implemented using various python libraries and machine learning algorithms. In statistics, linear regression is a linear approach to modeling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables). The case of one explanatory variable is called simple linear regression; for more than one, the process is called multiple linear regression. This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable. I have used different models to find the best-suited one on the basis of accuracy and generalization.

2.5 Data Display

A robust frontend UI has been created using the ReactJS framework with basic grid layouts having various functionalities to enhance the look of UI/UX. All the data is manipulated with the help of Java from and to the database. Java Servlets handle all the functionality of data modification or handling using all the API formats and coding standards.

Chapter 3

Problem Statement / Requirement Specifications

The proposed application is a full-stack application that is intended to store and display various information of different invoices present to help people working in accounts receivable departments in their day-to-day activities. It also has the prediction functionality that allows the user to track the delay in the clearance of invoices or the expected time in which the invoices would be cleared. The second main part of the project is the backend implementation with the frontend application. The problem statements for both sections are identified and written below.

1) Problem statement for Machine Learning model:

   - View the invoice data from various buyers.
   - See fields of invoices from a particular buyer/business.
   - Perform data preprocessing on the invoice date.
   - Get account-level analytics to easily visualize and interpret the data - EDA and Feature Engineering.
   - Get a prediction on when the invoice is going to get cleared.

2) Problem statement for Web Application development:

   - To build a full-stack invoice management application using JDBC, ReactJS, Java, and Servlets.
   - Build a responsive Receivables dashboard.
   - Visualize data in the form of grids.
   - Visualize data in the form of graphs.
   - Perform searching operations on the invoices.
   - Add and edit data in the editable fields of the grid.
   - Delete data of selected rows in predefined templates.

3.1 Project Planning

The various aspects of the projects are to be planned rigorously beforehand so that all features of the project are completed and implemented successfully.

The machine learning model needs to be implemented with the help of the given data set that will allow the users to predict the delay and expected time of the pending due invoice clearance.In statistical modeling, regression analysis is a set of statistical processes for estimating the relationships between a dependent variable and one or more independent variables. The most common form of regression analysis is linear regression, in which one

finds the line that most closely fits the data according to a specific mathematical criterion. For example, the method of ordinary least squares computes the unique line that minimizes the sum of squared differences between the true data and that line. Less common forms of regression use slightly different procedures to estimate alternative location parameters or estimate the conditional expectation across a broader collection of non-linear models. Regression analysis is primarily used for two conceptually distinct purposes. First, regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Second, in some situations, regression analysis can be used to infer causal relationships between the independent and dependent variables. Importantly, regressions by themselves only reveal relationships between a dependent variable and a collection of independent variables in a fixed dataset. To use regressions for prediction or to infer causal relationships, respectively, a researcher must carefully justify why existing relationships have predictive power for a new context or why a relationship between two variables has a causal interpretation. The latter is especially important when researchers hope to estimate causal relationships using observational data.

We need to see all results by fitting the data in different linear and nonlinear models like linear regression, K-means regression, KNN, XGBoost regressor, etc. to get the best accuracy and a generalized model. A generalized model is one that not only performs well on train data but also gives good accuracy on test data or unseen data.

For Java backend, Servlets would need to be made. Each function from fetching data from data to editing data and deleting them should be implemented using different servlets. The servlets should be fully functional and will enable smooth exchange and updating of data in the database and frontend.

3.2 Project Analysis

The project needs to be carried out in a certain manner so that there is no ambiguity or to ensure a smooth flow of work throughout the project. First, we need to make the machine learning model so that we can test it on the real-time data we have. After that, we would be making servlets and APIs.
In statistics, linear regression is a linear approach to modeling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables). The case of one explanatory variable is called simple linear regression; for more than one, the process is called multiple linear regression. In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, the conditional mean of the response given the values of the explanatory variables (or predictors) is assumed to be an affine function of those values; less commonly, the conditional median or some other quantile is used. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the response given the values of the predictors, rather than on the joint probability distribution of all of these variables, which is the domain of multivariate analysis. Linear regression was the first type of regression analysis to be studied rigorously and to be used extensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

