

Machine Learning Model for Annual Recurring Revenue Forecasting

Juan David Arroyave Ramirez

March 2025

1 Abstract

This abstract details the development of a Machine Learning model designed to forecast the Annual Recurring Revenue (ARR) for a company in the financial sector. The dataset used for this process comprised 171,721 rows and 17 columns, each containing financial information about current clients, including gross wages and other fees contributing to revenue calculations.

To develop the training model, supervised learning techniques were employed to predict outcomes in tabular data by understanding the relationships between dependent and independent variables. Python was used as the programming language within Visual Studio Code. After testing various regression algorithms, the Light Gradient Boosting Machine (LGBM) was selected for its superior accuracy, aligning predictions more closely with the dataset.

Given the model's complexity and its purpose for extrapolation, there remains room for improvement through hyperparameter tuning using grid search and pipelines. The primary goal of this model is to accurately predict the revenue of prospects for future months. Currently, the model achieves a training accuracy of 71% and a testing accuracy of 62%.

The model was deployed following an Exploratory Data Analysis (EDA) phase, utilizing the PyCaret library to identify suitable models. An application was created using Tkinter and PyInstaller, allowing users without programming skills to input relevant data, access charts, and execute the model.

2 Introduction

One of the biggest concerns in science and technology is the practical application of developed tools. Instead of complicating tasks, the goal is to improve quality of life by making things faster, easier, and more efficient. Technology represents evolution and innovation, as past tools are enhanced or replaced with new inventions and lab developments that solve daily life problems on both large and small scales. Technology continues to advance rapidly, and the use of computers and machines significantly impacts all sectors, driving digital transformation and converting old processes into more efficient ones.

Although companies are distinguished by their distinct products or services, they continuously strive to enhance their internal processes. In the face of intense competition, businesses focus on expanding their operations and delivering the highest quality to consumers, employees, and partners in general. Therefore, this digital transformation has integrated Artificial Intelligence (AI) into many processes. Although not all industries have adopted it yet, AI is poised to become a benchmark for all sectors due to its wide-ranging applicability, nowadays, people can get free access to Generative Pre-trained Transformer (GPT) models. Those can perform complex calculations based on user requests.

Artificial Intelligence (AI) is globally recognized for its use of GPT models in chatbots and virtual assistants. Additionally, AI encompasses robots that can simulate human behavior, tools for image and content creation, complex data analysis, and numerous other applications across various fields. To function accurately, AI relies on training models operating in the background, driving their mechanisms efficiently. This is where Machine Learning plays a crucial role as the powerful engine that brings AI to life.

Does AI need to learn something to achieve intelligence? The answer is yes, and here is where Machine Learning (ML) takes place. ML is the foremost field that empowers systems to autonomously learn and adapt based on the data they receive. It concentrates on developing training models equipped with algorithms capable of making highly accurate predictions or decisions, closely resembling human-like calculations. Machine Learning can be trained to solve problems of Regression, Classification, Dimensionality Reduction, Clustering, and Reinforcement. When a model is complete, it gains the capability to comprehend complex data, being reliable for new outputs as needed.

In this scenario, a regression problem arises where a company needs to predict the annual recurring revenue for new clients who will be joining progressively over the coming months. Therefore, the Machine Learning model needs to analyze tabular data to identify the significant key features to make accurate forecasts. This scientific paper will detail the development of the training model and the results.

3 Literature Review

To start the literature review, it is essential to establish a solid foundation in the field of Machine Learning and its various branches. Additionally, it is crucial to have a clear understanding of the problem that needs to be solved.

Machine Learning Overview Jean Philippe Cedric N'DRI Quantitative Researcher - Data Scientist 20th January 2025.

This reference provides an overview of Machine Learning training models, including supervised, unsupervised, and reinforcement learning. It covers key models such as Regression, classification, Clustering, and Dimensionality Reduction. Additionally, this reference explains how to identify when a model is overfitting or underfitting.

Revenue Forecasting Example (Using Linear and Seasonal Regression)

This reference provides an example of using Linear Regression to predict and set revenue goals based on existing trends. Which is useful to have an idea of how to face the current problem of predicting the ARR.

Simple Linear Regression model to predict the Salary based on Years of experience (Yash Kulkarni)

This reference gives an example of how to calculate with Linear Regression the Salary based on years of experience,

Walmart Weekly Revenue Prediction This reference provides a weekly revenue prediction from the Walmart Sales Dataset applying linear regression with the Random Forest Regressor.

Using Linear Regression in The Analysis of Financial-Economic Performances

This reference executes an analysis of economic performance by applying a multiple linear regression model.

Financial stock market forecast using evaluated linear regression-based machine learning technique

This reference applies linear regression to predict the future value of a company's final stocks, analyzing prices, and the stock market in general to train the model.

4 Problem Description

In the first instance, the phenomenon that needs to be solved is predicting the annual recurring revenue (ARR) for clients and prospects joining the company. This financial calculation determines the potential impact of these clients on the company's revenue. Staying ahead is a priority to remain competitive in the current market, therefore, implementing these technologies supports the company's goals of expansion and growth.

To predict the ARR, the company provided a .CSV dataset with a shape of (171721, 17). The features include Holding-ID, Client-ID, Client-Other-ID, Client-Name, Adj Pay Period, Month, Year, Min Month Invoice, Max Month Invoice, Gross Wages, WC, CycleData.Days Per Cycle, WSE Count Billed, DivisionDataWSE.WSE Total, Periods, Adjusted Monthly WSEs, and ARR as the target. These features contain information about clients and payments while preserving confidentiality. The dataset includes data from all months of 2022 to 2024, as well as the first months of 2025. Although the objective is to predict the target for future months, the model will perform extrapolation since the predictive analysis will extend beyond the range of the training data.

To develop the machine learning model, Visual Studio Code was utilized as the primary tool for executing Python scripts. Given the nature of the dataset and the target outcomes, the problem was identified as a regression task. The objective is to discern patterns, associations, and historical data to establish relationships between dependent and independent variables for target prediction, so regression is needed in these cases. Consequently, this is a supervised learning task involving continuous output.

Once the problem was identified, and the company's expectations with the model, an Exploratory Data Analysis (EDA) into Visual Studio Code was carried out according to the workflow: Load Raw Data, Descriptive Analysis, Adjustment of Variable Types, Detection and Treatment of Missing Data, Identification of Atypical Data, and Correlation of Variables. After this, the processes to develop the machine learning model were carried out according to the standard workflow: Data Collection, Data Preprocessing, Model Design, Model Training, Model Evaluation, Perform Inferences, and Conclusions.

