Using Machine Learning Techniques to Predict Purchase by Customers Based on Dynamic Pricing

Bharkavi Sachithanandam[1], Lohith Nagaraja[1]
[1]Computer Science Department, [1]San Jose State University
{*bharkavi.sachithanandam,lohithraj.surenagaraja*}@*sjsu.edu*

***Abstract*—The development of the internet has made human life sophisticated. E-commerce applications have made life easy by enabling us to purchase anything and everything online. E-commerce companies generate vast amounts of data through online sales platforms, customer loyalty programs, and other sources. Analyzing this data can provide valuable insights into customer behavior and preferences, which can be leveraged to increase sales and customer loyalty. Dynamic pricing or price optimization is the concept of offering goods at different prices, which vary according to the customer's demand. The pricing of the commodity can be done based on competitors' pricing, supply, demand and conversion rates and sales goals. However, dynamic pricing strategies in e-commerce can be complex and dynamic, making it challenging to predict customer purchase behavior accurately. This project aims to develop a predictive model that recommends optimal prices to the e-commerce website's marketing and sales teams to improve customer engagement and increase revenue. Given all the transactional data from e-commerce websites, we propose an approach using Machine Learning to dynamically vary prices by offering discounts to various customer segments. The effectiveness of the predicted pricing can then be evaluated by using another Machine Learning model to predict customer behavior after the price has been changed.**

***Index Terms*—Dynamic Price Prediction, Clustering, Kmeans Clustering, Logistic Regression, Decision trees, random forest, xgboost, support vector machines.**

I. Introduction

Large amounts of data are produced by e-commerce businesses through online sales platforms, customer loyalty programs, and other sources. Insights into client behavior and preferences can be gained through the analysis of this data, and these insights can be used to boost sales and win over more repeat business. The idea behind dynamic pricing, also known as price optimization, is to provide items at various rates that change in response to customer demand. Commodity price can be determined by looking at what competitors are charging, supply, demand, conversion rates, and sales targets. Accurately predicting client purchasing behavior is difficult due to the complexity and dynamic nature of dynamic pricing schemes used in e-commerce.The objective of this project is to create a predictive model that advises the marketing and sales teams of the e-commerce website on the best prices to charge in order to boost consumer engagement and revenue. We present a strategy employing machine learning to dynamically alter prices by providing discounts to different client categories based on all the transactional data from e-commerce websites. The success of the anticipated pricing can then be assessed by forecasting customer behavior using a different machine learning model after the price has changed.The data obtained from online resources like Kaggle is processed. The customers are grouped using the clustering algorithm based on few selected attributes and helps in customer segmentation. Additionally the data is then used to select a dynamic price for each cluster followed by training predictive models like Logistic Regression,Random Forest, XGBoost Classifiers and Accuracy,F1 Scores are used to evaluate the predictive algorithms.

The remainder of this report is organized as follows: the related work is reviewed in Section II. The dataset is described in Section III.The preprocessing and feature extraction is explained in Section IV and Section V. The clustering and dynamic pricing strategies are discussed in VI. The purchase prediction by users and brief description of the models used are explained in VII. Furthermore, results and conclusions are discussed in Section VIII and IX.

II. Related Works

[1] analyzes the dynamic pricing of Airbnb in Amsterdam using K-Means Clustering. The authors collect two types of data from the Airbnb website - Calendar data and Listing data. Based on the type of data, preprocessing is done separately for each. The preprocessing of Listing data includes feature selection, removal of unused data or missing data, data transformation and outlier removal and data visualization. The preprocessing of Calendar data involves data visualization for monthly and weekly timeframes followed by grouping the data. In both these datasets, unwanted symbols like comma(,),dollar($) etc are removed. The data is then clustered or grouped with the clustering algorithm known as K-means Algorithm with K value as 3. Once the clustering is done, a dynamic pricing model is created by combining the two datasets and evaluation of the model is done.The model is evaluated by calculating the Accuracy. The Authors reported a 96.21% accuracy.

Numburu et al. [2] propose a novel hybrid algorithm combining Extreme Gradient Boosting with Natural Gradient Boosting(X-NGBoost). This model is compared with ensemble models like XGBoost, CatBoost and LightBoost and is found to perform better than existing ensemble boosting models.The dataset is collected from Kaggle named as E-Commerce_Participants_Data. The authors perform exploratory data analysis and preprocessing on the dataset. Preprocessing involves Normalisation,logarithmic transformation to reduce the skewness of the original data, one-hot encoding