3.3 System Design

3.3.1 Design Constraints

The Web Application runs on ReactJS with JAVA as the Backend. Java 1.8 is the standard runtime environment to process the backend information. The UI is rendered on ReactJS installed with NodeJS or yarn. The date prediction feature is implemented on Python for which version 3.6 or later is recommended. Python Flask has been used to deploy the date prediction model for which the recommended version is 1.1.2. The python libraries which are required to run the prediction model are NumPy, pandas, scikit-learn, seaborn, matplotlib, and xgboost. The latest version of Pandas is preferred to avoid errors during the execution.
The Analytics view functionality in the frontend requires React chart-js-2 library to work.
To debug or perform further development in the product, the following IDEs are preferred for each part-

- Backend - Eclipse IDE
- Python - Jupyter Notebook (with Anaconda) or Google Colab
- ReactJs - Visual Studio Code

3.3.2 System Architecture **OR** Block Diagram

A robust system architecture is defined to ensure smooth workflow of the application. The whole workflow can be visualized in the below flowchart:

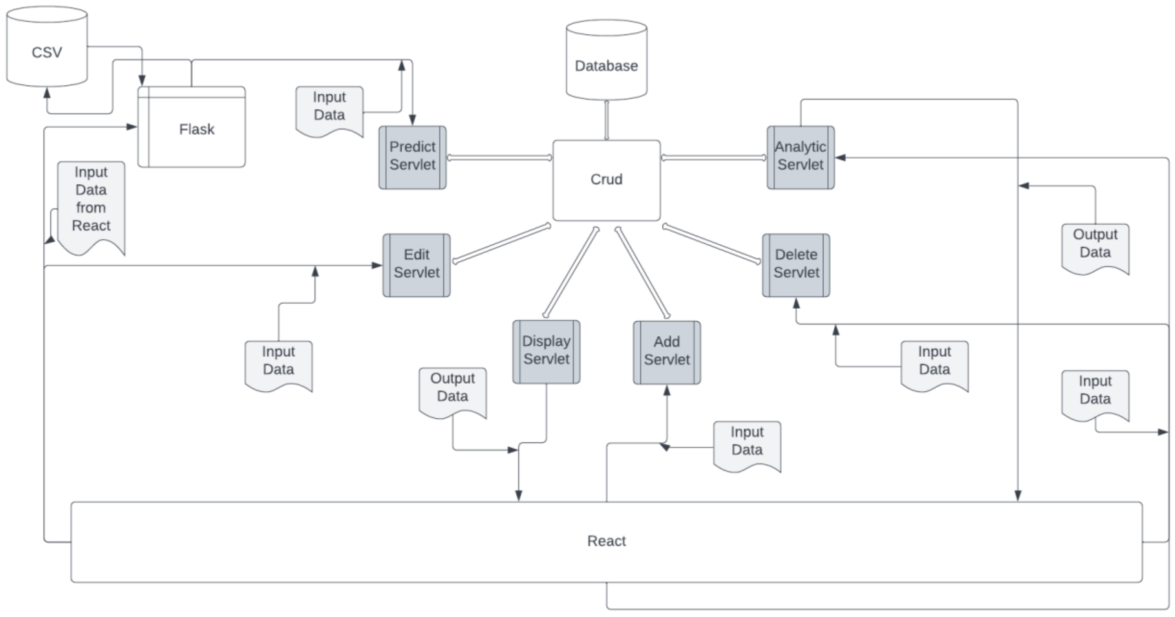


Figure 1. Application workflow architecture

Chapter 4

Implementation

4.1 Methodology

4.1.1. Prediction of Aging Buckets using Machine Learning Techniques

The machine learning part has been implemented using various python libraries and machine learning algorithms. First, we imported the data and stored in a pandas data frame object to perform further operations. The dataset used consisted of 50000 rows with 18 features providing certain information about the invoices. The dataset was split into two sets(in 80:20 ratio)-

1. Train set which consisted of all previous invoice transactions (already completed/paid)
2. Test set which contains information about current invoices (still due in payment)

