**A Project** Report on

Dynamic Pricing Optimization in

E-Commerce Website

Submitted to the Dept. of Information Technology, SNIST

in the partial fulfillment of the academic requirements for the award of

B.Tech (Information Technology)

Under JNTUH

By

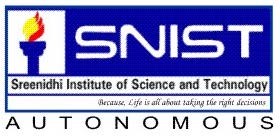
**B. Adithya Reddy (21311A12E2)**

**K. Shashi Vardhan (21311A12J0)**

**N. Shiva Sai (21311A12K0)**

Under the guidance of

**Mr. K. Niranjan Reddy (Assistant Prof., IT Dept.)**



Department of Information Technology

School of Computer Science and Informatics

Sreenidhi Institute of Science and Technology (An Autonomous Institution) Yamnampet, Ghatkesar Mandal, R. R. Dist., Hyderabad – 501301

Affiliated to

**Jawaharlal Nehru Technological University Hyderabad**

**2023-2024**

CERTIFICATE

This is to certify that the Project report on **Dynamic Pricing Optimization in E-Commerce** is a bonafide work carried out by **B. Adithya Reddy(21311A12E2), K. Shashi Vardhan(21311A12J0), N. Shiva Sai(21311A12K0)** in the partial fulfilment for the award of B.Tech degree in Information Technology, Sreenidhi Institute of Science and Technology, Hyderabad, affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH), Hyderabad under our guidance and supervision.

The results embodied in the Project work have not been submitted to any other University or Institute for the award of any degree or diploma.

**Assistant Prof., IT Dept Assoc.Prof.,IT Dept Head of the Dept**

**Mr. K. Niranjan Reddy Dr. P.N. Shiva Jyothi Dr. Sunil Bhutada**

DECLARATION

We, **B. Adithya Reddy(21311A12E2), K. Shashi Vardhan(21311A12J0), N. Shiva Sai(21311A12K0)** students of **Sreenidhi Institute of Science and Technology, Yamnampet, Ghatkesar,** studying IIIrd year IInd semester, **Information Technology** solemnly declare that the project, titled **“Dynamic Pricing Optimization in E-Commerce Website”** is submitted to **Sreenidhi Institute of Science and Technology** for partial fulfillment for the award of degree of Bachelor of technology in **Information Technology**.

It is declared to the best of our knowledge that the work reported does not form part of any dissertation submitted to any other University or Institute for award of any degree.

ACKNOWLEDGEMENTS

We thank K. Niranjan Reddy sir, giving valuable suggestions and encouragement in completing the Minor – Project work within the stipulated time.

I would like to express our sincere thanks to **Dr. T. Ch. Siva Reddy**, Principal, **Dr. Sunil Bhutada,** Professor &Head of the Department of Information Technology, **Mr. K. Niranjan Reddy** Assistant Professor of the Department of Information Technology, **Sreenidhi Institute of Science and Technology** (An Autonomous Institution), Hyderabad for permitting us to do our Project.

Finally, We would also like to thank the people who have directly or indirectly helped us and parents and friends for their cooperation in completing the Project.

B. Adithya Reddy 21311A12E2

K. Shashi Vardhan 21311A12J0

N. Shiva Shashi 21311A12K0

INDEX

|  |  |  |
| --- | --- | --- |
| **S. No** | **Description** | **Page. No** |
| **1** | Abstract | 1 |
| **2** | Introduction | 2 |
| **3** | UML Diagrams | 3-5 |
| **4** | Steps to understand and run the jupyter notebook for dynamic pricing optimization in e-commerce | 6-7 |
| **5** | Code explanation and steps | 8-10 |
| **6** | Source code | 11-23 |
| **7** | Output | 24 |
| **8** | Requirements | 25 |

**9** Conclusion 26

**10** Future Scope27

**11** Bibliography 28

**ABSTRACT**

This project focuses on developing a dynamic pricing model for ecommerce products, aimed at optimizing pricing strategies to enhance profitability. We began by analyzing a dataset containing key variables such as visitor numbers, purchases, product categories, and historical pricing. Through exploratory data analysis and visualization, we identified significant patterns and correlations. Leveraging these insights, we implemented a dynamic pricing strategy that adjusts prices based on demand and supply multipliers derived from percentile calculations.