of the categorical features and grouping of like-products. The processed dataset is then used to train ensemble models like XGBoost, CatBoost and LightBoost and are used as baseline models. The X-NGBoost is then trained and hyperparameter tuning is performed. The X-NGBoost model is found to perform better than the other ensemble models. [3] proposes a dynamic pricing strategy for online retailers. The authors present an algorithm that combines online clustering and multi-armed bandit techniques. The online clustering algorithm groups customers and products into clusters based on their behavior and characteristics. The authors use an online clustering algorithm to adapt to the dynamic nature ofcustomer behavior and ensure that the clusters are up-to-date. After grouping the customers and products into clusters, the authors use a multi-armed bandit algorithm to set the optimal price for each cluster. The multi-armed bandit algorithm learns the optimal price for each cluster based on customer responses, allowing for real-time pricing adjustments. The paper has the following advantages.
It proposes a novel approach to dynamic pricing that considers the context of customers and products. It is unique compared to traditional methods that rely on fixed prices or historical data. Additionally, the authors demonstrate that their approach outperforms traditional pricing methods regarding revenue and profit. However, the study uses simulated rather than real-world data, which may or may not be an ideal methodology for real-world applications. Also, the paper focuses on using online clustering to group customers and products into clusters. However, it does not compare the performance and accuracy parameters of the proposed approach with traditional clustering methods. There are several possible extensions to this research. The first one is an in-depth study of the method for product clustering. Second, to highlight the benefit of clustering techniques for the revenue management problem with inventory constraints. Third, it will be interesting to include substitutability of products and even assortment decisions.
We also have a data-driven approach for dynamic pricing under competition on online marketplaces [4]. The authors consider a two-stage game where sellers simultaneously choose prices in the first stage and then face stochastic demand in the second stage. The goal is to develop a pricing algorithm that maximizes the seller's profit under competition. The authors develop a machine learning-based approach to predict the demand function for each product on the marketplace, incorporating the effects of both price and competitor behavior. The approach involves training a neural network model on historical data to estimate demand curves and simulate the market outcome under different pricing scenarios.
The authors then formulate the pricing problem as a Markov decision process (MDP) and use dynamic programming techniques to solve for the optimal pricing strategy. One advantage of the proposed approach is that it incorporates the effects of both price and competitor behavior. It uses machine learning techniques to estimate demand curves and simulate the market outcome under different pricing scenarios. The research also provides insights into the effects of competition and pricing dynamics on the marketplace, which can help sellers make more informed pricing decisions. However, the proposed approach assumes that sellers can access historical data on demand and competitor behavior, which may only sometimes be available in practice.
Moreover, the approach may need to generalize better to other marketplaces with different characteristics of market structures.Several possible research extensions could be pursued based on the findings and approach presented in the above research. The paper focuses on the effects of price and competitor behavior on demand. However, other factors may influence buyer behavior, such as product features, brand reputation, or customer reviews. Future research could explore the incorporation of additional factors into the demand model.

III. Dataset Description

The dataset is obtained from Kaggle Competition - Acquired Value Shoppers Challenge. The dataset zip contains different comma-separated-value(csv) files. The three datasets used for in this project are:

- transaction.csv - Contains data about the transactions made by various customers over a period of one year. This dataset is 22 GB when uncompressed and has over 350 million entries with over 3,00,000 unique customers.
- train.csv - Contains data about each customer, the incentive given to them and purchase data like the product, brand, quantity of the product and the purchase amount of the product and whether the customer is a repeated customer or not. This dataset has over 2300 unique samples. The dataset is balanced with equal number of yes and no samples for the feature - ”repeater”.
- Offer.csv - Contains details about each incentive offered to the customer.

IV. Preprocessing

A. *Offers.csv*

Offers.csv contains the details about the various incentives given to the customers.The dataset is inspected for missing and duplicate data. The dataset does not have any missing or duplicate data.

B. *train.csv*

Train.csv contains the details about the incentives given to a customer and the and purchase data like the product, brand, quantity of the product and the purchase amount of the product and whether the customer is a repeated customer or not. This dataset has over 2300 unique samples. The dataset is balanced with equal number of yes and no samples for the feature - ”repeater”. The dataset is explored for missing and duplicate values and no missing or duplicate data is found. The offer given to each customer is indicated by an offer id. This offer id column is replaced by the respective offer value by performing a join with Offers.csv based on offer id column in both the datasets.

C. *Transactions.csv*

Transactions.csv contains details about the various transactions made by 3,00,000 different customers over a period of a year with almost 350 million samples. The dataset does not have any missing data or duplicate samples. However the dataset is too large to be processed by personal computers with 16GB RAM. Thus this dataset is processed parallelly and in the form of chunks. The dataset contains customer ID on which a join is performed with the train.csv.

Libraries used:Modin Pandas,Numpy

V. Feature Extraction

The Transactions.csv is used to derive features like purchase by category (PCT), purchase by quantity (PQT), purchase by company (PCY), purchase by brand (PBD)(Collectively will be referred to as purchase metrics hereafter).Firstly, the total quantity purchased by the customer is customers, followed by total purchase amount of the customer. So these derived values are calculated by using the following formula.

$$PCT = \frac{total\_purchase\_amount}{total\_number\_of\_categories}$$

$$PQT = \frac{total\_purchase\_amount}{total\_quantity\_purchased}$$

$$PCY = \frac{total\_purchase\_amount}{total\_count\_of\_companies}$$

$$PBD = \frac{total\_purchase\_amount}{total\_number\_of\_brands}$$

All these features are extracted from transactions.csv for each customer and updated to train.csv based on the customer ID column which is used to identify each customer. *Libraries used:Numpy*

VI. Clustering and Dynamic Price Prediction

The preprocessed train.csv with the extracted features is now used to cluster the customers based on their purchase metrics. The customers are clustered into 4 groups using K-means Clustering. For each cluster, a dynamic price is determined. This price will differ for each cluster based on the purchase metrics of the customers.

