Google Analytics Customer Revenue Prediction

Wanning Zhou Yubing Gou Lu Qian Ning Li

December 11, 2023

Abstract

This study validates the 80/20 rule at Google's merchandise store by analyzing the revenue concentration from a select group of customers. Utilizing Kaggle's 2016-2017 dataset, we initially applied preprocessing techniques, including timestamp transformation and natural language processing, to extract vital information. Subsequently, various classification models were employed to forecast customer purchase behavior. The models demonstrated high accuracy: 0.90 for original cleaned data, 0.83 for downsampled, and 0.95 for oversampled data. A time series model was also developed, showing robust predictive capability. These findings underscore the significance of precise marketing strategies to effectively utilize customer data for optimizing revenue.

1 Introduction

The purpose of this project is to analyze a customer dataset from the Google Merchandise Store, which sells Google swag. The analysis covers the timeframe from 2016 to 2017. By understanding customer behaviors, the project aims to inform marketing strategies, customer targeting, and demand forecasting. The dataset, obtained from Kaggle [1], is utilized for two prediction models. The first model is a classification model predicting whether a customer will make a purchase based on available customer information and web traffic. The second model focuses on a time series analysis.

2 Related Work

With the advancement of machine learning and data science techniques, the use of predictive models has had a positive impact on business decision-making. This involves analyzing historical data and patterns to make predictions about customers' decisions or future outcomes.

Supervised binary classification models are widely used for predicting customer behaviors. Commonly employed models, such as Naive Bayes, Logistic Regression, K-Nearest Neighbors, Decision Tree, Random Forest, and Support Vector Machine are frequently utilized in studies and benchmarks [2]. Additionally, models like XGboost have shown superior performance, particularly in customer transaction fraud detection problems [3]. In addition to these widely used techniques, neural networks represent an effective approach for successful classification problem-solving. Examples include the backpropagation neural network (BPNN), radius basis function neural network (RBFNN), general regression neural network (GRNN), probabilistic neural network (PNN), and complementary neural network (CMTNN) [4]. Notably, the multilayer perceptron (MLP) demonstrates better performance in addressing imbalanced data issues [5]. Due to the diverse array of classification approaches available, we employed different models for the classification task.

Time series prediction has found applications in various industries, such as forecasting the daily closing prices of stocks, predicting the average daily prices of retail stores, and forecasting the quarterly unemployment rate for a state. Various time series models have been employed to address these challenges. The Autoregressive Integrated Moving Average Model (ARIMA) stands out as the most popular and widely used model [6] in time series prediction. In addition to traditional models, machine learning techniques like the support vector machine (SVM) [7] have been applied for time series forecasting. Furthermore, deep learning models, including the multi-layer perceptron (MLP), backpropagation (BP), recurrent neural networks (RNNs), convolutional neural network (CNN), and long-short term memory (LSTM), have been introduced to tackle time series prediction problems [8]. Despite the availability of these advanced models, we chose to utilize the fundamental and widely recognized ARIMA model for our time series forecasting task.

3 Data Preparation

3.1 Data Cleaning

After exploring the data, we noticed that there are 24 constant features, meaning that the values of these features are the same in all the observations and can be considered as redundant data. In other words, these features do not provide useful information in predicting the target values. In this case, we removed these 24 constant features from the dataset. Table 1 is the summary of these constant features.

socialEngagementType	device.browserVersion	device.browserSize
device.operatingSystemVersion	device.mobileDeviceBranding	device.mobileDeviceModel
device.mobileInputSelector	device.movileDeviceInfo	device.mobileDeviceMarketingName
device.flashVersion	device.language	device.screenColors
device.screenResolution	geoNetwork.cityId	geoNetwork.latitude
geoNetwork.longitude	geoNetwork.networkLocation	totals.visits
totals.bounces	totals.newVisits	trafficSource.adwordsClickInfo.criteriaParameters
trafficSource.isTrueDirect	trafficSource.adwordsClickInfo.isVideoAd	trafficSource.campaignCode

Table 1: Constant Features

We also found that more than 98% of the initial target value ”totals.transactionRevenu” are null values, meaning that the users did not make any transactions. In this case, we filled these null values with 0. Meanwhile, there exist some features that also contain more than 97% null values. Before dropping these features directly, we compared the distribution of null values when the initial target value ”totals.transactionRevenu” is not 0. However, as shown in Table 2, 5 features still have more than 96% null values. Therefore, we also removed these 5 features from the dataset.

	Null Count	**Null Percentage**
trafficSource.adwordsClickInfo.page	11061	0.960573
trafficSource.adwordsClickInfo.slot	11061	0.960573
trafficSource.adwordsClickInfo.gclld	11059	0.960399
trafficSource.adwordsClickInfo.adNetworkType	11061	0.960573
trafficSource.adContent	11372	0.987581

Table 2: Distribution of Null Values

3.2 Data Preprocessing

3.2.1 General Process

We added one categorical feature ”madeTransation” as the target value for the classification problem: we set the value as 0 when the corresponding revenue ”totals.transactionRevenue” is 0, meaning that the user did not make any transactions; otherwise, we labeled it as 1, representing the user did make a transaction.

Additionally, after taking a closer look at the data, we noticed that most users who made transactions are located in the Americas from Figure 1. More specifically, as Figure 2 shows, most of these users are from the United States. In this case, we decided to narrow down our data to only focus on users who are located in the United States for further analysis.

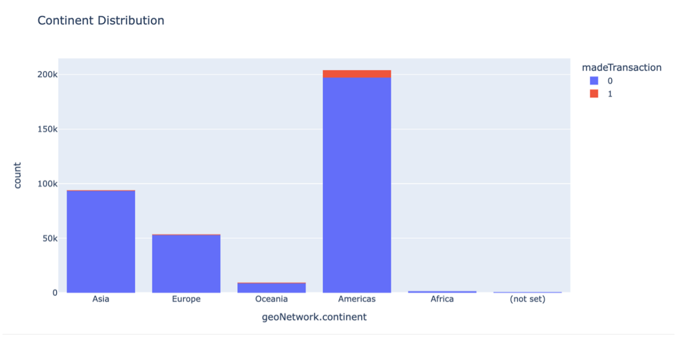

Figure 1: Distribution Among Continents

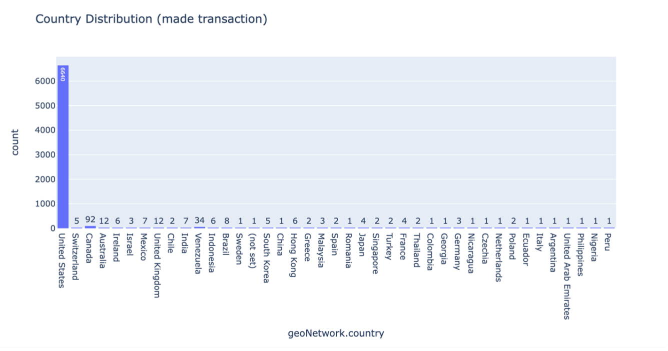

Figure 2: Distribution Among Countries

3.2.2 Add Local Timestamps

It is important to analyze whether the local time that users visit the website would affect their decision to make a transaction. However, the original data only provides a UNIX timestamp feature ”visitStartTime”, and a feature ”geoNetwork.city” which represents the city users are located in. In other words, the original data does not provide users’ local timestamps directly. To compute the local timestamps, we first utilized pandas built-in function ”pandas.Timestamp” [9] to convert the provided UNIX timestamp to UTC timestamp. Additionally, we applied a Python API - GeoPy [10] to locate the coordinates of provided cities, and then mapped them to corresponding time zones. Furthermore, we converted the UTC timestamps to users’ local timestamps based on the time zones. Extra time features, such as hour, weekday, and month were also added for future analysis. Table 3 is an example of a partial result dataset after this process.