To predict the adjusted prices, we trained a RandomForestRegressor model. Initial results were promising, but we further refined the model using hyperparameter tuning with GridSearchCV, which significantly improved its performance. The final model demonstrated high accuracy with an R² score of 0.94, alongside a Mean Absolute Error of 51.13 and a Mean Squared Error of 5744.00. This refined model not only provides accurate price predictions but also facilitates a more strategic approach to pricing, ultimately contributing to enhanced profitability and more effective decision-making in ecommerce.

**INTRODUCTION**

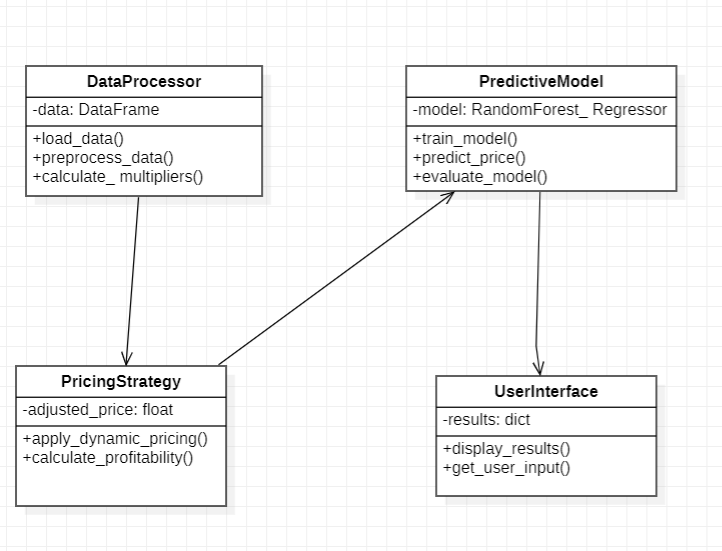
In the competitive realm of e-commerce, optimizing pricing strategies is essential for enhancing profitability and maintaining market competitiveness. Dynamic pricing, which involves adjusting prices based on real-time demand and supply metrics, has emerged as a critical approach for achieving these goals. This project aims to develop and implement a dynamic pricing strategy using a dataset named ecommerce\_pricing.csv, which contains comprehensive information on visitor counts, purchase numbers, product categories, customer loyalty, and historical prices.

The dataset provides a rich source of information to analyze patterns and trends that influence purchasing behavior and pricing dynamics. By leveraging this data, the project seeks to create a dynamic pricing model that adapts to fluctuations in demand and supply. This involves calculating demand and supply multipliers based on percentile thresholds, defining price adjustment factors, and evaluating their impact on profitability.

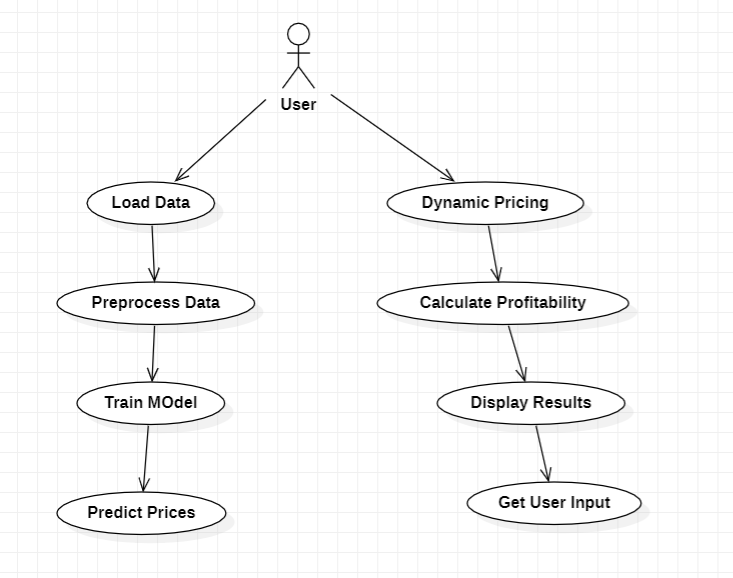
In addition to developing the pricing strategy, the project includes training a predictive model to forecast adjusted prices using machine learning techniques. The RandomForestRegressor algorithm is employed, with hyperparameter tuning through GridSearchCV to enhance model accuracy. This predictive capability will support decision-making by providing forecasts that reflect the potential impact of various pricing adjustments. The ultimate goal is to refine pricing strategies and improve overall profitability through data-driven insights and advanced analytical methods.