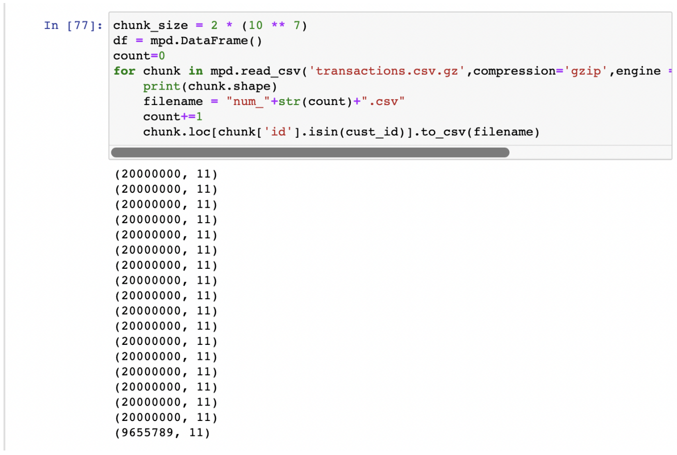
A. *K-means Clustering*

K-means clustering is an Unsupervised Algorithm,that groups data into different subsets based on their simlarities. K denotes the number of clusters into which the data has to be split in. K-means clustering is a Centroid Based algorithm, whose aim is to minimise the sum of distances between the data point for each cluster. Each data point is assigned to the closest centroid, a variance is calculated and a new centroid is placed at each cluster. Table I shows the total number of data points per cluster.

B. *Dynamic Price Prediction*

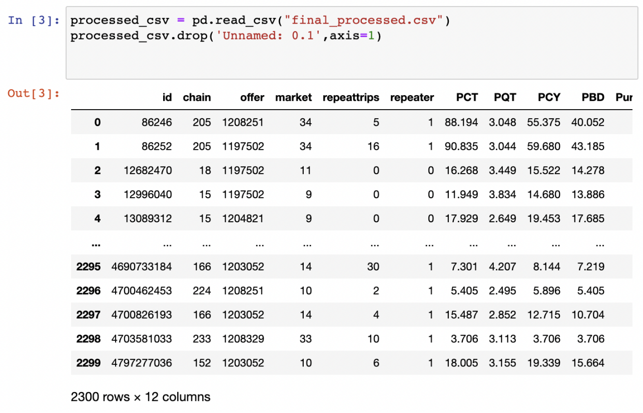
1) Common Dynamic Pricing Strategies: : Rules-based pricing, machine learning, and statistical methods are all examples of dynamic pricing. Pricing is determined by predetermined rules or conditions, such as the day of the week, the season, and the quantity requested. When demand is high or at its peak, prices may be more flexible, whereas when demand is low or at its peak, prices may be less volatile. This method offers a versatile solution to solve values dynamically using presumptions. Machine learning techniques, in contrast, evaluate enormous amounts of data using algorithms to find patterns that significantly influence ideal pricing. These algorithms take into account things like current application usage, price competition, demographic data, and previous purchasing patterns. The relationship between these elements and consumer purchasing behavior can be examined by training machine learning models on historical data, allowing them to estimate and suggest pricing that are likely to be profitable. The estimate or median of client purchase amounts from transaction records are summary facts that are widely used in statistical methods for dynamic rate assignment. These numbers show how willing consumers are to pay for a product. Charges may be established for that reason by using the suggest or median buy amount as a reference factor. In fact, businesses often use a combination of these methods to find variable pricing. By using rule-based pricing, machine learning, and statistical methods together, businesses can adjust their pricing in real time, improve revenue, and increase customer satisfaction by offering competitive and competitive pricing strategies. each person. The choice of method(s) depends on the data available, the desired level of sophistication, and the specific business needs,goals.