visitStartTime	**UTC Time**	**geoNetwork.city**	**TimeZone**	**LocalTime**	**Hour**	**Weekday**	**Month**
1472843059	2016-09-02 19:04:19+00:00	Mountain View	America/Los_Angeles	2016-09-02 12:04:19.00	12	4	9
1472812649	2016-09-02 10:37:29+00:00	New York	America/New_York	2016-09-02 06:37:29.00	06	4	9

Table 3: Example with Local Timestamps

3.2.3 Natural Language Processing

We also found that there are 3 text-based features in the dataset: ”trafficSource.referralPath”, representing the URL of pages before user being directed to the current website; ”trafficSource.keyword”, representing the keywords user used to search in the searching engine; ”geoNetwork.networkDomain”, representing the domain of the user’s internet service provider or organization through which they access the internet. In order to gather more useful information from these variables for further analysis, we applied some natural language processing techniques to these features.

More specifically, for feature ”trafficSource.referralPath”, we visualized the distribution of each URL among users who made transactions. As we can see from Figure 3, most users who made transactions were referred from ”/” or ”Not found”, which does not provide much meaningful information. In this case, we removed this feature from the dataset.

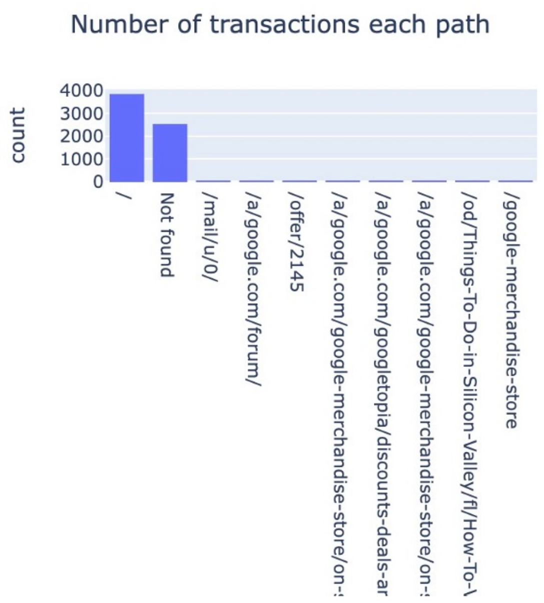

Figure 3: Distribution of Each Referral Path

For feature "trafficSource.keyword", we utilized some regular expression techniques and counted the frequency of each word. However, we noticed that users who made transactions only used a limited amount of meaningful keywords while searching (Table 4), so we also removed this feature from the dataset.

Keyword	**Frequency**
targeting	5
shirt	2
youtube	2
stickers	1
mug	1

Table 4: Keyword Frequency Analysis

For feature "geoNetwork.networkDomain", we applied a Python API - Klazify [11] to gather more information about the content of the network domains, and categorize them into Klazify's self-defined domain types. As Table 5 shows, the distribution of domain types among users who made transactions is reasonable and provides some useful information. For this reason, we decided to use the resulting domain types as part of further analysis.

Internet & Telecom 1225	Business & Industrial 39	Jobs & Education 39	Computers & Electronics 29	Travel 27	private 25
Arts & Entertainment 9	Shopping 9	Finance 8	Health 8	People & Society 6	Law & Government 5
Real Estate 5	Online Communities 4	Home & Garden 1	News 1	Reference 1	

Table 5: Distribution of Domain Types

4 Classification

4.1 Exploratory Data Analysis

4.1.1 Categorical Variables

We initially visualized the distribution of each categorical feature among all users to observe whether transactions were made or not (Figure 4 to Figure 12).

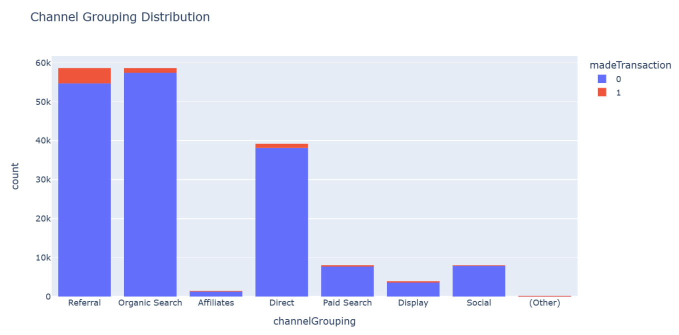

Figure 4: Channel Grouping Distribution

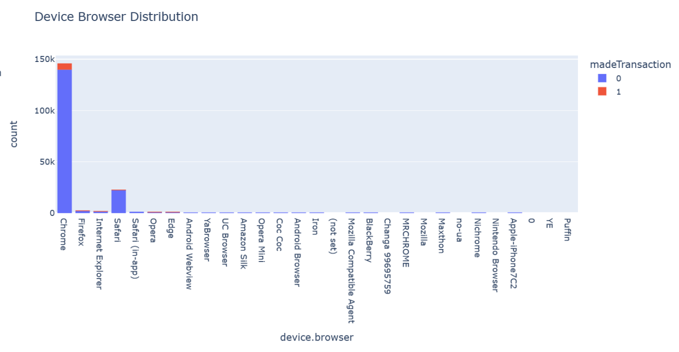

Figure 5: Device Browser Distribution

Figure 4 reveals that most users came to the store through referral and organic search, while most users who made transactions were via referral. Figure 5 shows that majority of users selected 'Chrome' and 'Safari' as their device browsers. In particular, 'Chrome' exhibited the most frequently used browser among users who made transactions.