The Python scripts for Exploratory Data Analysis and the Model Training were executed in Visual Studio Code using Jupyter Notebooks, with the Python kernel version 3.12.5. The following libraries were imported: pandas, numpy, seaborn, matplotlib, scipy, plotly, scikit-learn, and the regressor algorithm LightGBM. Following this evaluation, the PyCaret library was utilized to identify the most suitable algorithms for training the model. Orthogonal Matching Pursuit achieved the highest score according to PyCaret; however, it did not yield the desired R^2 score upon execution. Before using PyCaret, Linear Regression, Gradient Descent, and GBM Boosting algorithms were applied, but their training results were suboptimal. Ultimately, the LightGBM algorithm provided the best predictability with 90% of training and 10% of testing.

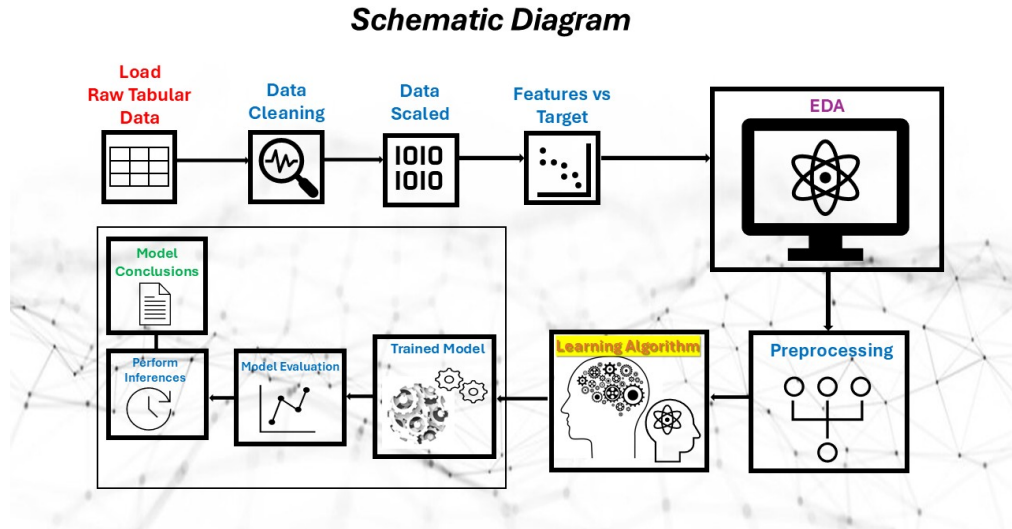


Figure 1: Schematic Diagram

5 Exploratory Data Analysis (EDA)

The Exploratory Data Analysis (EDA) was conducted as follows:

Importing necessary libraries:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
import plotly.express as px
import plotly.graph_objects as go
```

Loading Raw Data: The dataset used to predict the Annual Recurring Revenue (ARR) is named ARR-DATASET. It is provided in a comma-separated values (CSV) file, with the ARR labeled under the header “Target.” The dataset name will be identified as “df” within the code. “X” will be the feature columns for the training and “y” the Target.

Codes:

```

X = df.drop(columns=["Target"])
y = df["Target"]
df = pd.DataFrame(X)
df["Target"] = y df = pd.read_csv("ARR - DATASET.csv")
X = df.drop(columns=["Target"])
y = df["Target"]
df = pd.DataFrame(X)
df["Target"] = y

```

Descriptive Analysis: In this descriptive analysis, it is important to understand the dataset's shape, headers, and the information it contains, such as unique values, nulls, numerical data, and categorical data that impact the target.

Codes:

Dataset overview

```

df.info()
df.head(3)
df.tail(3)
df.describe().round(2)
print(df.isnull().sum()) print(df.shape)
df.info()
df.head(3)
df.tail(3)
df.describe().round(2)
print(df.isnull().sum())

```

Analyzing unique values per feature:

```

for feature in feature_names :
    unique_count = df[feature].nunique()
    print(": values".format(feature, unique_count))
    print(df.shape)
    print("The number of rows is: ", df.shape[0])
    print("The number of features is: ", df.shape[1]) feature_names = list(df.columns.values[:
-1])
for feature in feature_names :
    unique_count = df[feature].nunique()
    print(": values".format(feature, unique_count))
    print(df.shape)
    print("The number of rows is: ", df.shape[0])

```

```
print("The number of features is: ", df.shape[1])
float
```

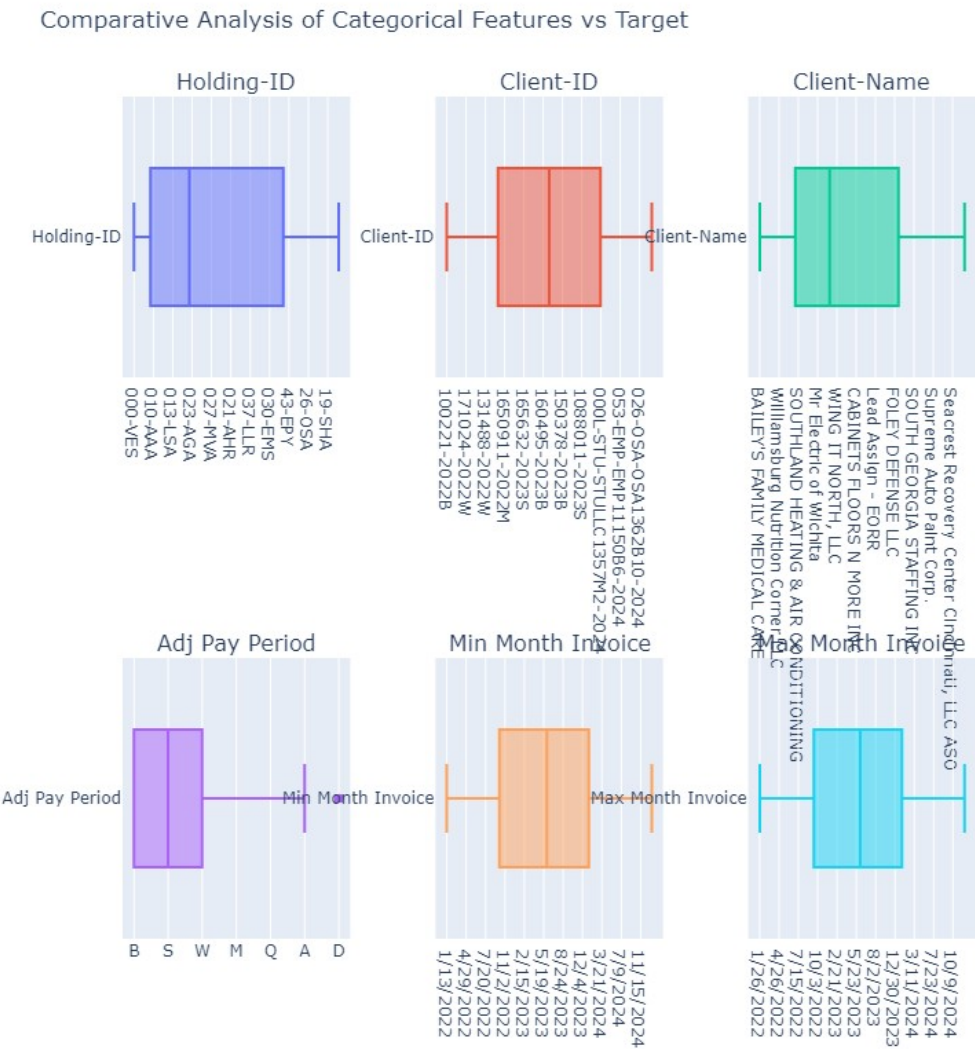


Figure 2: Categorical Data vs Target

Numerical Features Analysis