2) Our Approach: Statistical Dynamic Pricing: We used a statistical approach to dynamic pricing in our project because there was a lack of precise data on product demand, rival prices, or real-time activities. Our objective was to provide dynamic pricing for various user groupings based on their purchasing patterns. Initially, we used the appropriate aggregation techniques to classify our clients into focused groups. Based on consumer resemblance, company history, or other pertinent characteristics, these groupings are formed. Each group represented a collection of customers who shared common traits and purchasing tendencies. We used statistics to determine the dynamic price inside each cluster. We determined the mean purchase amount for each cluster's total number of consumers. The dynamic price for that specific cluster was based on this mean value. We attempted to capture the average purchasing power of clients inside that cluster by setting the dynamic price as the mean purchase amount. By utilizing the mean purchase amount as the dynamic price, we could cater to the purchasing behavior and preferences of each cluster. This statistical approach allowed us to account for the variations in purchase power across different groups of



```
In [77]: chunk_size = 2 * (10 ** 7)
         df = mpd.DataFrame()
         count=0
         for chunk in mpd.read_csv('transactions.csv.gz',compression='gzip',engine =
             print(chunk.shape)
             filename = "num_"+str(count)+".csv"
             count+=1
             chunk.loc[chunk['id'].isin(cust_id)].to_csv(filename)

         (20000000, 11)
         (20000000, 11)
         (20000000, 11)
         (20000000, 11)
         (20000000, 11)
         (20000000, 11)
         (20000000, 11)
         (20000000, 11)
         (20000000, 11)
         (20000000, 11)
         (20000000, 11)
         (20000000, 11)
         (20000000, 11)
         (20000000, 11)
         (20000000, 11)
         (20000000, 11)
         (20000000, 11)
         (9655789, 11)
```

(a) Parallel Loading of CSV to dataframe



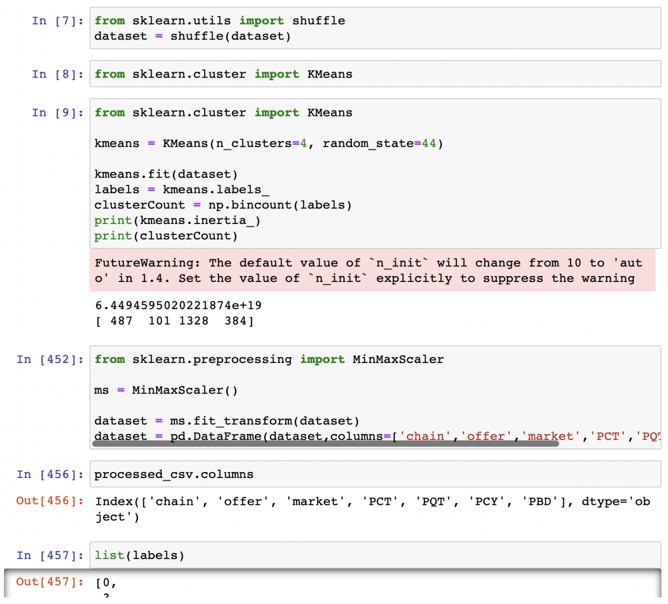
```
In [3]: processed_csv = pd.read_csv("final_processed.csv")
        processed_csv.drop('Unnamed: 0.1',axis=1)
```

Out[3]:

	id	chain	offer	market	repeattrips	repeater	PCT	PQT	PCY	PBD	Pur
0	86246	205	1208251	34	5	1	88.194	3.048	55.375	40.052	
1	86252	205	1197502	34	16	1	90.835	3.044	59.680	43.185	
2	12682470	18	1197502	11	0	0	16.268	3.449	15.522	14.278	
3	12996040	15	1197502	9	0	0	11.949	3.834	14.680	13.886	
4	13089312	15	1204821	9	0	0	17.929	2.649	19.453	17.685	
...	...	...	...	...	...	...	...	...	...	...	
2295	4690733184	166	1203052	14	30	1	7.301	4.207	8.144	7.219	
2296	4700462453	224	1208251	10	2	1	5.405	2.495	5.896	5.405	
2297	4700826193	166	1203052	14	4	1	15.487	2.852	12.715	10.704	
2298	4703581033	233	1208329	33	10	1	3.706	3.113	3.706	3.706	
2299	4797277036	152	1203052	10	6	1	18.005	3.155	19.339	15.664	

2300 rows × 12 columns

(b) Preview after feature extraction



```
In [7]:  from sklearn.utils import shuffle
         dataset = shuffle(dataset)

In [8]:  from sklearn.cluster import KMeans

In [9]:  from sklearn.cluster import KMeans

         kmeans = KMeans(n_clusters=4, random_state=44)

         kmeans.fit(dataset)
         labels = kmeans.labels_
         clusterCount = np.bincount(labels)
         print(kmeans.inertia_)
         print(clusterCount)

         FutureWarning: The default value of `n_init` will change from 10 to 'aut
         o' in 1.4. Set the value of `n_init` explicitly to suppress the warning

         6.4494595020221874e+19
         [ 487  101 1328  384]

In [452]: from sklearn.preprocessing import MinMaxScaler

          ms = MinMaxScaler()

          dataset = ms.fit_transform(dataset)
          dataset = pd.DataFrame(dataset,columns=['chain','offer','market','PCT','PQ

In [456]: processed_csv.columns

Out[456]: Index(['chain', 'offer', 'market', 'PCT', 'PQT', 'PCY', 'PBD'], dtype='ob
          ject')

In [457]: list(labels)

Out[457]: [0,
```