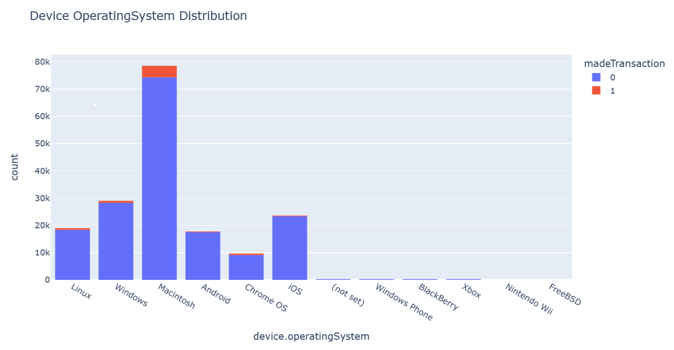

Figure 6: Device OperatingSystem Distribution

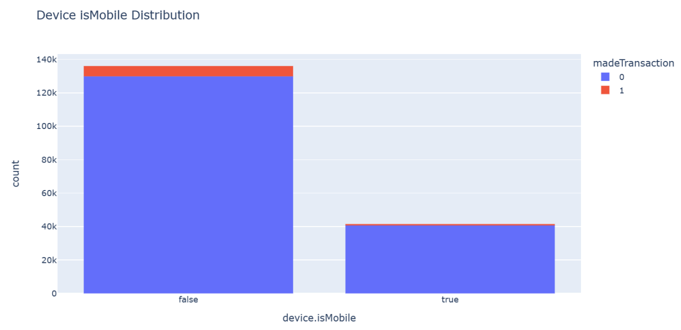

Figure 7: Device isMobile Distribution

From Figure 6, we notice that the top five device operating systems are 'Macintosh,' 'Windows,' 'iOS,' 'Linux,' and 'Android'. Notably, users who made transactions preferred 'Macintosh' as the operating system. It was also worth mentioning that the majority of users who made transactions did not use mobile devices (Figure 7).

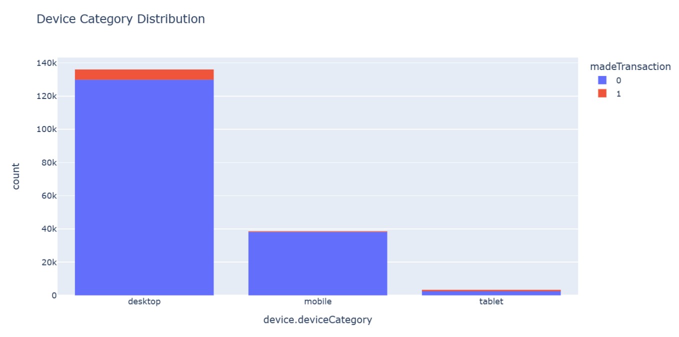

Figure 8: Device Category Distribution

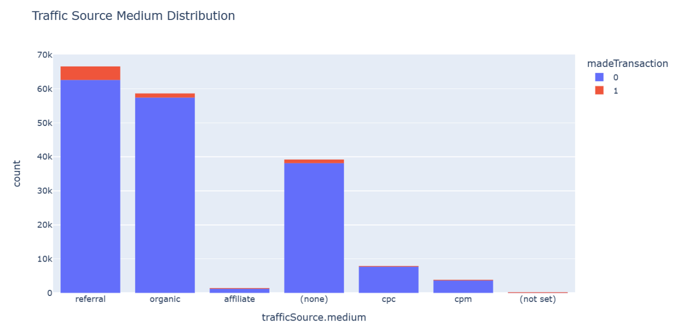

Figure 9: Traffic Source Medium Distribution

As we can see from Figure 8, the majority of users who made transactions used desktop devices. Additionally, as Figure 9 shows, most users preferred 'referral' and 'organic' for the 'Traffic Source Medium' feature. Notably, 'referral' has the highest proportion of users who made transactions.

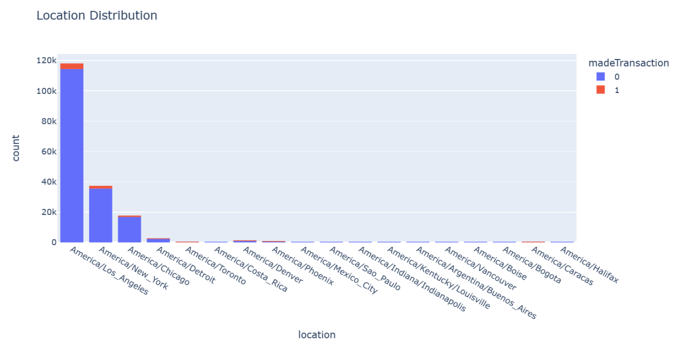

Figure 10: Location Distribution

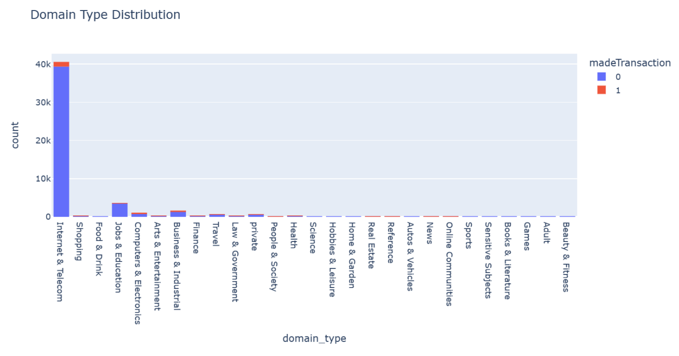

Figure 11: Domain Type Distribution

In terms of users' location, Figure 10 reveals that the majority of users were from Los Angeles, New York, and Chicago. Moreover, most users were classified as 'Internet Telecom,' 'Jobs Education,' and 'Business Industrial for their domain types (Figure 11).

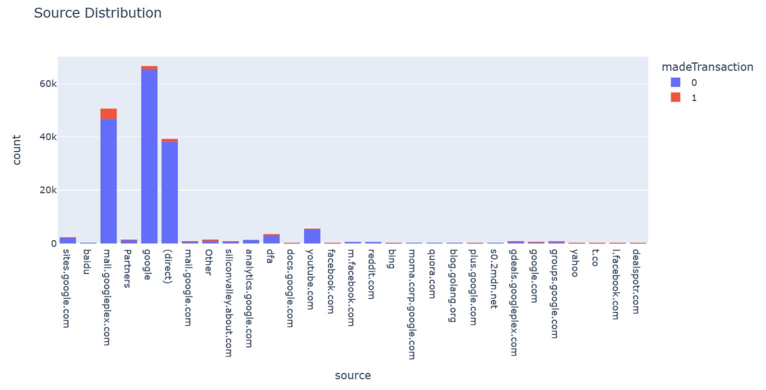

Figure 12: Source Distribution

In terms of search sources, as Figure 12 shows, the majority of users was directed from 'google,' 'mall.googleplex.com,' '(direct),' 'youtube.com,' 'dfa,' and 'sites.google.com.

4.1.2 Continuous Variables

We also observed from Figure 13 and 14 that users' visit hours to the store impacted their decision to make transactions.