**UML DIAGRAMS**

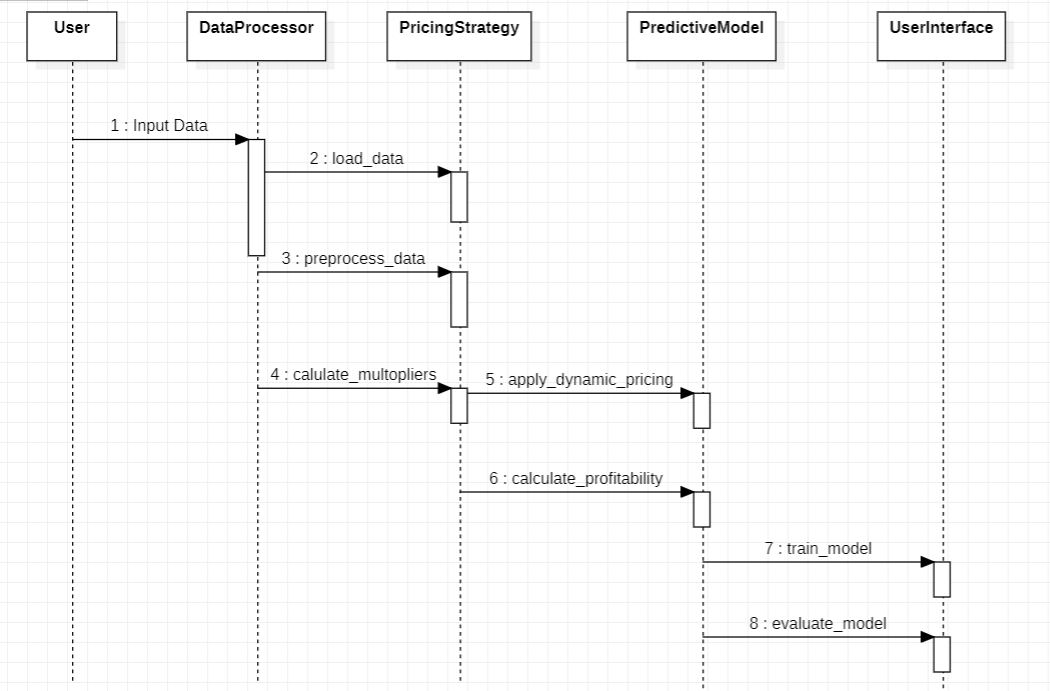
**CLASS DIAGRAM:**



**USE CASE DIAGRAM:**

****

**SEQUENCE DIAGRAM:**

****

**STEPS TO UNDERSTAND AND RUN THE JUPYTER NOTEBOOK FOR DYNAMIC PRICING OPTIMIZATION IN E-COMMERCE**

**1. Set Up Jupyter Notebook:**

* Open Jupyter Notebook on your local machine or through a cloud service.
* Create a new notebook or upload your existing one.

**2. Import Necessary Libraries:**

* Ensure you have the required Python libraries installed, such as Pandas, NumPy, Plotly, Scikit-learn, and Matplotlib.
* Import these libraries in the first few cells of your notebook.

**3. Load and Preprocess Data:**

* Upload your dataset (ecommerce\_pricing.csv) to your working directory.
* Load the dataset into a Pandas DataFrame.
* Check for missing or inconsistent data and perform any necessary data cleaning.

**4. Conduct Exploratory Data Analysis (EDA):**

* Visually explore the dataset to understand relationships between variables.
* Create plots like scatter plots, box plots, and correlation matrices to identify patterns.

**5. Implement Dynamic Pricing Strategy:**

* Calculate demand and supply multipliers based on visitor and purchase data.
* Adjust prices dynamically using the multipliers and predefined thresholds.
* Analyze the profitability of products under dynamic pricing.

**6. Train a Predictive Model:**

* Select relevant features and the target variable for your model.
* Split the dataset into training and testing sets.
* Train a RandomForestRegressor model using the training data.