(c) K-means Clustering

Fig. 1: Code Snippets of Preprocessing steps

ClusterNo:	Total_No_datapoints
0	231
1	778
2	1084
3	207

TABLE I: Number of data points in each cluster

customers, enabling us to offer competitive and personalized pricing. *Libraries used:scikit-learn*

VII. Predicting Purchase by Users

In the course of the exploratory data analysis (EDA), we looked at the impact of various variables on consumer purchasing decisions. According to our investigation, the "chain" and "market" factors both had a considerable influence on purchasing decisions. Within the dataset, the "chain" factor stands in for several chains or franchises, and we saw significant differences in consumers' purchasing patterns among these chains. This implies that a customer's purchase behavior is significantly influenced by the particular chain with which they contact. Similar to the "market" component, which stands for various market locations, purchasing decisions were also clearly influenced by it. This suggests that the market environment, such as location or regional preferences, has a big impact on consumer buying habits. Additionally, in line with intuitive expectations, our analysis showed a negative correlation between "dynamic_price" and purchasing choices. Customers typically show a lower likelihood of making a purchase when the dynamic price rises. This discovery underscores the significance of pricing tactics in influencing consumer behavior and the significance of carefully selecting the appropriate price points to increase purchase likelihoods.

Furthermore, we discovered that the "offer_value" variable had a negligible influence on purchasing choices. Although the