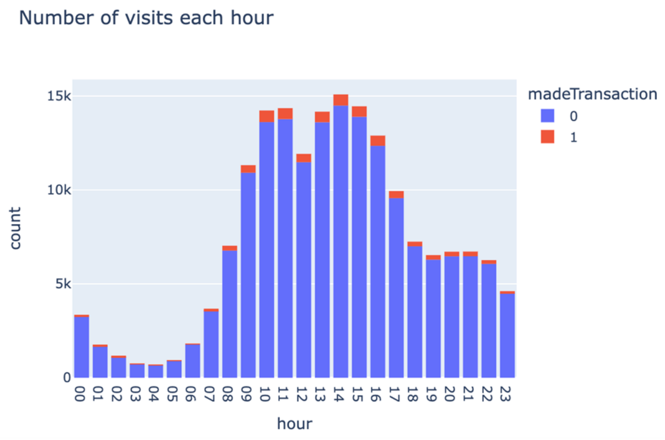

Figure 13: Number of visits each hour Distribution

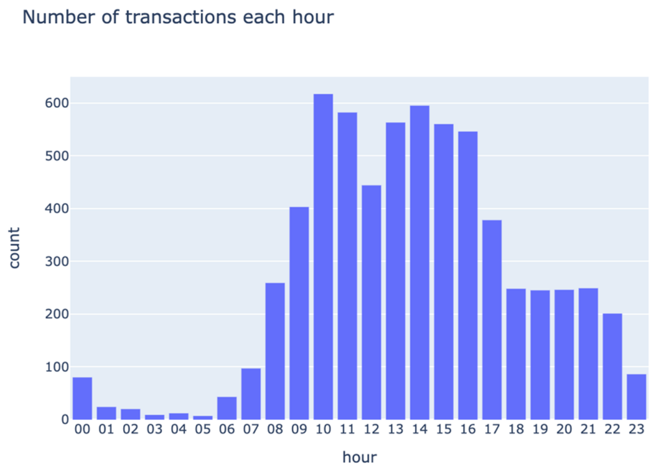

Figure 14: Number of transactions each hour Distribution

4.2 Feature Selection

4.2.1 Correlation

We applied the Label Encoder method to encode all categorical variables and generated a heatmap of the entire dataset (Figure 15).

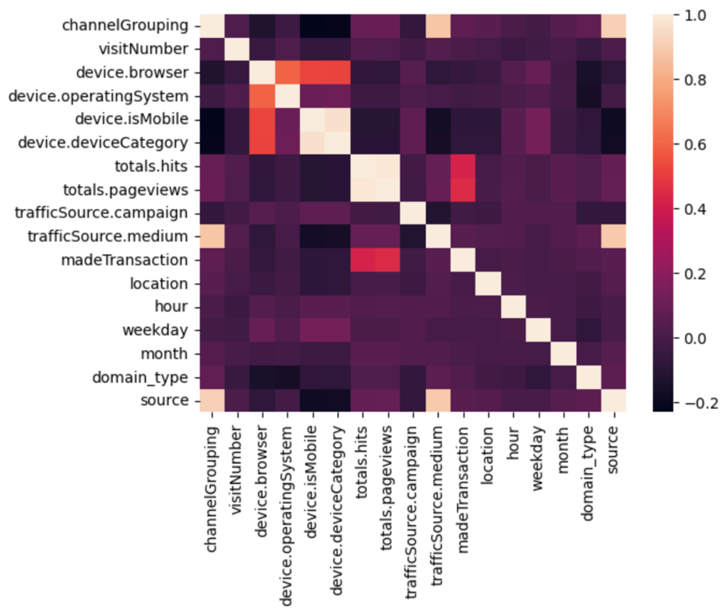

Figure 15: Correlation Heatmap

From the heatmap above, we identified that the two continuous variables, "totals.pageviews" and "totals.hits" exhibit higher correlations with our target value "madeTransaction".

4.2.2 Chi-Squared Test of Independence

Given that the relationship between the categorical features and the target value 'madeTransaction' was not evident in the correlation heatmap above, chi-squared tests of independence were conducted for all categorical features versus the target value "madeTransaction". These tests aimed to determine whether each categorical variable is related to the target value or not. The null hypothesis for a chi-squared test asserted that there is no relationship between the two variables.

As depicted in Table 6, the resulting p-values for all categorical features are less than 0.05, allowing us to reject the null hypothesis. In other words, all categorical features exhibited a relationship with the target value 'madeTransaction.' Consequently, all features would be used to train the classification models.

Variable	**p-value**
channelGrouping	0
device.browser	1.0254e-136
device.operatingSystem	0
device.isMobile	3.0711e-267
device.deviceCategory	9.1615e-266
trafficSource.campaign	1.5612e-17
trafficSource.medium	0
location	2.6204e-78
domain type	1.7598e-05
source	0

Table 6: p-value Table for All Categorical Features

4.3 Imbalanced Dataset

However, before training the classification models, we observed a significant imbalance in the dataset with only 6514 instances of transactions, while the majority of the dataset did not made transactions. To address this imbalance, we experimented with SMOTE (Synthetic Minority Oversampling Technique) for oversampling and the NearMiss Algorithm for undersampling [12]. The effects are illustrated below.

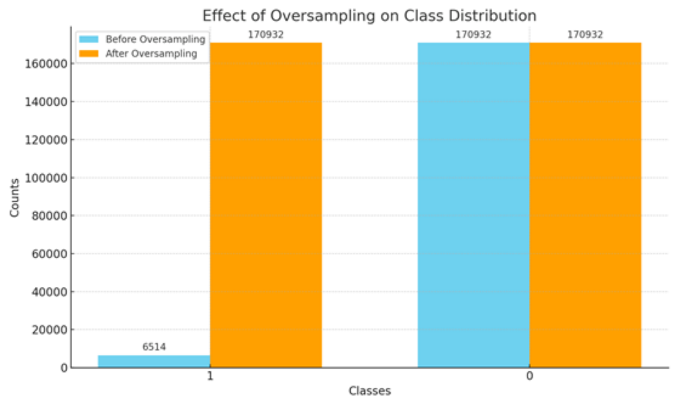

Figure 16: Effect of Oversampling on Class Distribution

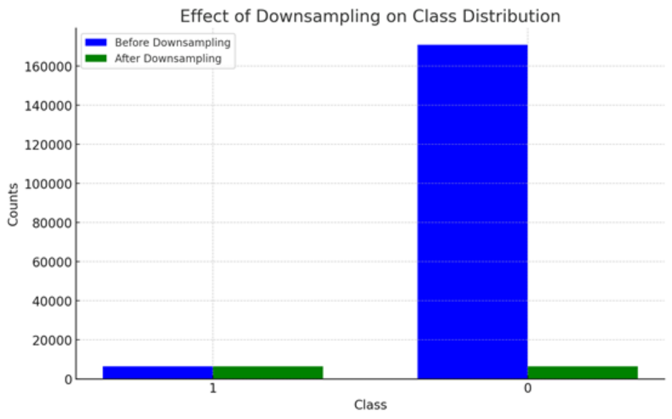

Figure 17: Effect of Downsampling on Class Distribution

4.4 Methodology

We applied different machine learning and deep learning models to classify whether a user would make transactions, including Logistic Regression model, Random Forest model, XGboost model, Support Vector Machine model, Light GBM model, and one Multilayer Perceptron model. Each model was trained using three types of dataset: the original cleaned dataset, the downsampled dataset and the oversampled dataset.

For every model, we used 70 percent of the dataset for training and 30 percent for testing the model.

4.4.1 Logistic Regression

Logistic regression is a prevalent model for binary classification due to its simplicity and efficiency. This algorithm estimates the probability of a binary response based on one or more predictor variables.