**7. Evaluate the Initial Model:**

* After training, evaluate the model using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R²).
* Analyze the predictions to understand the model’s initial performance.

**8. Perform Hyperparameter Tuning:**

* Use GridSearchCV to tune hyperparameters for the RandomForestRegressor.
* Train the model with the best parameters found through the grid search.

**9. Evaluate the Tuned Model:**

* Evaluate the tuned model's performance using the same metrics (MAE, MSE, R²).
* Compare the results with the initial model to assess improvements.

**10. Visualize Results:**

* Create visualizations to compare actual vs. predicted prices.
* Use scatter plots and other relevant charts to illustrate the model's effectiveness.

**11. Save or Export Your Work:**

* Save the final model and any important results.
* Export the notebook or download any necessary files for future reference.

**12. Execute the Notebook:**

* Run all cells in the notebook to execute the entire workflow and generate the final results.

**CODE EXPLANATION AND STEPS**

**Data Overview**

The initial steps involve loading and previewing the dataset to understand its structure and content.

**Loading Data:**

This reads the CSV file containing e-commerce pricing data into a Pandas Data Frame

**Preview of Data:**

Displays the first five rows of the dataset, which includes columns like Number\_of\_Visitors, Number\_of\_Purchases, Product\_Category, etc.

**Descriptive Statistics:**

Provides a summary of the numerical columns, showing statistics like mean, standard deviation, min, and max values.

**Data Visualization**

Visualization techniques are used to explore relationships between variables.

**Scatter Plot (Number of Purchases vs. Historical Price):**

Creates a scatter plot to visualize the relationship between the number of purchases and historical prices, with an overlaid trendline.

**Box Plot (Historical Price by Product Category):**

Displays the distribution of historical prices across different product categories using a box plot.

**Correlation Matrix:**

Visualizes correlations between numerical variables using a heatmap.

**Implementing Dynamic Pricing Strategy**

Dynamic pricing strategies are calculated based on demand and supply factors.

**Demand and Supply Multipliers:**

These steps calculate demand and supply multipliers based on the percentiles of Number\_of\_Visitors and Number\_of\_Purchases.

**Price Adjustment:**

Adjusts historical prices based on the demand and supply multipliers, resulting in a dynamically calculated price.

**Profit Percentage:**

Calculates the profit percentage comparing the adjusted price to the historical price.

**Profitability Visualization:**

Creates a donut chart to show the distribution of profitable and loss products based on the dynamic pricing.

**Model Training**

A RandomForestRegressor is trained to predict adjusted prices.

**Feature and Target Preparation:**

Selects features and the target variable (adjusted\_price) for model training.

**Model Training:**

Trains a RandomForestRegressor using the prepared training data.

**Model Evaluation**

The model's performance is evaluated on test data.

**Evaluation Metrics:**

Mean Absolute Error (MAE): 259.9664692542472

Mean Squared Error (MSE): 320094.9289621058

R-squared Score (R2): 0.6328757510276888

Calculates MAE, MSE, and R² scores to assess the model's accuracy.

**Data Preprocessing for Hypertuning**

Further data preprocessing is done to prepare for hyperparameter tuning.

**Preprocessing Pipeline:**

Uses StandardScaler and OneHotEncoder within a ColumnTransformer to preprocess numerical and categorical features.

**Hyperparameter Tuning with GridSearchCV**

Hyperparameters of the RandomForest model are optimized using GridSearchCV.

**Grid Search:**

Conducts a grid search to find the best hyperparameters for the RandomForestRegressor.

**Final Model Training:**

Trains a RandomForestRegressor using the best parameters identified by the grid search.

**Final Model Evaluation**

The model's performance is evaluated on test data.

**Evaluation Metrics:**

Mean Absolute Error (MAE): 51.126310015651114

Mean Squared Error (MSE): 5743.99861858873

R-squared Score (R2): 0.9383067825136141

Calculates MAE, MSE, and R² scores to assess the model's accuracy.