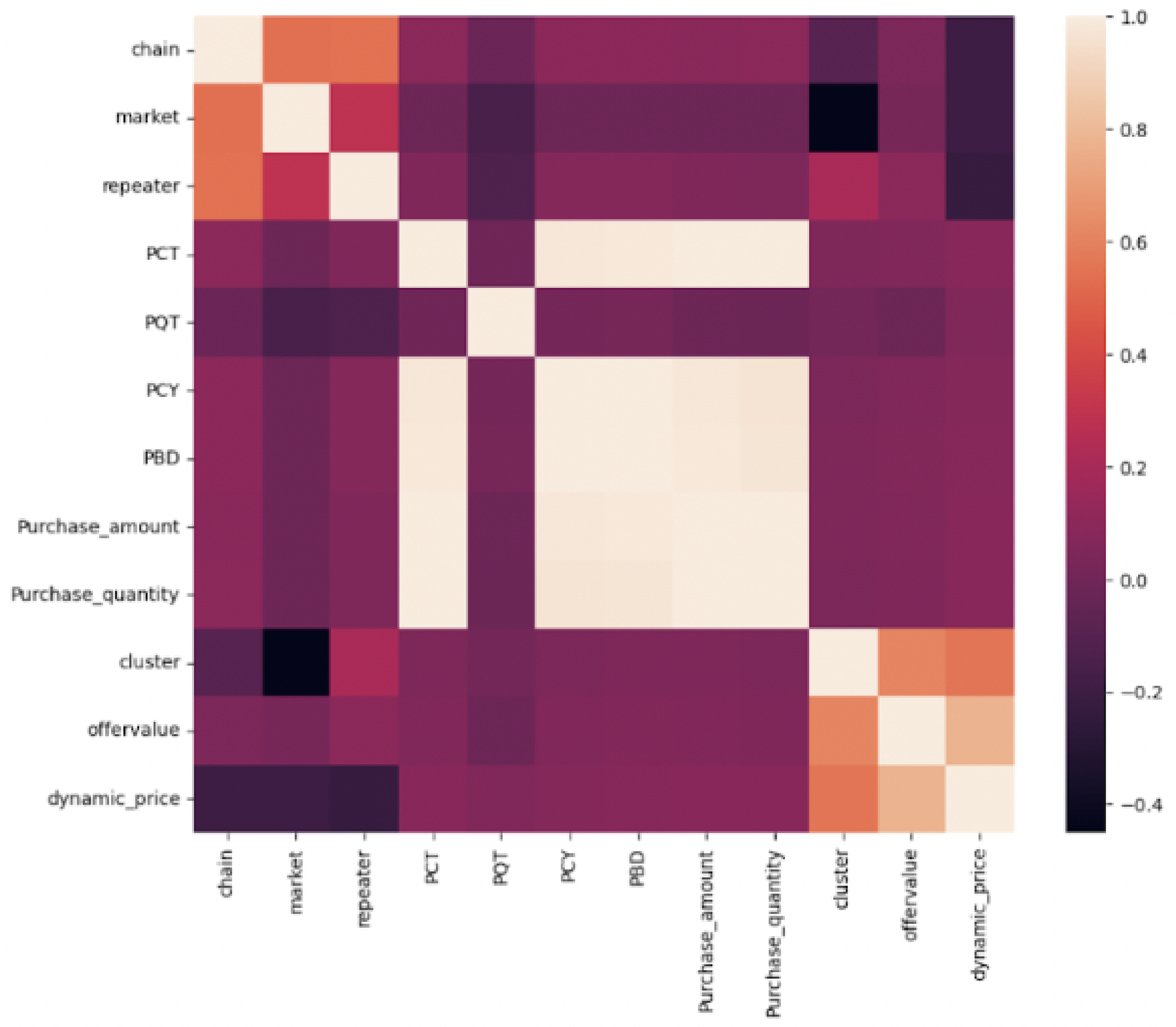


Fig. 2: Heatmap of the features

effect was not as strong as the chain, market, or dynamic price, it does indicate that promotional offers or discounts can have some impact on consumer purchasing decisions. On the precise influence and efficacy of various offer values, more research and experimentation may give further information. These EDA findings highlight how crucial it is to take chain and market dynamics into account when comprehending consumer purchasing behavior. Additionally, they stress the significance of promotional offers and dynamic pricing techniques in maximizing purchase probability. Our modeling and prediction efforts will be informed by these data, allowing us to create a more precise and reliable predictive model for client purchase behavior based on dynamic pricing.

A. *Recursive Feature Elimination*

Recursive Feature Elimination (RFE), with logistic regression as the estimator, was used to pick the features. With this method, the most important dataset features that can be used to forecast a target variable will be found. The code removes particular columns from the original dataset in order to first construct the feature matrix and target variable. The dataset is then divided into training and testing sets. The RFE algorithm is used to apply the top 7 features based on relevance to a produced instance of the logistic regression estimator. The most pertinent features are found by fitting the RFE object to the feature matrix and target variable. The feature matrix is then changed to solely keep the chosen features. This feature selection approach concentrates on the informative features for forecasting the target variable and helps minimize the dataset's dimensionality. After that, the chosen characteristics can be applied to modeling and prediction tasks.

B. Price Prediction

We used several machine learning models in our investigation to further evaluate the predicted performance. Support Vector Machines (SVMs), Decision Trees, Random Forest, XGBoost, and Logistic Regression were the models used. Each model has its own distinct qualities and advantages, enabling us to examine various facets of the data and assess their prediction power. We used Stratified K-fold cross-validation during the training phase to ensure robustness and reduce any biases in our model evaluation. This method reduced the likelihood of imbalanced splits by maintaining the proportion of samples for each class in the training folds. We used Stratified K-fold in order to get accurate and objective performance estimates for each model, which strengthened the validity of our conclusions. We were able to thoroughly evaluate the performance of many machine learning models on our dataset as well as the predictive potential of the chosen features thanks to the combination of RFE feature selection, multiple machine learning models, and stratified K-fold cross-validation. This strategy made guaranteed that our analysis was comprehensive, trustworthy, and less prone to biases and overfitting.

C. Description of models used

A brief decscription of the models used as described below.

1) Logistic Regression: :Logistic Regression is a statistical analysis method to predict a binary outcome based on past data. It analyses the relationship between the independent variables or features. It is mainly used for binary classification tasks as it predicts the probability of a particular data point belonging to a particular class.Logistic regression takes the output of linear regression function as input and uses a sigmoid function to estimate the probability for the given class.

2) Decision Trees: : Decision Tree Classifiers are a type of Supevised Algorithms that uses a set of rules to make decisions. Decision Tree works by initially selecting the best attribute using the Attribute Selection Measures to split the records. A decision is make by breaking the dataset into smaller subsets and tree building is done recursively until there are no more remaining attributes or instances.

3) Random Forest Classifier: : Random forest is an ensemble machine learning algorithm. It is widely used for classification and regression predictive modeling problems. It also has sensible heuristics for configuring these hyperparameters. Random forest ensemble is an ensemble of decision trees and a natural extension of bagging. It is an extension of bootstrap aggregation (bagging) of decision trees and can be used for classification and regression problems. In bagging, a number of decision trees are created and each tree is created from a different bootstrap sample of the training dataset. A bootstrap sample is a sample of the training dataset where a sample may appear more than once in the sample, referred to as sampling with replacement. Bagging is an effective ensemble algorithm as each decision tree is fit on a slightly different training dataset, and in turn, has a slightly different performance. Unlike normal decision tree models, such as classification and regression trees (CART), trees used in the ensemble are unpruned, making them slightly overfit to the training dataset. This is desirable as it helps to make each tree more different and have less correlated predictions or prediction errors.

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.80	0.84	0.76	0.80
Decision Trees	0.89	0.87	0.91	0.90
Random Forest	0.92	0.96	0.89	0.92
XGBoost	0.94	0.97	0.90	0.94
SVM	0.81	0.90	0.72	0.80

TABLE II: Results of all models

4) XGBoost Classifier: : XgBoost stands for Extreme Gradient Boosting.Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor's error.XGBoost is an implementation of Gradient Boosted decision trees.In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers then ensemble to give a strong and more precise model.