In our project, logistic regression was selected as the primary classifier. It was implemented in Python using the LogisticRegression function from the `scikit-learn` library. One advantage of logistic regression is that it does not require extensive hyperparameter tuning. However, performance can be affected by the choice of solver and regularization term. We utilized the default 'lbfgs' solver and the 'l2' regularization, which was suitable for our moderately-sized dataset.

The regularization strength, controlled by the inverse of regularization parameter 'C', was not altered from its default value since the initial model accuracy was high (0.96).

4.4.2 Random Forest

In our analysis, we applied the RandomForestClassifier from the *Scikit-learn* library, a robust ensemble learning method used for classification tasks. This method combines multiple decision trees to produce a more generalized model with reduced overfitting. The classifier was initialized with 100 trees (`n_estimators=100`) and a fixed random seed (`random_state=42`) to guarantee reproducible results.

We trained the classifier using the `fit` method on our training data (`X_train`, `y_train`). After training, we predicted the class labels for the test set (`X_test`) with the `predict` method and obtained class probabilities using `predict_proba`. We isolated the probabilities associated with the positive class for subsequent evaluation.

We assessed the model's performance using accuracy and AUROC scores. The high accuracy (0.97) and AUROC (0.96) values indicate the model's effectiveness in accurate classification and its capability in discriminating between classes.

4.4.3 XGboost

XGBoost, an ensemble learning technique, stands out due to its capability to handle complex relationships within the data. Its strength lies in its ability to harness the collective power of multiple decision trees, iteratively improving performance through boosting. The model's interpretability allows for insights into feature importance, aiding in understanding the factors influencing purchase decisions.

The model training process involved specifying parameters conducive to binary classification. The specified parameters were set as follow:

- `objective`: 'binary:logistic'. To handle a binary outcome.
- `max_depth`: 3.
- `learning_rate`: 0.1.
- `n_estimators`:100.
- `eval_metric`: auc (Area Under the Curve).

The model aims to optimize performance by focusing on the Area Under the ROC Curve, a crucial measure for binary classification models.

For robust evaluation, a five-fold cross-validation technique was employed. This process involves splitting the data into five subsets, training the model on four and validating on the fifth, repeating this process across all combinations. The model's performance, measured through accuracy metrics, revealed accuracy score of 0.93, highlighting the model's proficiency in accurately predicting customer purchase behavior, especially when trained on the oversampled data.

4.4.4 Support Vector Machine

A support vector machine (SVM) is a machine learning model to solve complex classification and regression problems, and it is one of the most commonly used machine learning models. The primary objective of SVM is to identify a hyperplane to distinguish the data points of different classes.

The SVM model is implemented from the *Scikit-learn* library, and different choices of parameters were made based on the datasets.

For the original cleaned data, which was imbalanced, the class weight of the two classes was computed and set as the model parameter, while other parameters were set as the default values. The accuracy score for the model was 0.89, which was not bad.

When fitting the model with downsample data, standardization has been applied by adding *StandardScaler* in *Scikit-learn* library. Additionally, the regularization parameter (squared L2 penalty) was added to the model. Determination of the regularization parameter was achieved by computing and visualizing the cross entropy of several models, resulting in 4.8 as the best regularization parameter. The accuracy score for this model was 0.83, which was relatively low.

Oversample data was also fitted to the model. Since the number of data is relatively big, the accuracy score of the model was the highest with default parameters compared with other two SVM models: 0.94.

4.4.5 Light GBM

The LightGBM model, a gradient boosting framework that uses tree-based learning algorithms, is known for its efficiency and high speed. It is particularly effective for large-sized data and can handle the balance of label distribution automatically.

We set the model parameters, emphasizing the binary objective for a two-class classification problem. The parameters were chosen to optimize for accuracy and to manage overfitting and computational efficiency. Key parameters included:

- `boosting_type`: 'gbdt' which stands for Gradient Boosting Decision Tree.
- `objective`: 'binary' to specify the binary log loss (or logistic regression) for binary classification.
- `metric`: 'accuracy' to focus on the classification accuracy as the performance measure.
- `num_leaves`: 31, defining the number of leaves in one tree.
- `learning_rate`: 0.05, determining the impact of each tree on the final outcome.
- `feature_fraction`: 0.9, to select a subset of features for each tree.

The model was trained using a `num_boost_round` of 100, which is the number of boosting iterations.

Post-training, we predicted the outcome for the test set and converted the probabilities to binary output using a threshold of 0.5. The accuracy of the model was evaluated, resulting in a score of 0.84, indicating high predictive performance for the task at hand.

4.4.6 Multilayer Perceptron

A Multilayer Perceptron (MLP) is a class of feedforward artificial neural network that consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training.

Our project utilizes TensorFlow's Keras API to construct an MLP for binary classification. The data preprocessing involved feature scaling using `StandardScaler` to normalize the feature space. We addressed class imbalance by computing class weights using Scikit-learn's `compute_class_weight` function and converted these weights into a format compatible with Keras. The MLP architecture comprised an input layer followed by two hidden layers with 64 neurons each, using ReLU activation, and an output layer with a sigmoid activation function for binary classification.

The model was compiled with the Adam optimizer and binary crossentropy as the loss function. It was trained on the scaled training data for 20 epochs with a batch size of 32, using class weights to adjust for imbalances and a validation split of 20%. Model performance was evaluated on the test set, achieving an accuracy of 0.90.

4.5 Model Comparison

4.5.1 Accuracy

To ensure robustness in our results, each model's accuracy was assessed across three different dataset compositions: Downsampled Data, Oversampled Data, and Original Cleaned Data.

The accuracies derived from these diverse datasets provide insight into each model's ability to generalize from the training data to unseen data, considering the challenges of class imbalance. This approach allowed us to discern not only the best-performing models but also the impact of dataset manipulation on predictive performance. The collective results are systematically presented in the accompanying Table 7, facilitating a clear and concise comparison of model efficacies under varying conditions.

Accuracy	**Downsample Data**	**Oversample Data**	**Original Cleaned Data**
Logistic Regression	0.80	0.91	0.96
Random Forest	0.83	0.98	0.97
XGboost	0.83	0.92	0.96
SVM	0.83	0.94	0.89
Light GBM	0.83	0.95	0.84
Multilayer Perceptron	0.84	0.95	0.91

Table 7: Comparison of Different Classification Models

4.5.2 Confusion Matrix

In examining the performance of the Random Forest Classifier, depicted in the Figure 18, we observe a high accuracy rate on the original cleaned dataset. However, there is a relatively low True Positive (TP) to False Positive (FP) ratio. This discrepancy is mainly attributed to the imbalanced nature of the input data. Implementing down-sampling techniques reveals an enhanced predictive capability for the minority class (Class 1), but at the expense of overall accuracy, as evidenced by the decrease in accuracy from 0.97 to 0.83 showing in the Figure 19.