**SOURCE CODE**

**import** pandas **as** pd

**import** numpy **as** np

**import** pandas **as** pd

**import** plotly.express **as** px

**import** plotly.graph\_objects **as** go

data **=** pd**.**read\_csv("ecommerce\_pricing.csv")

print(data**.**head())

Number\_of\_Visitors Number\_of\_Purchases Product\_Category \

0 152 36 Electronics

1 229 72 Fashion

2 142 59 Electronics

3 64 79 Electronics

4 156 60 Fashion

Customer\_Loyalty\_Status Number\_of\_Past\_Purchases Average\_Rating \

0 Silver 9 1.05

1 Bronze 43 1.34

2 Gold 1 1.83

3 Bronze 12 1.11

4 Gold 39 1.73

Time\_of\_Purchase Discount\_Offered Historical\_Price

0 Night 19 614.23

1 Evening 23 718.99

2 Evening 49 279.90

3 Night 48 419.41

4 Morning 31 130.67

print(data**.**describe())

Number\_of\_Visitors Number\_of\_Purchases Number\_of\_Past\_Purchases \

count 300.000000 300.000000 300.000000

mean 178.726667 50.673333 24.640000

std 73.831943 27.201834 14.366805

min 50.000000 5.000000 0.000000

25% 111.000000 27.000000 12.000000

50% 179.000000 51.500000 26.000000

75% 243.250000 71.000000 36.250000

max 299.000000 99.000000 49.000000

Average\_Rating Discount\_Offered Historical\_Price

count 300.000000 300.000000 300.000000

mean 3.067767 25.143333 496.519567

std 1.192290 15.176645 288.794723

min 1.030000 0.000000 20.920000

25% 2.022500 12.000000 235.447500

50% 3.170000 25.000000 504.765000

75% 4.152500 39.000000 739.627500

max 4.990000 49.000000 996.730000

fig **=** px**.**scatter(data, x**=**'Number\_of\_Purchases',

y**=**'Historical\_Price',

title**=**'Number of Purchases vs. Historical Price',

trendline**=**'ols')

fig**.**show()

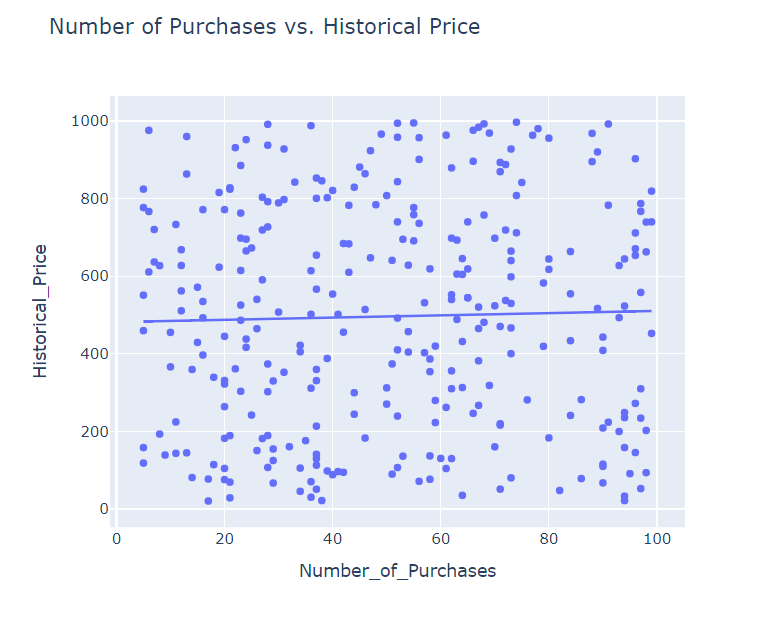


fig **=** px**.**box(data, x**=**'Product\_Category',

y**=**'Historical\_Price',

title**=**'Historical Price Distribution by Product Category')

fig**.**show()



numeric\_data **=** data**.**select\_dtypes(include**=**[np**.**number])

corr\_matrix **=** numeric\_data**.**corr()