```
numerical_cols = df.select_dtypes(include=['number']).columns
print(numerical_cols)
df.hist(figsize=(10,10),bins=10,edgecolor='blue',color='silver')
float
```

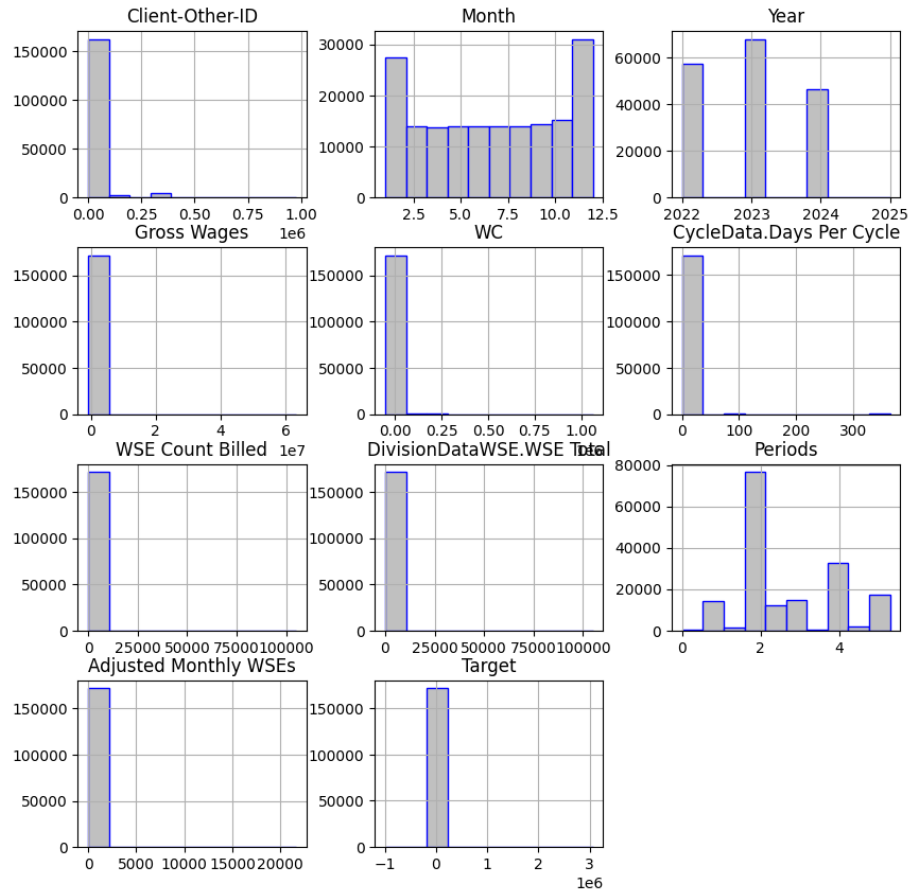


Figure 3: Categorical Data vs Target

Detection and Treatment of Missing Data:

Codes:

```
df = df.dropna()
```

Identification of Atypical Data (Outliers)

Codes:


```

numerical_ols = df.select_dtypes(include = ['float64','int64'])
Q1 = numerical_ols.quantile(0.25)
Q3 = numerical_ols.quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = (numerical_ols < lower_bound)|(numerical_ols > upper_bound)
outliers_df = numerical_ols[outliers.any(axis = 1)]
print(outliers_df)

```

Outliers Chart

```

plt.figure(figsize=(10, 6))
sns.scatterplot(data=outliers_df)
plt.title('Outliers detected in DataFrame')
plt.xlabel('Index')
plt.ylabel('Value')
plt.show()
float

```

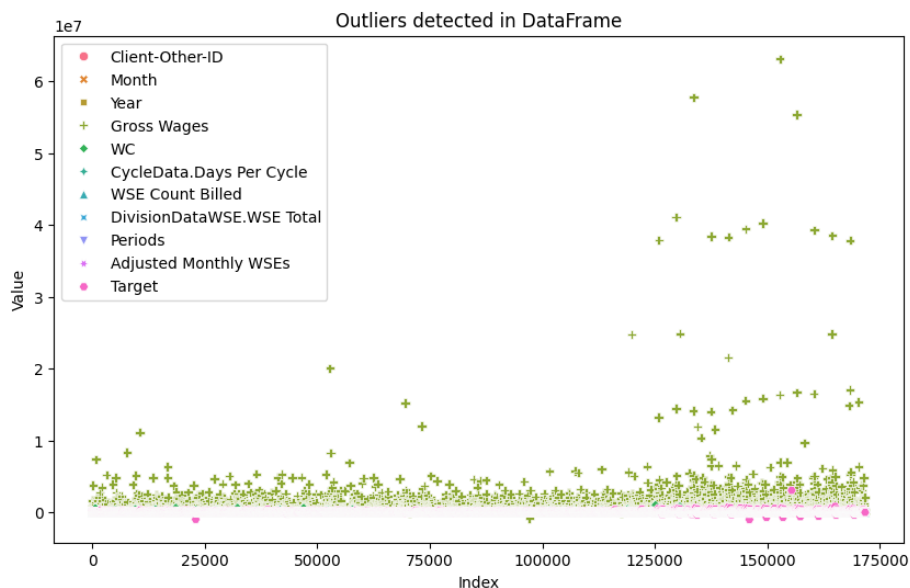


Figure 4: Outliers in Dataset

As represented most outliers correspond to the "Gross Wages" feature.

Since the company runs payroll for big or small business, the Gross Wages may vary. So, it is okay to have high pikes, and removing outliers is not desirable.

Adjustment of Variable Types

*In this scenario the procedure as follows is to normalize the data in "Gross Wages" configuring the negatives to the median and applying **FEATURE ENGINEERING** to do a **Log Transformation**.*

Detecting negatives in feature:

```
negative_values = df[df["GrossWages"] < 0].print(negative_values)
```

Configuring negatives detected to the median:

```
df["Gross Wages"] = df["Gross Wages"].apply(lambda x: x if x >= 0 else 0.01)
```

Log Transformation:

```
df["Gross Wages Log"] = np.log1p(df["Gross Wages"])
plt.figure(figsize=(12, 5))
sns.histplot(df["Gross Wages Log"], bins=50, kde=True)
plt.title("Histogram of Log-Transformed Gross Wages")
plt.show()
```

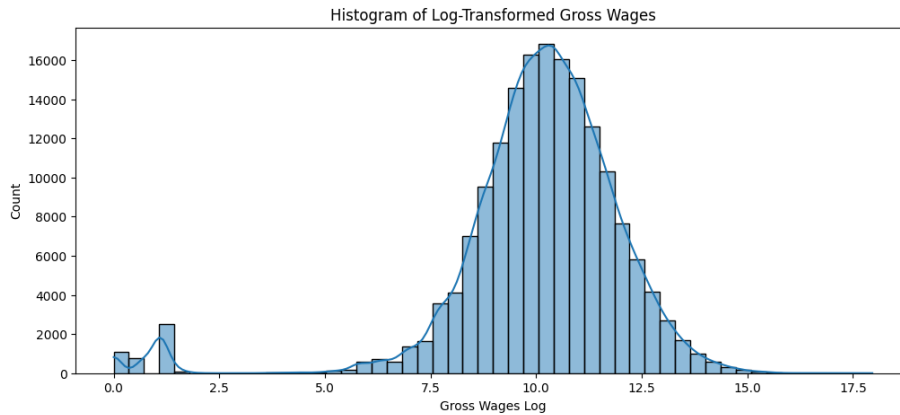


Figure 5: Log Transformation for "Gross Wages" feature

After the transformation, making sure the new feature is into float64 type for analysis.

```
df["Gross Wages Log"] = np.log1p(df["Gross Wages"]).astype('float64')
Replacing Null(NaN) with the median in the new feature.
median_value = df["GrossWagesLog"].median()
```

```
df["Gross Wages Log"] = df["Gross Wages Log"].fillna(median_value)
```

Application of the correlation matrix with numerical values.

Code:

```
numerical_ols = df.select_dtypes(include = ['float64', 'int64'])
corr_matrix = numerical_ols.corr()
plt.figure(figsize=(12, 6))
sns.heatmap(corr_matrix, annot = True, cmap = "Blues", fmt = ".2f", linewidths =
0.5)
plt.title("Correlation Matrix")
plt.show()
target_corr = corr_matrix["Target"].sort_values(ascending = False)
print(target_corr)
```

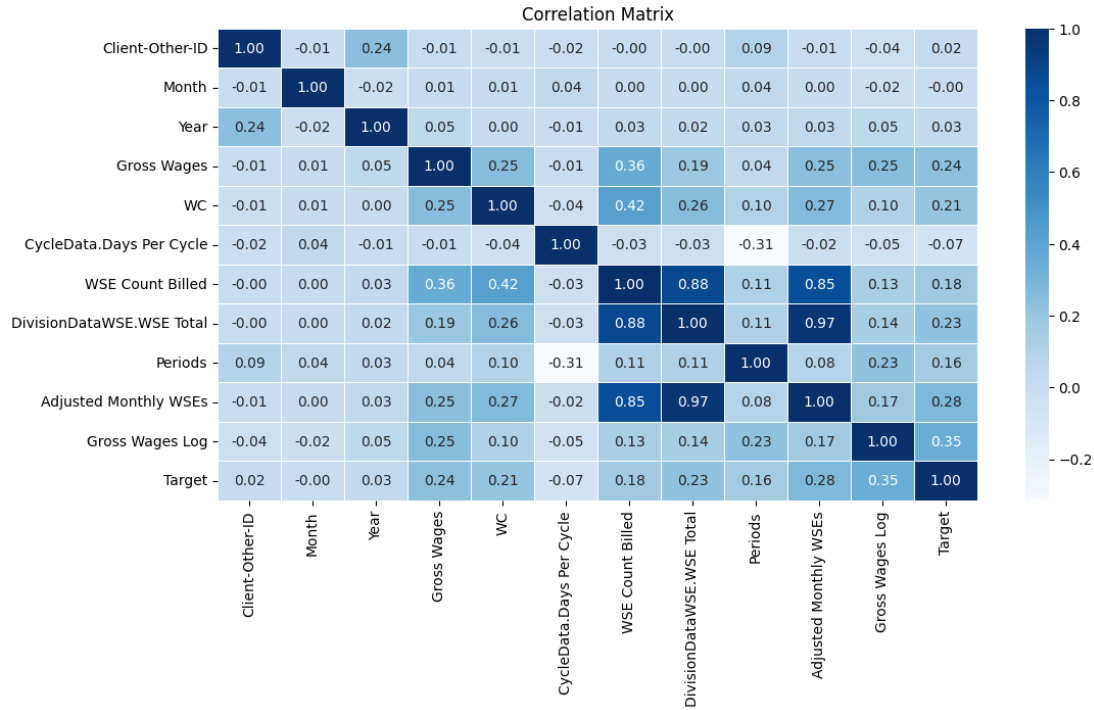


Figure 6: Correlation Matrix with numerical values

Result

Feature	Value
Target	1.000000
Gross Wages Log	0.349414
Adjusted Monthly WSEs	0.276971
Gross Wages	0.236668
DivisionDataWSE.WSE Total	0.227998
WC	0.205147
WSE Count Billed	0.175841
Periods	0.162032
Year	0.030959
Client-Other-ID	0.017576
Month	-0.001595
CycleData.Days Per Cycle	-0.070610

Table 1: Correlation Values for Features

6 Solution Implementation

Dropping unnecessary features according to EDA results.