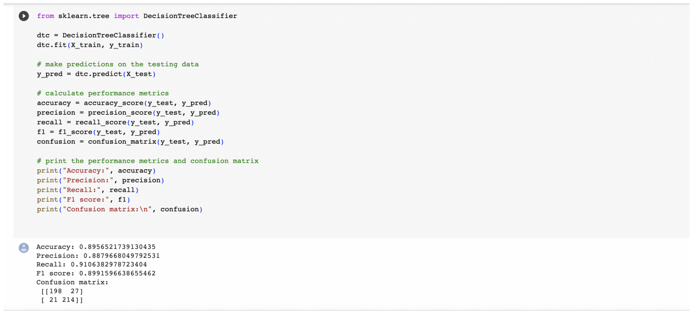
5) Support Vector Machines (SVM): : The objective of the support vector machine algorithm is to find the hyperplane in an N-dimensional space that distinctly classifies the data points. SVM is one of the most popular supervised learning algorithms used for classification. The goal of SVM algorithm is to create the best line or decision boundary that can separate the data space into classes and easily and accurately assign the new data point in the correct category in the future.SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

Libraries used:scikit-learn,matplotlib

VIII. Results

Our investigation' findings show how well various machine learning models perform when it comes to forecasting the target variable. To learn more about each model's prediction skills, we assessed its accuracy, precision, recall, and F1 score as well as the corresponding confusion matrices.

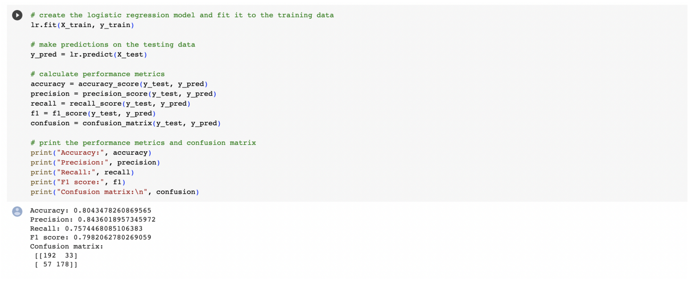
The models that were evaluated with the highest accuracy were Random Forest and XGBoost, with scores of 0.92 and 0.94, respectively. In correctly identifying the target variable, these models showed dependable performance. An outstanding precision of 0.96 was attained using Random Forest, showing a high percentage of accurately predicted positive cases. XG-



```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)

# make predictions on the testing data
y_pred = dtc.predict(X_test)

# calculate performance metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)

# print the performance metrics and confusion matrix
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 score:", f1)
print("Confusion matrix:\n", confusion)
```

```
Accuracy: 0.8956521739130435
Precision: 0.8879668049792531
Recall: 0.9106382978723404
F1 score: 0.8991596638655462
Confusion matrix:
 [[198  27]
 [ 21 214]]
```

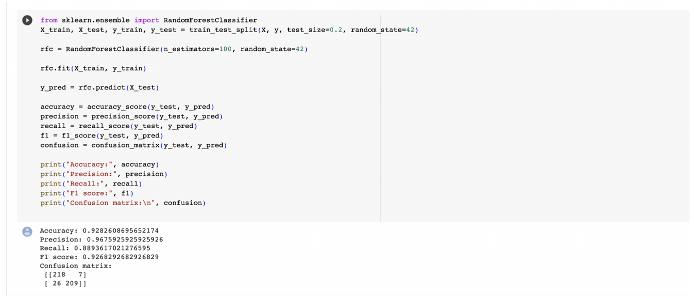
(a) Decision Tree



```
# create the logistic regression model and fit it to the training data
lr.fit(X_train, y_train)

# make predictions on the testing data
y_pred = lr.predict(X_test)

# calculate performance metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)

# print the performance metrics and confusion matrix
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 score:", f1)
print("Confusion matrix:\n", confusion)
```

```
Accuracy: 0.8043478260869565
Precision: 0.8436018957345972
Recall: 0.7574468085106383
F1 score: 0.7982062780269059
Confusion matrix:
 [[192  33]
 [ 57 178]]
```

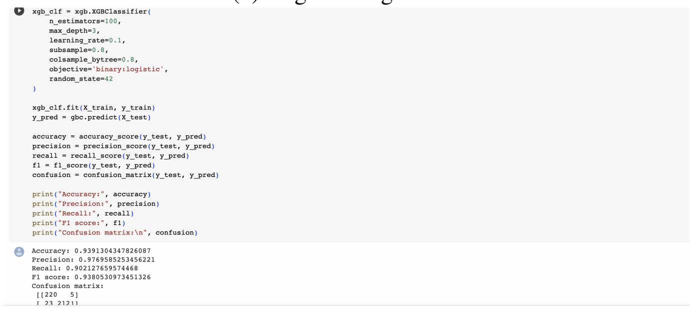
(b) Logistic Regression



```
from sklearn.ensemble import RandomForestClassifier
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

rfc = RandomForestClassifier(n_estimators=100, random_state=42)

rfc.fit(X_train, y_train)

y_pred = rfc.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 score:", f1)
print("Confusion matrix:\n", confusion)
```

```
Accuracy: 0.9282608695652174
Precision: 0.9675925925925926
Recall: 0.8893617021276595
F1 score: 0.9268292682926829
Confusion matrix:
 [[218   7]
 [ 26 209]]
```