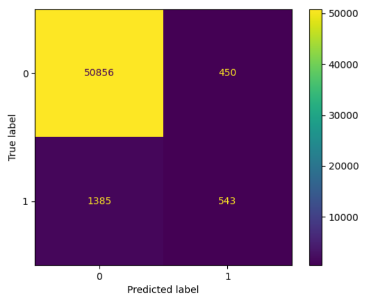

Figure 18: Original Cleaned Data

Accuracy:0.97

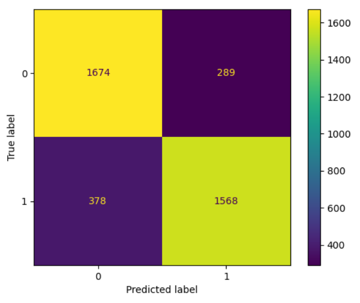

Figure 19: Downsampling Data

Accuracy:0.83

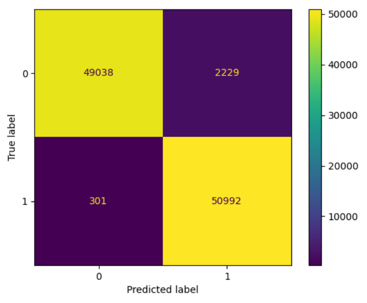

Figure 20: Oversampling Data

Accuracy:0.98

Furthermore, when evaluating the effects of over-sampling, as shown in the Figure 20, while there is a boost in accuracy to 0.98, this technique is generally not preferred. This is because it modifies the inherent distribution of the dataset, potentially introducing artificial bias. The scatter plot Figure 21 and Figure 22 clearly illustrates the over-sampling effect, with an influx of new data points in the top-right quadrant, representing an overrepresentation that does not align with real-world scenarios.

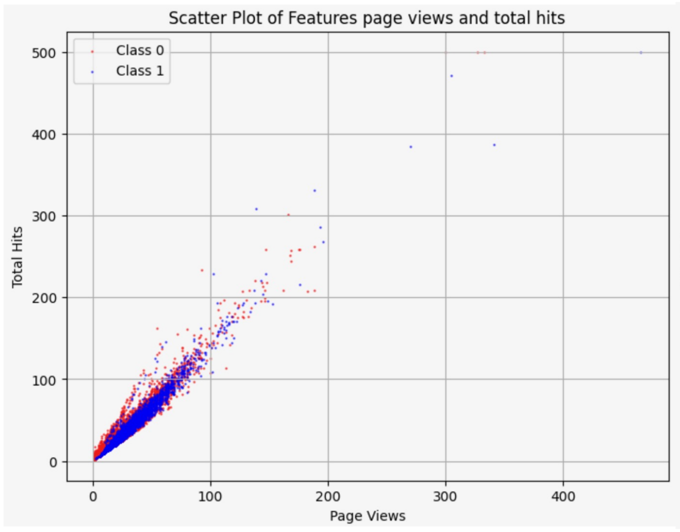

Figure 21: Scatter Plot using Original Data

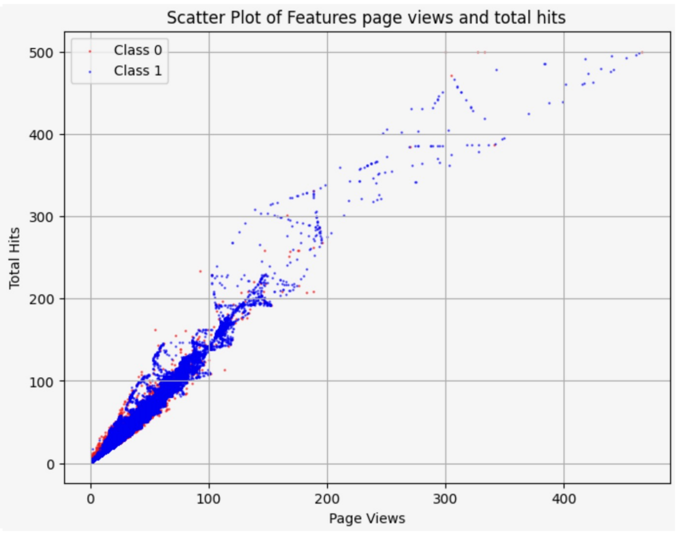

Figure 22: Scatter Plot using Oversampling Data

To address these issues, we explored alternative models capable of intrinsic class weight adjustment. As demonstrated in the third image, the Multilayer Perceptron (MLP) was configured to ’balanced’ weight mode, which assigns greater importance to the minority class (Class 1) during the training process. In an MLP, this reweighting directly influences the loss function, prioritizing the correct classification of under-represented classes. This method allows for the maintenance of a high level of model performance, evidenced by an accuracy of 0.91 for the original data, without the need to artificially alter the distribution of the original dataset.

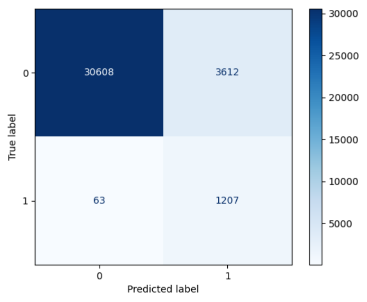

Figure 23: Original Cleaned Data

Accuracy:0.91

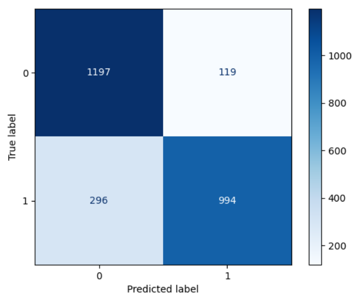

Figure 24: Downsampling Data

Accuracy:0.84

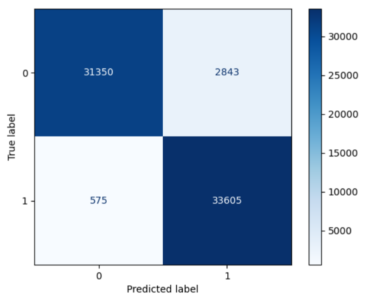

Figure 25: Oversampling Data

Accuracy:0.95

4.5.3 Features Contribution

In our model’s architecture, the contribution of individual features to the predictive power was quantified and is visually represented in the feature importance chart. Numerical attributes, particularly ’totals.pageviews’, ’visitNumber’, and ’totals.hits’, surfaced as the most consequential, with ’totals.pageviews’ being the paramount predictor by a significant margin. This underscores the critical role of user engagement, as measured by page views and site interactions, in forecasting the revenue generated by the Google store.

Temporal features like ’month’ and ’hour’ also emerged as influential factors, underscoring the temporal dynamics of user behavior and its impact on revenue. The prominence of ’month’ suggests a possible seasonality effect in purchasing patterns, while ’hour’ may reflect peak usage times, both of which are crucial for understanding time-sensitive user activity.

Other notable features include ’source’ and ’location’, indicating that the origin of traffic and the geographical context of users are insightful for predicting revenue outcomes. Additionally, system-related features such as ’device.operatingSystem’ and 'device.isMobile' have notable importance scores, indicating the significance of the user's device preference in their interaction with the store.

In contrast, 'channelGrouping', 'weekday', and 'device.deviceCategory' present a moderate influence, which may still be actionable for nuanced segmentation and targeted marketing strategies. The least impacting features, like 'domain.type', 'traffic.Source.campaign', and 'device.browser', might offer marginal insights but could be critical in a composite model that leverages a broad spectrum of subtle user behaviors and preferences.