```
df = df.drop(columns=["Holding-ID", "Client-ID", "Client-Other-ID", "Client-Name", "Min Month Invoice", "Max Month Invoice", "Gross Wages"])
```

Encoding Categorical Variables since high relation with Target applying OneHotEncoder

```
Code: from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(sparse_output = False, drop = "first")
encoded_payperiod = encoder.fit_transform(df[["Adj Pay Period"]])
encoded_payperiod_df = pd.DataFrame(encoded_payperiod,
columns=encoder.get_feature_names_out(["Adj Pay Period"]))
df = pd.concat([df.drop(columns=["Adj Pay Period"]), encoded_payperiod_df], axis =
1)
```

Normalizing and Scaling the data

Note: Since the model is extrapolating, we need to keep Month and Year as int64 for better forecast, also the One-Hot-Encoder technique already converted the 'Adj Pay Period' feature to a scaled level, so no need to scale it again.

```
scaler = StandardScaler()
scaled_features = ["GrossWagesLog", "WC",
                  "DivisionDataWSE.WSE Total", "Adjusted Monthly WSEs",
                  "CycleData.Days Per Cycle", "WSE Count Billed", "Periods"]
df[scaled_features] = scaler.fit_transform(df[scaled_features])
X = df[scaled_features + ["Month", "Year", "Adj Pay Period_B",
                          "Adj Pay Period_D", "Adj Pay Period_M", "Adj Pay Period_Q",
                          "Adj Pay Period_S", "Adj Pay Period_W"]]
y = df["Target"]
X = pd.DataFrame(X, columns=X.columns)
```

Correlation Matrix (Dataset cleaned)

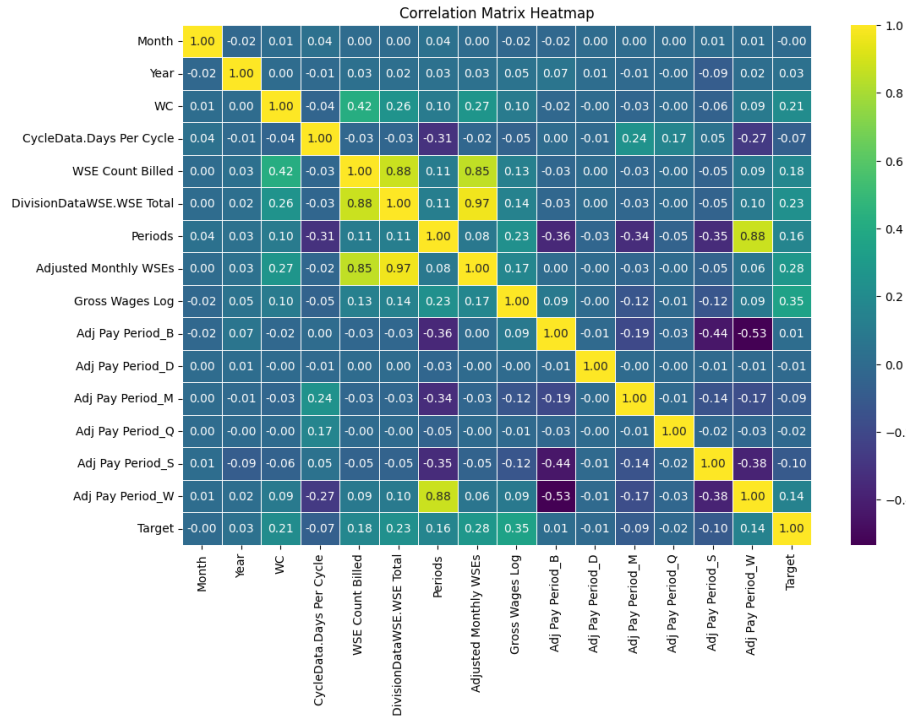


Figure 7: Correlation Matrix after data cleaned

6.1 Machine Learning Model:

The model was trained in first instance with three different algorithms:

- Linear Regression
- Gradient Descent
- GBM Boosting

However, the result was not optimal due to the complexity of the dataset. After further review, the library PyCaret was utilized to determine the most suitable algorithm to train the model. This was the result of the PyCaret review:

Code:

```
from pycaret.regression import *  
import pandas as pd  
df = pd.read_csv("/ARR - DATASET.csv")  
s = setup(df, target='Target', session_id = 123)  
best_model = compare_models()  
best = s.compare_models()
```

Result:

	Description	Value
0	Session id	123
1	Target	Target
2	Target type	Regression
3	Original data shape	(171721, 17)
4	Transformed data shape	(171721, 23)
5	Transformed train set shape	(120204, 23)
6	Transformed test set shape	(51517, 23)
7	Numeric features	10
8	Categorical features	6
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Maximum one-hot encoding	25
14	Encoding method	None
15	Fold Generator	KFold
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	False
20	Experiment Name	reg-default-name
21	USI	6b2f

Figure 8: PyCaret Processing

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
omp	Orthogonal Matching Pursuit	4784.0885	216666096.5852	14908.4530	0.6257	2.3188	11.4354
knn	K Neighbors Regressor	4833.5457	234920824.0000	14341.8654	0.6029	1.4800	1.9754
et	Extra Trees Regressor	10271.7722	493586521.3922	21737.8768	0.0581	2.8505	8.6485
huber	Huber Regressor	9787.0269	484135339.0903	21607.9110	0.0546	2.8369	7.7367
lightgbm	Light Gradient Boosting Machine	10433.1092	496561188.0167	21825.3373	0.0482	2.8741	8.8533
xgboost	Extreme Gradient Boosting	10399.5575	507659974.4000	22032.2760	0.0336	2.8728	8.8295
gbr	Gradient Boosting Regressor	10451.9221	523177020.5224	22394.5512	0.0002	2.8763	8.8370
par	Passive Aggressive Regressor	10414.3748	523062429.4087	22394.1293	0.0001	2.8725	8.7669
dt	Decision Tree Regressor	10413.2029	523300272.7053	22397.4843	-0.0001	2.8725	8.7951
en	Elastic Net	10413.2328	523298188.6573	22397.4407	-0.0001	2.8725	8.7953
ridge	Ridge Regression	10413.2342	523298484.4205	22397.4472	-0.0001	2.8725	8.7953
rf	Random Forest Regressor	10413.4110	523311280.2475	22397.7073	-0.0001	2.8725	8.7953
lasso	Lasso Regression	10413.2351	523298390.8589	22397.4451	-0.0001	2.8725	8.7953
llar	Lasso Least Angle Regression	10413.2341	523298473.5503	22397.4470	-0.0001	2.8725	8.7953
lar	Least Angle Regression	10413.2342	523298484.4224	22397.4472	-0.0001	2.8725	8.7953
lr	Linear Regression	10413.2342	523298484.4224	22397.4472	-0.0001	2.8725	8.7953
dummy	Dummy Regressor	10413.2340	523298480.0000	22397.4471	-0.0001	2.8725	8.7953
ada	AdaBoost Regressor	10232.2617	524404541.8193	22423.7203	-0.0026	2.8498	8.9482
br	Bayesian Ridge	15770582.0380	1393969299493535.7500	31768567.9954	-2876188.0670	7.9640	11659.9894

Figure 9: PyCaret Results

According to *PyCaret*, the most suitable algorithm for training the model was *Orthogonal Matching Pursuit*. However, the results from this algorithm were not competitive. Consequently, after analyzing algorithms suitable for large and complex datasets, the *Light Gradient Boosting Machine (LightGBM)* was identified as a viable option. Upon implementation, *LightGBM* produced results that better aligned with the cost functions of *MSE* and *RMSE*, resulting in a slightly higher R^2 score.

Since this algorithm provided better results, a *Feature Importance Analysis* was conducted using the same algorithm.

Feature Importance Analysis Code:

```
import lightgbm as lgb
model = lgb.LGBMRegressor()
model.fit(X_train, y_train)
importance = model.feature_importances_
features = X_train.columns
plt.figure(figsize=(10,6))
sns.barplot(x=importance, y=features)
plt.xlabel("Feature Importance")
plt.ylabel("Feature Name")
plt.title("Feature Importance in LightGBM")
plt.show()
```

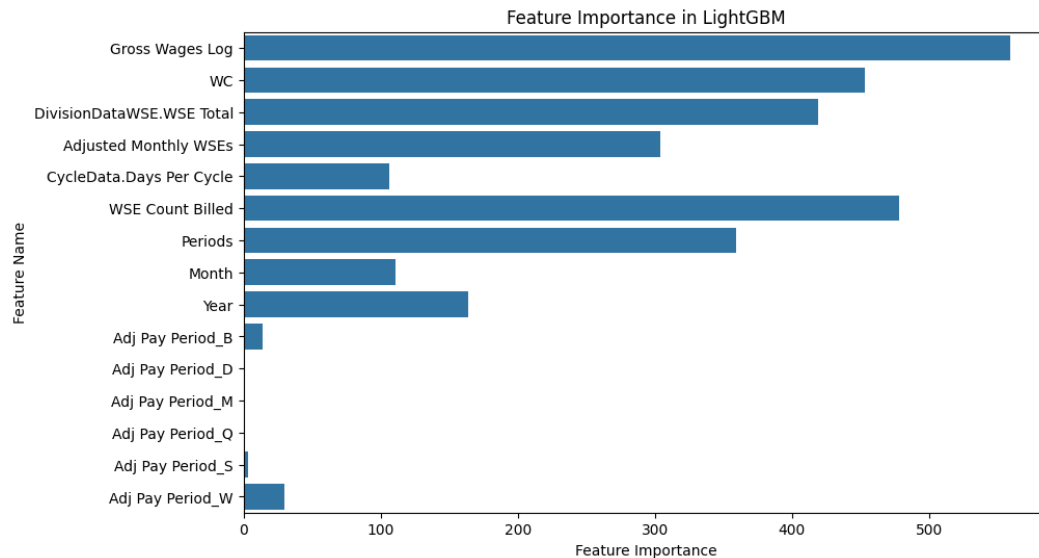


Figure 10: Feature Importance Results

Removing Low Importance Features detected by the Feature Importance Analysis

```
low_importance_features = ["AdjPayPeriod_D",  
"Adj Pay Period_M", "AdjPayPeriod_Q", "AdjPayPeriod_S"]
```

Light Gradient Boosting Machine (LightGBM-Algorithm)

```
import lightgbm as lgb  
from sklearn.model_selection import train_test_split  
from sklearn.metrics import mean_squared_error, r2_score  
import numpy as np
```

Train and split test

90% Training and 10% Testing.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state =  
100)
```

Initialize LightGBM Regressor

```
lgb_model = lgb.LGBMRegressor(  
    boosting_type = "gbdt",  
    n_estimators = 4000,  
    learning_rate = 0.005,  
    max_depth = 15,  
    num_leaves = 150,  
    min_child_samples = 30,  
    subsample = 0.9,  
    colsample_bytree = 0.9,  
    reg_alpha = 0.5,  
    reg_lambda = 0.5,  
    random_state = 100  
)  
Define callbacks callbacks = [  
    lgb.early_stopping(stopping_rounds = 50),  
    lgb.log_evaluation(period = 10)  
]  
lgb_model.fit(  
    X_train,  
    y_train,  
    eval_set = [(X_test, y_test)],  
    eval_metric = 'rmse',  
    callbacks = callbacks)
```

Predictions

```
y_train_pred = lgb_model.predict(X_train)
y_test_pred = lgb_model.predict(X_test)
Metrics
train_mse = mean_squared_error(y_train, y_train_pred)
test_mse = mean_squared_error(y_test, y_test_pred)
train_rmse = np.sqrt(train_mse)
test_rmse = np.sqrt(test_mse)
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)
```

The results provided by the LightGBM were:

- Training MSE: 140814499.05
- Test MSE: 143320735.44
- Training RMSE: 11866.53
- Test RMSE: 11971.66
- Training R²: 0.71
- Test R²: 0.62