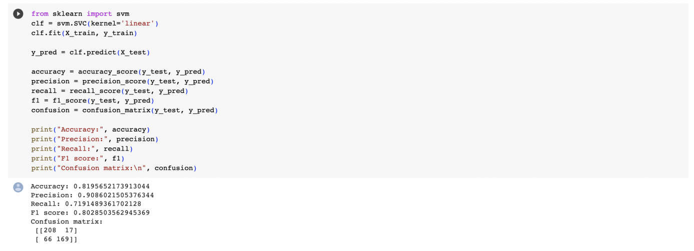
(c) Random Forest



```
xgb_clf = xgb.XGBClassifier(
    n_estimators=100,
    max_depth=3,
    learning_rate=0.1,
    subsample=0.8,
    colsample_bytree=0.8,
    objective='binary:logistic',
    random_state=42
)

xgb_clf.fit(X_train, y_train)
y_pred = gbc.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 score:", f1)
print("Confusion matrix:\n", confusion)
```

```
Accuracy: 0.9391304347826087
Precision: 0.9769585253456221
Recall: 0.902127659574468
F1 score: 0.9380530973451326
Confusion matrix:
 [[220   5]
 [ 23 212]]
```

(d) XG Booster



```
from sklearn import svm
clf = svm.SVC(kernel='linear')
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 score:", f1)
print("Confusion matrix:\n", confusion)
```

```
Accuracy: 0.8195652173913044
Precision: 0.9086021505376344
Recall: 0.7191489361702128
F1 score: 0.8028503562945369
Confusion matrix:
 [[208  17]
 [ 66 169]]
```

(e) Support Vector Machines

Fig. 3: Code Snippets from Model Training

Boost also demonstrated a high precision of 0.97, emphasizing its precise forecasts even further.

Excellent recall scores were attained by both Decision Trees and XGBoost, demonstrating their capacity to find a significant portion of true positive cases. While XGBoost scored a recall of 0.90, Decision Trees achieved a recall of 0.91. These models proved to be effective at precisely capturing the positive instances.

The precision and recall of the logistic regression model were balanced at 0.84 and 0.76, and the accuracy was decent at 0.80. When it came to categorizing the target variable, this model performed about as well as the Random Forest and XGBoost models, but not quite as well.

SVM exhibited a slightly lower accuracy of 0.81, with a precision of 0.90 and a recall of 0.72. While SVM displayed a high precision, indicating a low proportion of false positives, its recall was relatively lower, suggesting a tendency to miss some positive cases.

The confusion matrices provide additional insights into the model performance. Each confusion matrix illustrates the distribution of predicted and actual class labels, enabling a detailed examination of true positives, true negatives, false positives, and false negatives.

Overall, Random Forest and XGBoost emerged as the top-performing models in terms of accuracy and precision, indicating their strong predictive capabilities. These findings will guide us in selecting the most suitable model for predicting the target variable in our study.

IX. Conclusion and Future Work

Dynamic pricing, supported by machine learning techniques, offers businesses a powerful strategy to optimize pricing decisions and increase revenue. By leveraging real-time data analysis and considering various factors such as customer demographics, product demand, and competitor pricing, businesses can personalize their pricing strategies to maximize profitability. The results of our study highlight the effectiveness of dynamic pricing models, with Random Forest and XGBoost demonstrating high accuracy and precision in predicting customer purchase behavior.

Future research can focus on incorporating additional factors such as social media data, customer reviews, and product ratings to enhance the accuracy of dynamic pricing models.

Evaluating the impact of dynamic pricing on customer behavior, loyalty, and satisfaction can provide valuable insights for businesses. Leveraging real-time data analysis capabilities can further refine pricing strategies and adapt to changing market conditions. Integrating dynamic pricing with personalized marketing strategies can create a comprehensive customer experience, enhancing engagement and satisfaction. Ethical considerations, such as fairness and transparency in pricing, should also be explored in future research.

In conclusion, dynamic pricing combined with machine learning enables businesses to gain a competitive edge while providing value to customers. Continued research and development in this field will further optimize pricing strategies, enhance customer satisfaction, and drive business success.

Acknowledgment

Our Sincere thanks to our Professor - Dr. Mike Wu, for his continuous support and encouragement to take up this project.

References

[1] Fitrianingsih, D. A. Rahayu, and F. R. Zazila, "Dynamic pricing analytic of airbnb amsterdam using k-means clustering," in *2022 Seventh International Conference on Informatics and Computing (ICIC)*, 2022, pp. 01–07.

[2] A. Namburu, P. Selvaraj, and M. Varsha, "Product pricing solutions using hybrid machine learning algorithm," *Innovations in Systems and Software Engineering*, Jul 2022.

[3] S. Miao, X. Chen, X. Chao, J. Liu, and Y. Zhang, "Context-based dynamic pricing with online clustering," 2022.

[4] R. Schlosser and M. Boissier, "Dynamic pricing under competition on online marketplaces: A data-driven approach," 07 2018, pp. 705–714.