The stratification of feature importance across the spectrum not only enhances our understanding of the determinants of revenue but also aids in refining marketing strategies, inventory management, and customer engagement tactics. By focusing on high-impact features, businesses can more efficiently allocate resources to optimize the user experience and maximize revenue.

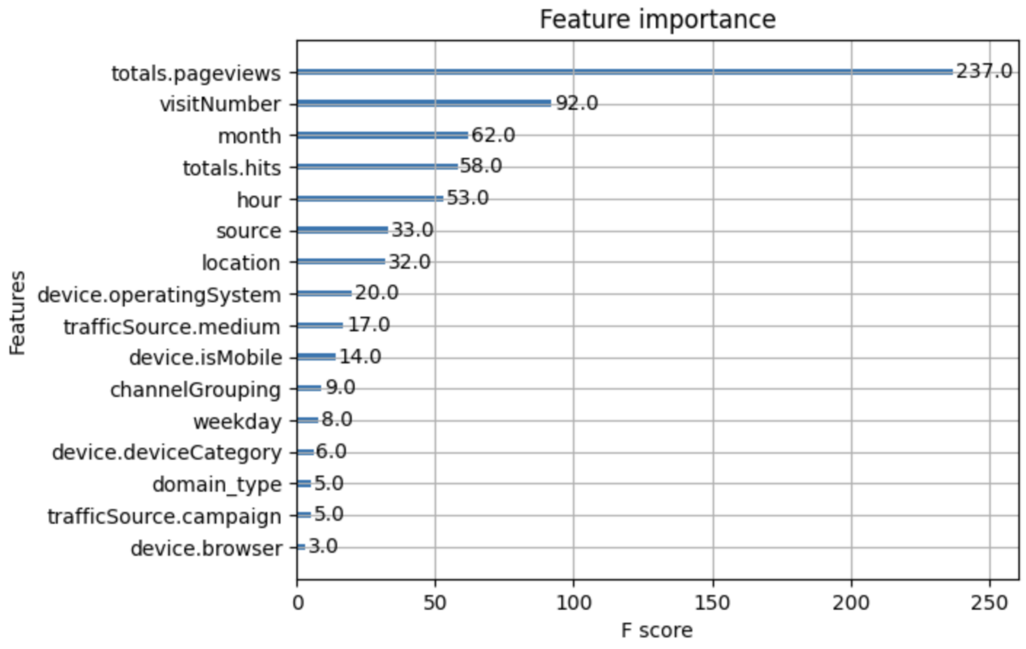

Figure 26: Features Importance

5 Time-Series Prediction

5.1 Data Processing

The objective of the second model in our analysis is to predict the daily total transaction revenue by employing the AutoRegressive Integrated Moving Average (ARIMA) model. Our dataset comprises daily transaction records for individual customers. For this model, the team focused on utilizing two essential columns: 'transactionRevenue' extracted from the 'totals' JSON blob column and the 'date' column for predictive analysis. The target variable for prediction is the daily total transaction revenue calculated by aggregating the transaction revenue for each customer, grouped by day. This aggregation provides a comprehensive view of revenue generated daily.

The dataset is transformed to compute the daily total transaction revenue, combining individual transactions to derive a holistic view for each day. Subsequently, the team divided the data into a training set encompassing the initial 11 months and a separate testing set representing the final month.

The following plots are visualization presented to facilitate a better understanding of the data:

Log of Daily Total Revenue: Visualizing the logarithm of daily total revenue for each day, offering insights into the revenue trends over time.

Daily Difference of Log Revenue: Calculated as the difference between the log revenue of consecutive days. This plot aids in observing the variations or patterns in revenue changes on a day-to-day basis.

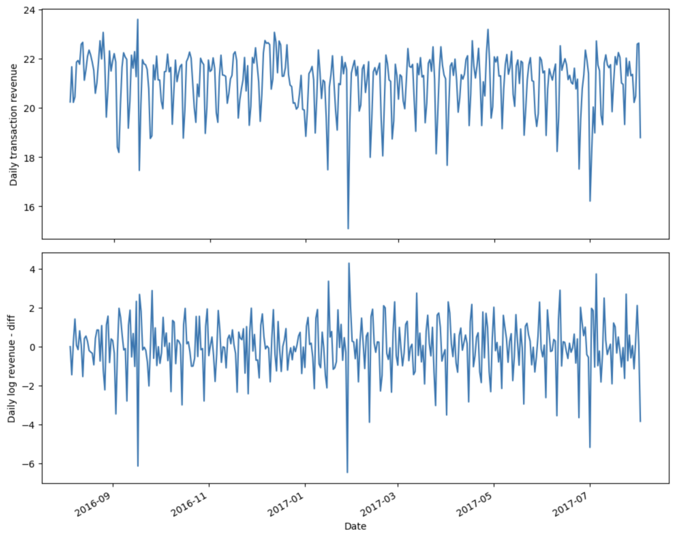

Figure 27: Top: Log of Daily Total Revenue

Bottom: Daily Differences of log revenue

5.2 Model

5.2.1 Optimize ARMA

The ARIMA model's effectiveness heavily relies on optimizing its parameters—Autoregressive (AR) order, Integrated (I) order, and Moving Average (MA) order—to achieve the best predictive performance. This subsection explores the process of optimizing the ARMA (p, q) components within the ARIMA model. To fine-tune the ARMA parameters, we experimented with various combinations of p and q values. The Akaike Information Criterion (AIC) was utilized as a metric for model selection, aiming to minimize it to attain the most suitable model fit. To fine-tune the ARMA parameters, we experimented with various combinations of p and q values, p and q values range from 1 to 4. The Akaike Information Criterion (AIC) was utilized as a metric for model selection, aiming to minimize it to attain the most suitable model fit. After exploring multiple combinations of AR and MA parameters, the model with (3, 3) as its optimal parameters, showcasing an AIC of 993.25, emerges as the preferred choice due to its superior fit.

5.2.2 Model Fit

The image shows a collection of diagnostic plots performed by a SARIMAX model (Seasonal Autoregressive Integrated Moving Average with eXogenous variables model). Here is an analysis of each of the four plots presented:

Standardized Residual Plot (Top Left):

This plot shows the standardized residuals of the model over time. Ideally, the residuals should be randomly scattered around zero, indicating that the model has captured most of the signal in the data. There should be no obvious patterns or trends in the residuals, which would suggest model inadequacies. In this plot, the residuals appear to be mostly random, suggesting the model fits well.

Histogram Plus Estimated Density (Top Right):

This plot includes a histogram and a Kernel Density Estimate (KDE) along with the normal distribution (N(0,1)) curve. The purpose is to compare the distribution of the model's residuals to a normal distribution. A good fit would show the histogram and KDE closely aligned with the normal curve. The histogram suggests the residuals are approximately normally distributed, as indicated by the fairly good fit of the KDE and normal curves to the histogram.

Normal Q-Q Plot (Bottom Left):

The Quantile-Quantile plot is used to assess whether the distribution of the residuals follows a normal distribution. If the points fall on the red line, the residuals are normally distributed. The plot shows that most points lie on the line, except for a few in the tails, which is common in real-world data.