fig **=** go**.**Figure(data**=**go**.**Heatmap(z**=**corr\_matrix**.**values,

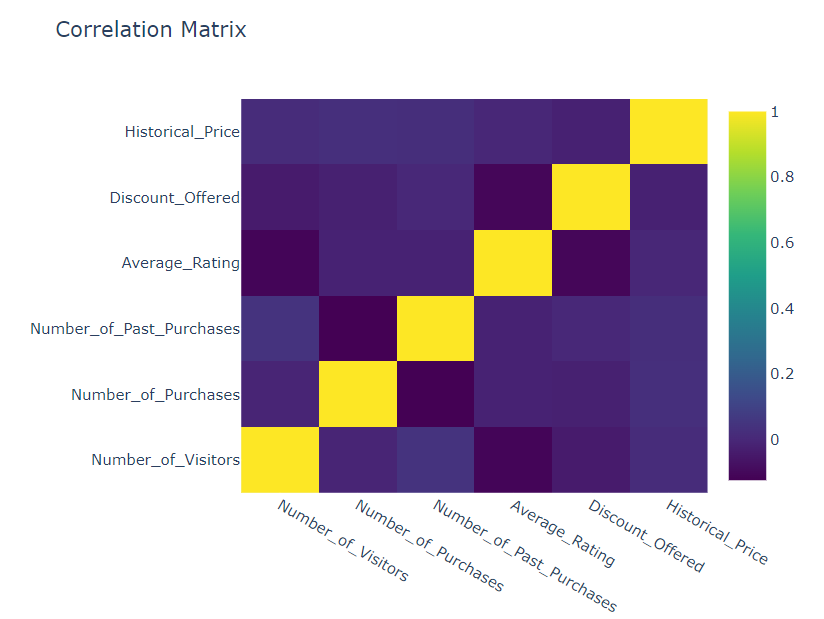
x**=**corr\_matrix**.**columns,

y**=**corr\_matrix**.**columns,

colorscale**=**'Viridis'))

fig**.**update\_layout(title**=**'Correlation Matrix')

fig**.**show()



high\_demand\_percentile **=** 75

low\_demand\_percentile **=** 25

data['demand\_multiplier'] **=** np**.**where(data['Number\_of\_Visitors'] **>** np**.**percentile(data['Number\_of\_Visitors'], high\_demand\_percentile),

data['Number\_of\_Visitors'] **/** np**.**percentile(data['Number\_of\_Visitors'], high\_demand\_percentile),

data['Number\_of\_Visitors'] **/** np**.**percentile(data['Number\_of\_Visitors'], low\_demand\_percentile))

high\_supply\_percentile **=** 75

low\_supply\_percentile **=** 25

data['supply\_multiplier'] **=** np**.**where(data['Number\_of\_Purchases'] **>** np**.**percentile(data['Number\_of\_Purchases'], low\_supply\_percentile),

np**.**percentile(data['Number\_of\_Purchases'], high\_supply\_percentile) **/** data['Number\_of\_Purchases'],

np**.**percentile(data['Number\_of\_Purchases'], low\_supply\_percentile) **/** data['Number\_of\_Purchases'])

demand\_threshold\_high **=** 1.2

demand\_threshold\_low **=** 0.8

supply\_threshold\_high **=** 0.8

supply\_threshold\_low **=** 1.2

data['adjusted\_price'] **=** data['Historical\_Price'] **\*** (

np**.**maximum(data['demand\_multiplier'], demand\_threshold\_low) **\***

np**.**maximum(data['supply\_multiplier'], supply\_threshold\_high)

)

data['profit\_percentage'] **=** ((data['adjusted\_price'] **-** data['Historical\_Price']) **/** data['Historical\_Price']) **\*** 100

profitable\_products **=** data[data['profit\_percentage'] **>** 0]

loss\_products **=** data[data['profit\_percentage'] **<** 0]

labels **=** ['Profitable Products', 'Loss Products']

values **=** [len(profitable\_products), len(loss\_products)]

fig **=** go**.**Figure(data**=**[go**.**Pie(labels**=**labels, values**=**values, hole**=**0.4)])

fig**.**update\_layout(title**=**'Profitability of Products (Dynamic Pricing vs. Historical Pricing)')

fig**.**show()