7 Conclusions

Prediction Entry

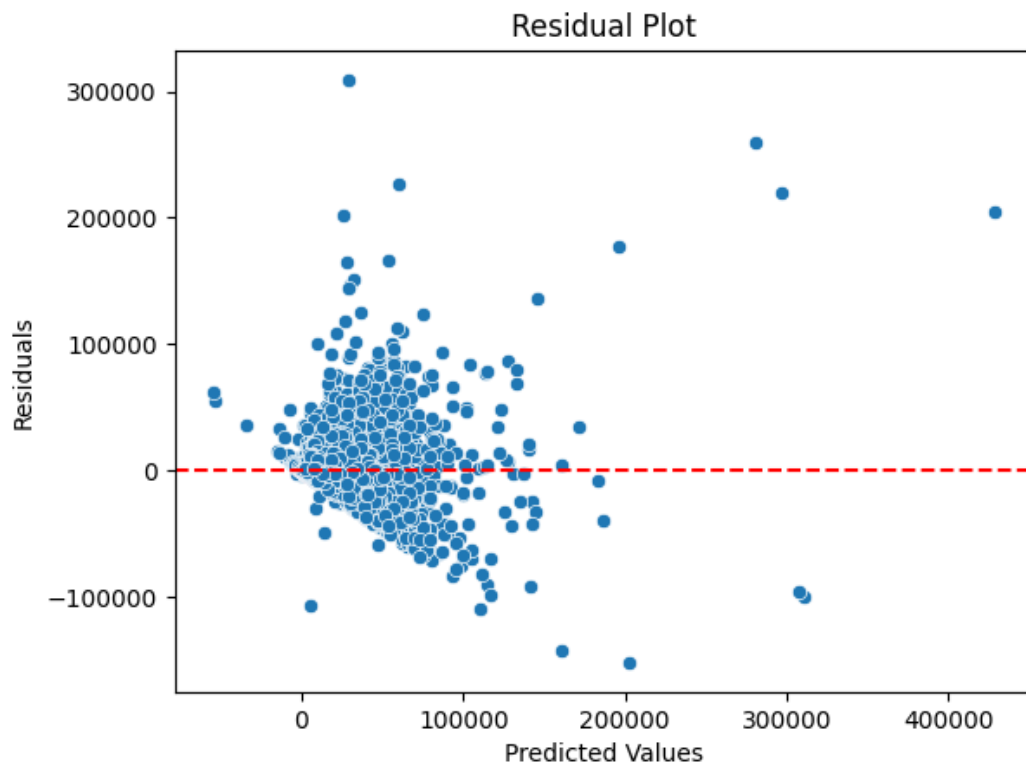
```
new_entry = pd.DataFrame(np.array([[ -0.83388, -0.149331, -0.130998, -0.166285, 0.962488,
-0.120979, -1.431828, 6, 2024, 0, 0]]) , columns=['Gross Wages Log', 'WC',
'DivisionDataWSE.WSE Total', 'Adjusted Monthly WSEs', 'CycleData.Days
Per Cycle', 'WSE Count Billed', 'Periods', 'Month', 'Year', 'Adj Pay Period'_B, 'Adj Pay Period'_W])
new_entry = new_entry[X_train.columns]
```

Prediction:

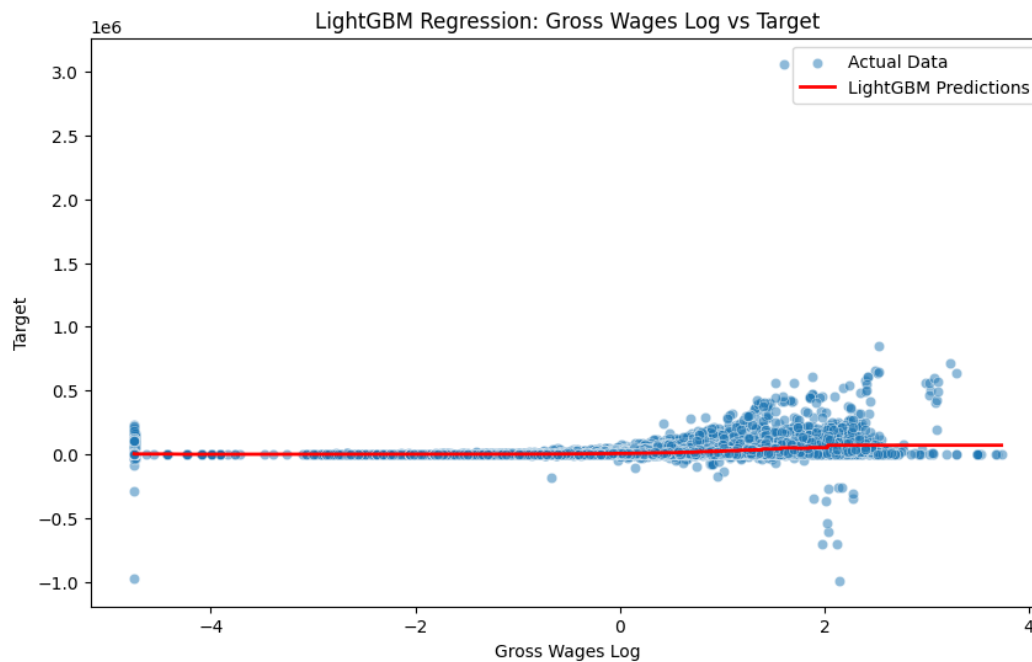
```
predicted_arr = lgb_model.predict(new_entry)print("The estimated ARR for this client is", predicted_arr)
```

Result: After testing the ARR with the input features, The estimated ARR for this client is [1032.06] compared to the original result of 1233.60. This indicates that the prediction is close to the actual target.

The model still has room for improvement. The results suggest a slight over-fitting tendency. The regression line intersects the data points, and the predictions are closely aligned with the target values.