This is done so that the model can be trained on the train set and then the model is implemented on the test set. The train set extracted was now split into 3 sets (having a ratio of 70:15:15). The new sets are named train set, validation set1, and validation set2 respectively. The train set is used to train the ML models, the validation set1 is used to cross-validate the model and tune hyperparameters to react to global maximum accuracy and the validation set2 is used to test the model performance on unseen data. Several preprocessing techniques and feature extraction techniques were used to extract the most important features from the dataset. It is very important to know the relation or dependency of features with each other. For this purpose, the Pearson Correlation matrix is plotted which tells us how strong a feature is dependent on another. This step helps in feature selection to further process the data. Afterward, the normalization of numerical data, numerical encoding of non-numeric data, converting amount value to a uniform currency, and removing outliers are done to make the dataset feasible for the model to train.
The main objective of this part of the system is to predict the aging bucket of the payment for which the invoice clear date has to be predicted. Hence for training the data a new feature, “delay” is introduced which is predicted by the model. Afterward, delay days can be grouped into different aging buckets. The following Machine Learning models have been tested for the delay prediction:

- Linear Regression
- Support Vector Machines (Default Kernel)
- Decision Trees
- Random Forest
- XgBoost

The full workflow of the prediction process can be understood with the help of the below flow diagram:

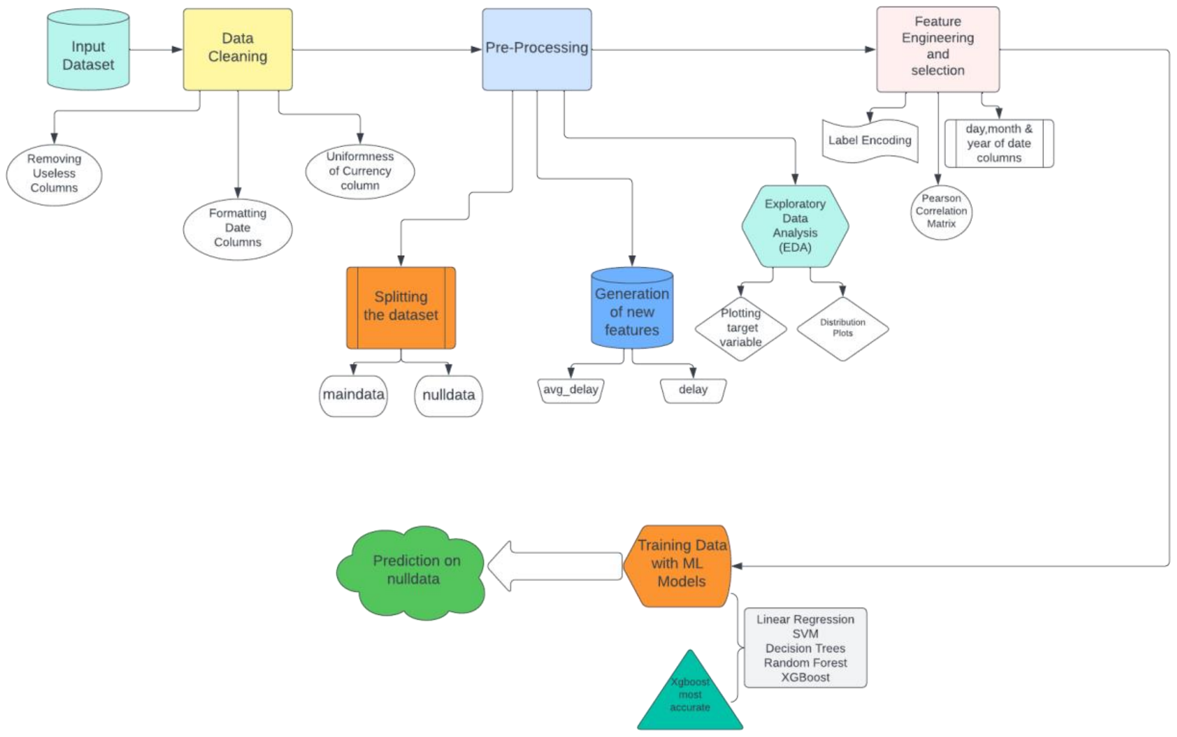


Figure 2. Machine learning workflow

4.1.2 Frontend

The frontend is created using HTML, CSS, Javascript, and ReactJs. Datagrid was used to create the main table and then implemented an infinite scroll in it. For the buttons, button functions have been used and connected to them using servlets to perform CRUD operations like adding, deleting, and updating invoice records. The add button adds new data to the table and the delete button deletes selected rows. The edit button displayed a form to update the details of a selected row. We also added a predict button that connects the UI to the ML model. By clicking on it we get the predicted payment date. Various pop-up dialog boxes with proper messages have been created for the add, delete and update to make the UI more user-friendly. The description of all the React Components is given below:
The overall workflow of the react components can be understood by the below flowchart:

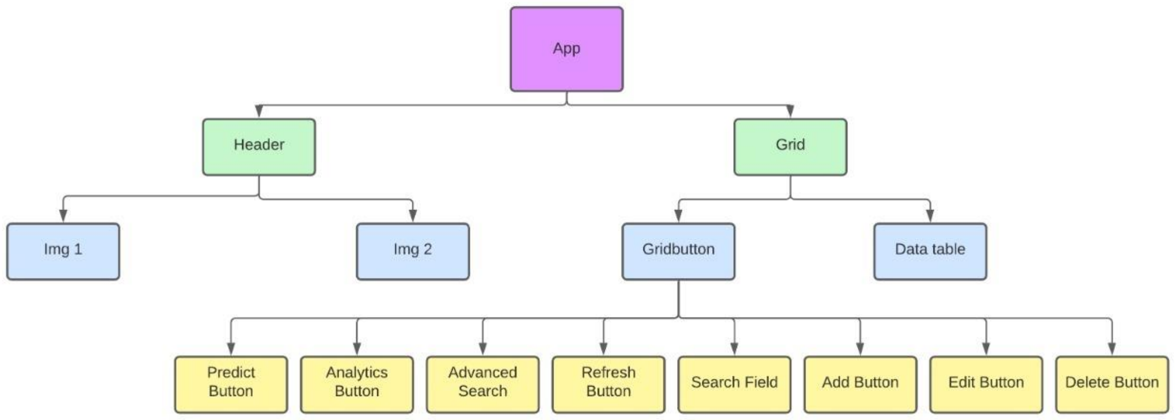


Figure 3. ReactJS component tree structure

Add Button:

The Add button will remain in the enabled state when no rows are selected.
Whenever one or more rows are selected, the Add button will not remain activated.
After clicking on the Add button, a dialog box will appear with all the fields for which values need to be added along with a Cancel and an Add button. The user can fill all the required fields before adding. If the user tries to click on add before all mandatory fields are filled, an error message will be displayed. Once the user clicks on the add button, after all, mandatory fields are filled, the new values will be displayed in the UI and it will remain persistent.

Edit Button:

Clicking on the edit button will allow the user to modify the editable fields in the data. The editable columns are Invoice Currency and Customer Payment Term. The edit button will remain in a disabled state by default. When the user selects one row, the Edit button gets enabled. If a user selects multiple rows, the edit button will remain disabled. Clicking on the Edit button displays a dialog box on the screen. The window contains the Invoice Currency and Customer Payment Term headers along with the existing data values for both, a Cancel and an Edit button. The user is able to edit the values. Once the user clicks on the Edit button, the new values will be displayed in the UI and will remain persistent.

Delete Button:

Clicking on the delete button will allow the user to delete records from the grid. The delete button will remain in a disabled state by default. When the user selects one or more rows, the delete button gets enabled. A pop-up will be displayed on clicking delete to confirm that the user wants to delete the selected records permanently. Once the user clicks on the delete button, the row(s) will be removed from the UI and it will remain persistent.

Search:

Users will be able to search for a customer by typing text in the Customer id integer field. Search is not case-sensitive. The UI suggests all the rows related to the customer number that the user will be typing.