Correlogram (Bottom Right):

It shows the autocorrelations of the residuals at different lags. If the model has captured all the autocorrelation in the data, then all points should fall within the blue band, which represents the significance threshold. In this case, most autocorrelations are within the band, indicating that the residuals are mostly uncorrelated, which is desirable in a well-fitted time series model.

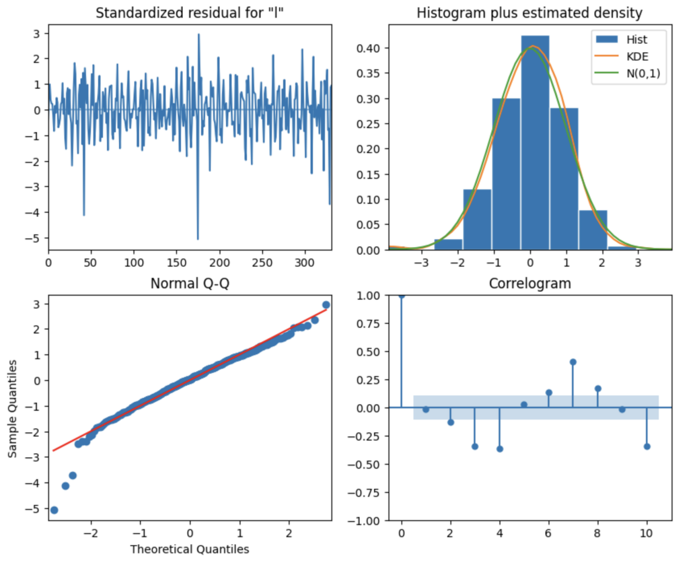

Figure 28: SARIMAX Diagnostics Plot

5.3 Prediction

The utilization of the ARIMAX model in predicting both daily revenue differences and daily log revenue for the past 30 days has yielded promising results. Through the integration of external factors alongside the time series data, the model showcased its capability to forecast these two crucial aspects of revenue trends. Visual representations in the form of line plots effectively illustrate the comparison between actual data, depicted in blue lines, and the ARIMAX predictions represented by red dotted lines. Upon inspection, the alignment between the predicted and actual values appears notably reasonable for both daily revenue differences and log revenue. This close match signifies the model's ability to capture underlying trends and fluctuations accurately. Overall, the visual assessment demonstrates the ARIMAX model's competence in incorporating external factors, leading to credible predictions that closely follow the observed patterns in revenue data over the last 30-day period.

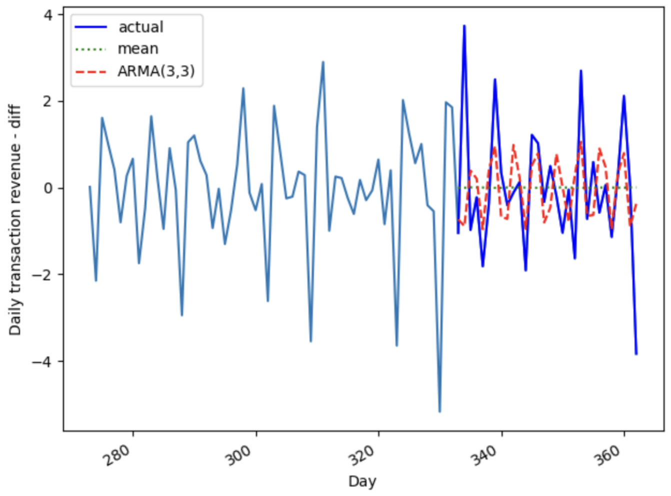

Figure 29: Daily Revenue Difference Prediction

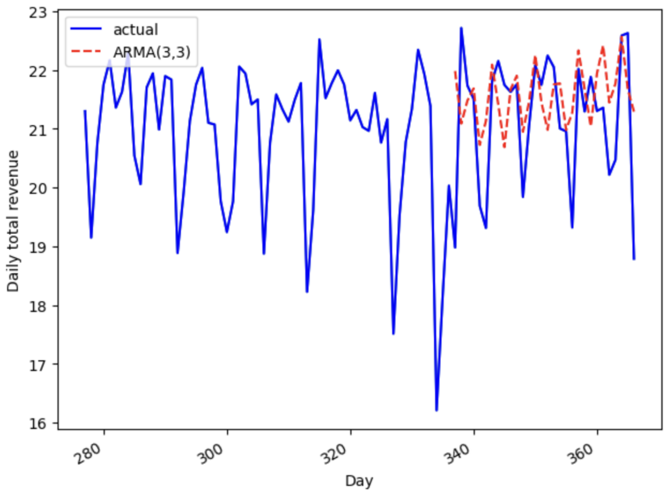

Figure 30: Daily Revenue Prediction

References

[1] (2023) Google analytics customer revenue prediction. Kaggle. [Online]. Available: https://www.kaggle.com/competitions/ga-customer-revenue-prediction/data 1

[2] H. De Oliveira, M. Prodel, and V. Augusto, "Binary classification on french hospital data: Benchmark of 7 machine learning algorithms," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2018, pp. 1743–1748. 1

[3] Y. Zhang, J. Tong, Z. Wang, and F. Gao, "Customer transaction fraud detection using xgboost model," in *2020 International Conference on Computer Engineering and Application (ICCEA)*, 2020, pp. 554–558. 1

[4] P. Jeatrakul and K. Wong, "Comparing the performance of different neural networks for binary classification problems," in *2009 Eighth International Symposium on Natural Language Processing*, 2009, pp. 111–115. 1

[5] Z. Liu, S. Zhang, W. Xiao, and Y. Di, "Cost-sensitive multi-layer perceptron for binary classification with imbalanced data," in *2018 37th Chinese Control Conference (CCC)*, 2018, pp. 9614–9619. 1

[6] S. Mou, Y. Ji, and C. Tian, "Retail time series prediction based on emd and deep learning," in *2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, 2018, pp. 425–430. 1

[7] U. Thissen, R. van Brakel, A. P. de Weijer, W. J. Melssen, and L. M. C. Buydens, "Using support vector machines for time series prediction," *Chemometrics and Intelligent Laboratory Systems*, vol. 69, no. 1, pp. 35–49, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169743903001114 1

[8] Z. Han, J. Zhao, H. Leung, K. F. Ma, and W. Wang, "A review of deep learning models for time series prediction," *IEEE Sensors Journal*, vol. 21, no. 6, pp. 7833–7848, 2021. 1

[9] pandas development team. (2023) pandas.timestamp documentation. pandas. Accessed: Dec.5, 2023. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.Timestamp.html 3

[10] GeoPy. (2023) Geopy documentation. Accessed: Dec.5, 2023. [Online]. Available: https://geopy.readthedocs.io/en/stable/ 3

[11] Klazify. (2023) Klazify - free content classification api. Accessed: Dec.5, 2023. [Online]. Available: https://www.klazify.com/ 4

[12] (2023) Ml — handling imbalanced data with smote and near miss algorithm in python. GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/ml-handling-imbalanced-data-with-smote-and-near-miss-algorithm-in-python/ 8