(a) Predicted vs Residuals



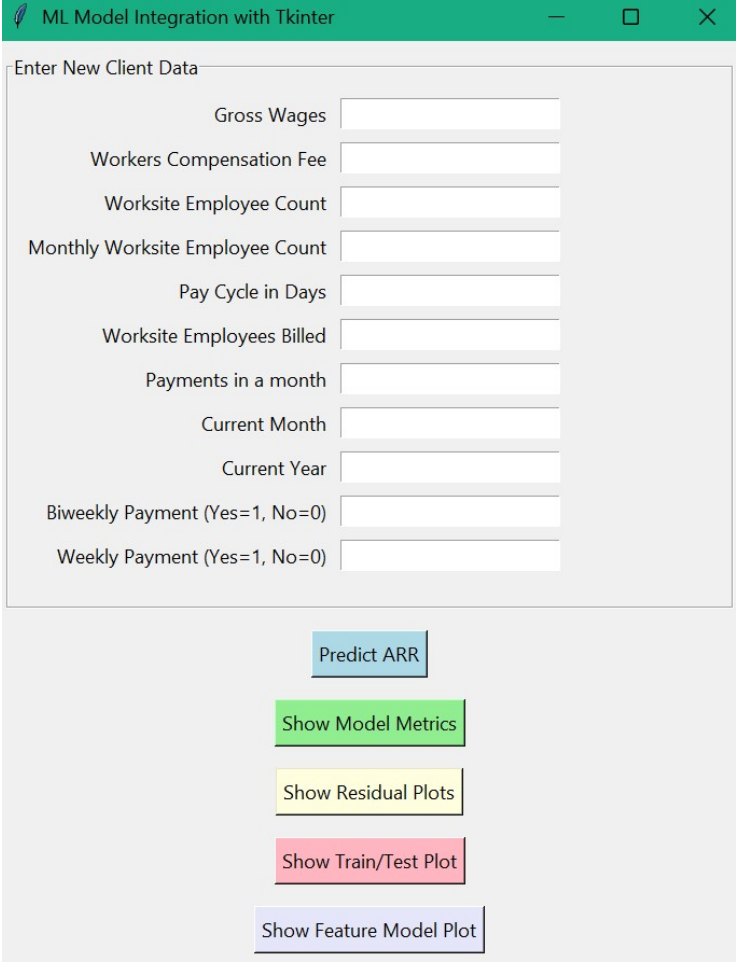
(b) Regression Line Target vs "Gross Wages Log"

Figure 11: Model Analysis

8 APP Development

To develop the executable application, the Python libraries **Tkinter** and **PyInstaller** were utilized to integrate the machine learning code into a user-friendly interface, enabling seamless input of information for the selected features.

The application is fully operational and displays the prediction output along with key performance metrics, including MSE, RMSE, R^2 , and Cross-Validation scores. Additionally, it features a section for generating regression line plots and residual plots, allowing for continuous model evaluation.



The screenshot shows a Tkinter window titled "ML Model Integration with Tkinter". The window contains a section titled "Enter New Client Data" with the following input fields:

- Gross Wages
- Workers Compensation Fee
- Worksite Employee Count
- Monthly Worksite Employee Count
- Pay Cycle in Days
- Worksite Employees Billed
- Payments in a month
- Current Month
- Current Year
- Biweekly Payment (Yes=1, No=0)
- Weekly Payment (Yes=1, No=0)

Below the input fields, there are five buttons arranged vertically:

- Predict ARR
- Show Model Metrics
- Show Residual Plots
- Show Train/Test Plot
- Show Feature Model Plot

Figure 12: Initial Application Interface

ML Model Integration with Tkinter

Enter New Client Data

Gross Wages	2000
Workers Compensation Fee	50
Worksite Employee Count	1
Monthly Worksite Employee Count	1
Pay Cycle in Days	15
Worksite Employees Billed	1
Payments in a month	2
Current Month	3
Current Year	2025
Biweekly Payment (Yes=1, No=0)	1
Weekly Payment (Yes=1, No=0)	0

Predict ARR

Prediction

The estimated ARR for this client is: 62426.22

OK

(a) Prediction 1

ML Model Integration with Tkinter

Enter New Client Data

Gross Wages	3500
Workers Compensation Fee	60
Worksite Employee Count	2
Monthly Worksite Employee Count	2
Pay Cycle in Days	15
Worksite Employees Billed	2
Payments in a month	2
Current Month	3
Current Year	2026
Biweekly Payment (Yes=1, No=0)	1
Weekly Payment (Yes=1, No=0)	0

Predict ARR

Prediction

The estimated ARR for this client is: 84723.29

OK

(b) Prediction 2

Figure 13: APP Analysis

9 References

- @articleapple1991, author = Michael W. Apple, title = *Is the New Technology Part of the Solution or Part of the Problem in Education?*, journal = *University of Dayton Review*, volume = 21, number = 1, article = 5, year = 1991, url = <https://ecommons.udayton.edu/udr/vol21/iss1/5>
- @articleke2017lightgbm, title=Lightgbm: A highly efficient gradient boosting decision tree, author=Ke, Guolin and Meng, Qi and Finley, Thomas and Wang, Taifeng and Chen, Wei and Ma, Weidong and Ye, Qiwei and Liu, Tie-Yan, journal=Advances in neural information processing systems, volume=30, pages=3146–3154, year=2017
- @articleolite2023, author = Diego Olite, F. M. and Morales Suárez, I. D. R. and Vidal Ledo, M. J., title = *Chat GPT: origen, evolución, retos e impactos en la educación*, journal = *Educación Médica Superior*, volume = 37, number = 2, year = 2023
- @bookbonaccorso2018, author = G. Bonaccorso, title = *Machine Learning Algorithms: Popular Algorithms for Data Science and Machine Learning*, publisher = Packt Publishing Ltd, year = 2018
- @miscroy2022, author = A. Roy, title = *Simulating Human Intelligence: Are We Close To It?*, howpublished = <https://becominghuman.ai/simulating-human-intelligence-are-we-close-to-it-463976a4c203>, note = Accessed: 2025-03-02
- @miscbuiltin2022, title = *What is Artificial Intelligence? How Does AI Work?*, howpublished = <https://builtin.com/artificial-intelligence>, note = Accessed: 2025-03-02
- @miscalameda2022, author = T. Alameda, title = *Machine learning: ¿qué es y cómo funciona?*, howpublished = <https://www.bbva.com/es/machine-learning-que-es-y-como-funciona/>, note = Accessed: 2025-03-02
- @microckcontent2021, author = G. Author, title = *Diseño UI y UX: ¡descubre cuál es la diferencia entre ambos!*, howpublished = <https://rockcontent.com/es/blog/ui-ux/>, note = Accessed: 2025-03-02
- @miscenterprisers2022, title = *What is Digital Transformation?*, howpublished = <https://enterpriseproject.com/what-is-digital-transformation>, note = Accessed: 2025-03-02