Analytic View:

To get insights from the existing data based on user inputs. The existing parameters would act as key points or outliers for the synthesis of data. So the analytics view will be a button in UI that responds to a new window on click event. The new window contains of parameters:

- Currency
- Due Date
- Baseline Create Date
- Clear Date

The user will have the privilege to go for a single parameter or multi-parameter based on their choices and preferences. On submitting the parameters the web application will open the dialog window which will provide the user with an illustration of a bar graph and pie chart which will be formed based on the parameterized data that the user had selected. The bar graph will be showing data for the total open amount and number of customers for all businesses.

Advanced Search:

The UI consists of the Advanced Search button. Clicking on this button will help the user to perform an advanced search on the data based on the following four fields: doc_id, cust_number, invoice_id, buisness_year

Refresh:

Clicking on the Refresh button will soft refresh the grid and display the grid with latest data.

Predict:

Clicking on this button will populate the Predicted Payment Date column on the UI with the predicted dates. When the user selects one or more Invoices and clicks on the Predict button, the Predicted Payment Date column should get populated only for those invoices. The button will get activated only upon selecting any of the Invoice(s). If no Invoice is selected, the button will be in a disabled state.

4.1.3 Backend

The API calls generated during the frontend were handled by Java Servlets running on Apache Tomcat Server. One servlet for each of add, edit, delete, and search functionality was created. These servlets use JDBC or Java Database Connectivity to get or change data present in the MySQL database. The basic structure of the Database is represented inside the Pojo Class which is the model or entity of the database. It contains all attributes(columns) of the table as class member variables which are fetched or assigned by using getters and setters methods.

The CRUD functionalities along with the Prediction and Analytical view have been defined under the Crud class. It contains all the necessary functions to facilitate flow data from the database to react through servlet using JDBC. The following functionalities are defined below:- *createConnection()*:-It is a static function that establishes a connection between the database and JDBC. It can be directly invoked without creating Crud object to avoid overload on the server.

DisplayData():-This function returns an ArrayList of object type Pojo which contains all the rows of the table.

addData():-It takes an object of type Pojo which contains all the attributes of the row which has to be inserted in the table. It returns 0 on failed processes and 1 on successful entry.

editData():-It takes an object of Pojo and extracts the required fields and updates the table. It returns 0 on failed processes and 1 on successful entry. Another utility function *fetchDetailsbySl()* fetches the data for the selected row from the grid through its serial number.

delete():-It accepts an in an array of sl_no of rows and deletes the respective entries. It returns an array of 0 on the failed process and an array of 1 on successful entry for each and every respective sl_no.

getAnalyticData():-It accepts ArrayList of String type and which contains required fields on the basis of which it sends ArrayList of Object type AnalyticPojo.

predictToDatabase():-It receives String of docID, agingBucket, and clear_date and sets according to docID.

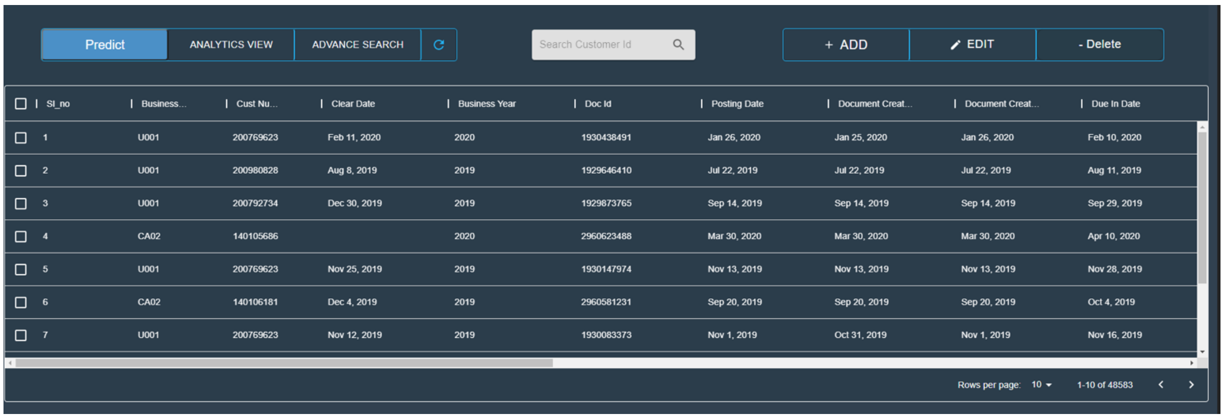
And at the last Servlets facilitated the data flow from ReactJS to JDBC and from JDBC to react through Servlet API in JAVA.

4.1.4 Flask Integration

The Flask integration involved integrating the ML model to provide the predictions of the invoice's future payment dates in the UI itself. This is done using Python and the Linear Regression ML model which was trained using the past dataset. Flask creates an application on the default port 5000 where it routes the incoming prediction requests to /get_prediction and performs the prediction from the custom New_Bucket module. The response is then posted back to the root route where it is fetched by the Servlet and the aging bucket is updated to the database.

4.2 Testing

There are several functionalities so as to ensure that all of them are working fine, each of them was tested individually. The Datatable component of our react application successfully renders the grid in which records are shown from the database. The same can be visualized in the below image:



Sl_no	Business...	Cust Nu...	Clear Date	Business Year	Doc Id	Posting Date	Document Creat...	Document Creat...	Due In Date
1	U001	200769623	Feb 11, 2020	2020	1930438491	Jan 26, 2020	Jan 25, 2020	Jan 26, 2020	Feb 10, 2020
2	U001	200980828	Aug 8, 2019	2019	1929646410	Jul 22, 2019	Jul 22, 2019	Jul 22, 2019	Aug 11, 2019
3	U001	200792734	Dec 30, 2019	2019	1929873765	Sep 14, 2019	Sep 14, 2019	Sep 14, 2019	Sep 29, 2019
4	CA02	140105686		2020	2960623488	Mar 30, 2020	Mar 30, 2020	Mar 30, 2020	Apr 10, 2020
5	U001	200769623	Nov 25, 2019	2019	1930147974	Nov 13, 2019	Nov 13, 2019	Nov 13, 2019	Nov 28, 2019
6	CA02	140106181	Dec 4, 2019	2019	2960581231	Sep 20, 2019	Sep 20, 2019	Sep 20, 2019	Oct 4, 2019
7	U001	200769623	Nov 12, 2019	2019	1930083373	Nov 1, 2019	Oct 31, 2019	Nov 1, 2019	Nov 16, 2019

Rows per page: 10 1-10 of 48583

Figure 4. Grid UI demonstration

4.3 Result Analysis OR Screenshots

The project successfully performs well on testing standards. The main objective of the Machine Learning model is to predict aging buckets which are completed with good accuracy. A total of five models were trained and tested for the delay prediction. The results of which can be summarized in the below table:

Model	**R2-Score**
Linear Regression	0.315850
Support Vector Machines	-0.007416
Decision Trees	0.620362
Random Forest	0.726154
XGBoost	0.738961

XGBoost performed the best in the delay prediction of invoices hence it had been used to predict all the clear_date rows which are missing.
A detailed analysis of the invoices can be found in the Analytics view section of the web application. A sample of the same can be visualized below:

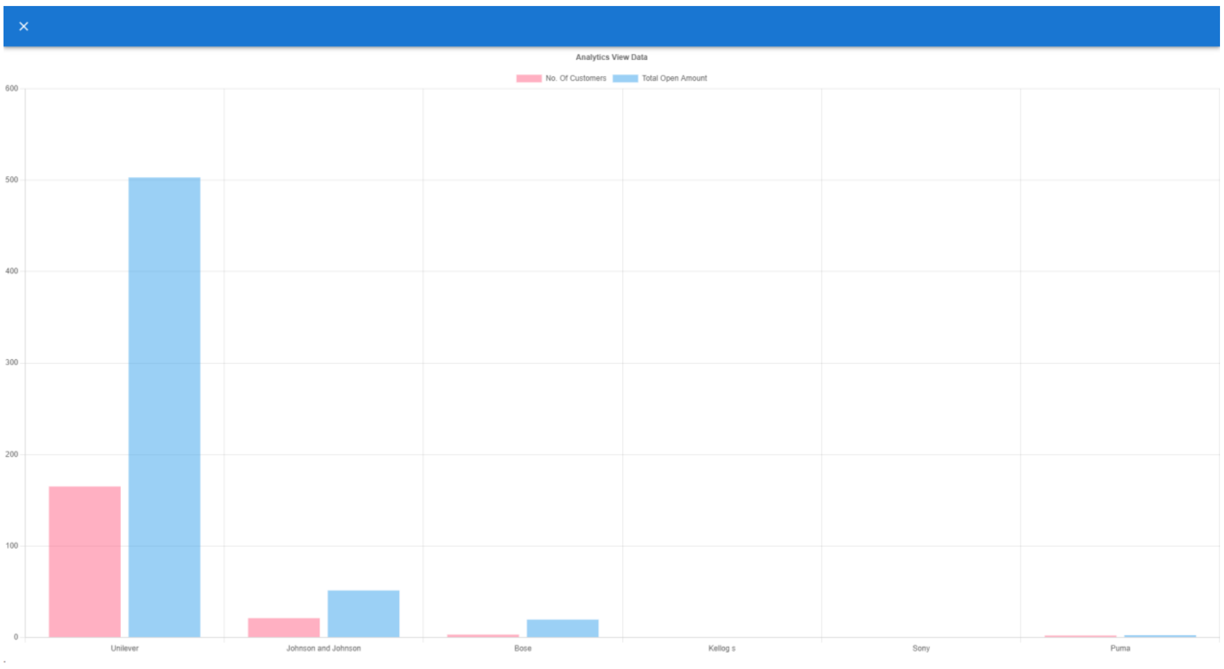


Figure 5. Analytics of the invoices in ChartJs

Chapter 5

Standards Adopted

5.1 Design Standards

The application is forged with a robust design that is user-friendly and with minimal bugs in UI and Backend. The system design follows a fully-functional Object-oriented approach in the backend with the help of Java classes and objects. The planning and design of the application are implemented with the SDLC lifecycle.

5.2 Coding Standards

The application source code is a well-developed and clean code that follows proper naming conventions for each entity used in the application.
The functions have been properly defined so as to give a meaningful name according to their functionality.

5.3 Testing Standards

The application has been tested in the localhost server environment. All the three servers which are used in the application have been tested under individual medium-latency computational architecture with a very bright scope of mass scaling.

Chapter 6

Conclusion & Future Scope

6.1 Conclusion

An invoice management system is any tool or platform that tracks invoices, dues and clear dates as well as enables the people, processes, and partnerships necessary for products to find their way to the customers who bought them. In this project, I have built an AI-Enabled FinTech B2B Invoice Management Application using a regression model to predict whether payment of an order will be delayed or not. The accuracy of the model turned out to be 78%. The same has been deployed using Flask Framework and React JS for the frontend. The whole full stack web app will enable businesses to easily track invoices and grow their firms with healthy partnerships. The whole system will make the workflow seamless and efficient.

6.2 Future Scope

The project can be deployed on a cloud platform for a larger approach to audiences. The model of the prediction system can also be more fine-tuned for greater accuracy. The database design can be more systemized so that the data in the database is uniform and can enable good data analysis and insights. UI/UX of the application can be improved for appealing to the users.

References

[1] Chen T, He T, Benesty M, Khoti Lovich V, Tang Y, Cho H, Chen K. Xgboost: extreme gradient boosting. R package version 0.4-2.,2015 .

[2] Mitchell R, Frank E. Accelerating the XGBoost algorithm using GPU computing. PeerJ Computer Science. 2017 Jul 24;3:e127.

[3] Javeed A. Performance optimization techniques for ReactJS. In2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), 2019, IEEE.

[4] Aggarwal S. Modern web-development using reactjs. International Journal of Recent Research Aspects, 2018.

[5] Dietrich SW, Urban SD, Kyriakides I. JDBC demonstration courseware using Servlets and Java Server Pages. ACM SIGCSE Bulletin, 2002.

[6] Cecchet E, Chanda A, Elnikety S, Marguerite J, Zwaenepoel W. Performance comparison of middleware architectures for generating dynamic web content. InACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing 2003 Jun 16 (pp. 242-261). Springer, Berlin, Heidelberg.