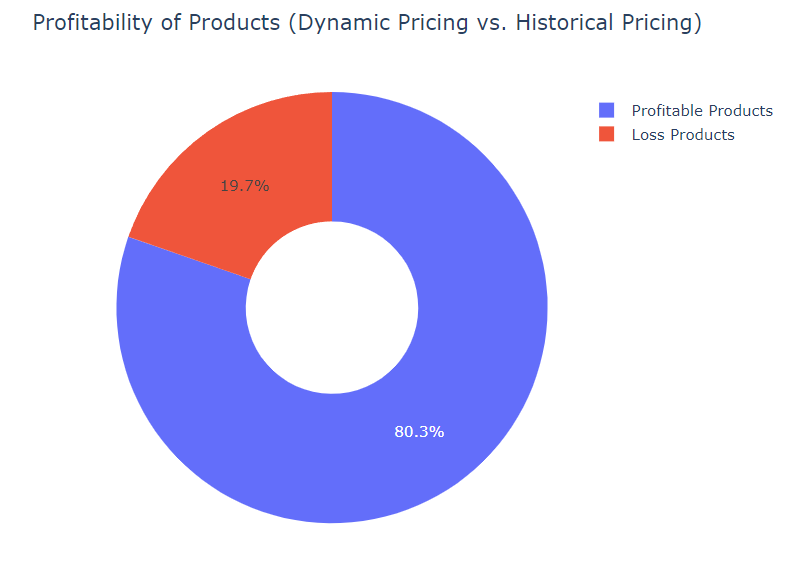


fig **=** px**.**scatter(data,

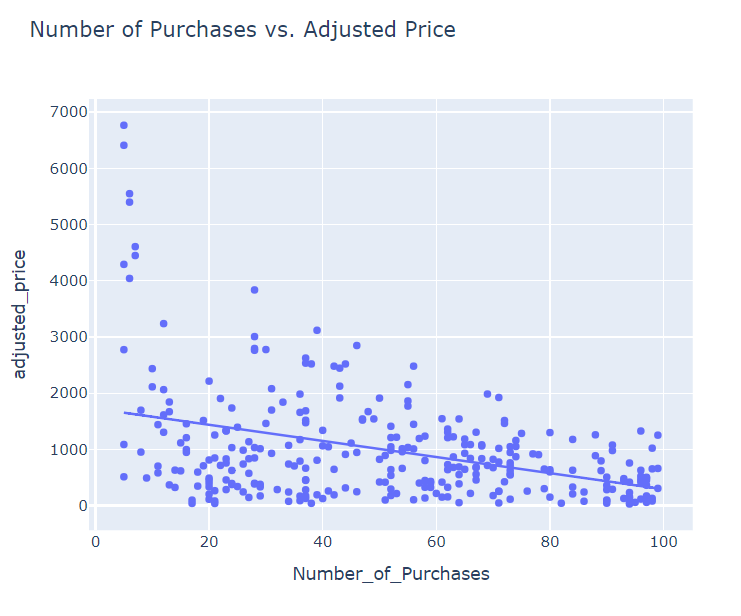
x**=**'Number\_of\_Purchases',

y**=**'adjusted\_price',

title**=**'Number of Purchases vs. Adjusted Price',

trendline**=**'ols')

fig**.**show()

****

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.ensemble **import** RandomForestRegressor

x **=** np**.**array(data[["Number\_of\_Visitors",

"Number\_of\_Purchases",

"Discount\_Offered",

"Historical\_Price"]])

y **=** np**.**array(data[["adjusted\_price"]])

x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(x, y, test\_size**=**0.2, random\_state**=**42)

y\_train **=** y\_train**.**ravel()

y\_test **=** y\_test**.**ravel()

model **=** RandomForestRegressor()

model**.**fit(x\_train, y\_train)

**def** predict\_price(number\_of\_visitors, number\_of\_purchases, discount\_offered, historical\_price):

input\_data **=** np**.**array([[number\_of\_visitors,

number\_of\_purchases,

discount\_offered,

historical\_price]])

predicted\_price **=** model**.**predict(input\_data)

**return** predicted\_price

user\_number\_of\_visitors **=** 100

user\_number\_of\_purchases **=** 20

user\_discount\_offered **=** 15

user\_historical\_price **=** 300

predicted\_price **=** predict\_price(user\_number\_of\_visitors,

user\_number\_of\_purchases,

user\_discount\_offered,

user\_historical\_price)

print("Predicted price:", predicted\_price)

Predicted price: [531.80519508]

**import** plotly.graph\_objects **as** go

y\_pred **=** model**.**predict(x\_test)

fig **=** go**.**Figure()

fig**.**add\_trace(go**.**Scatter(

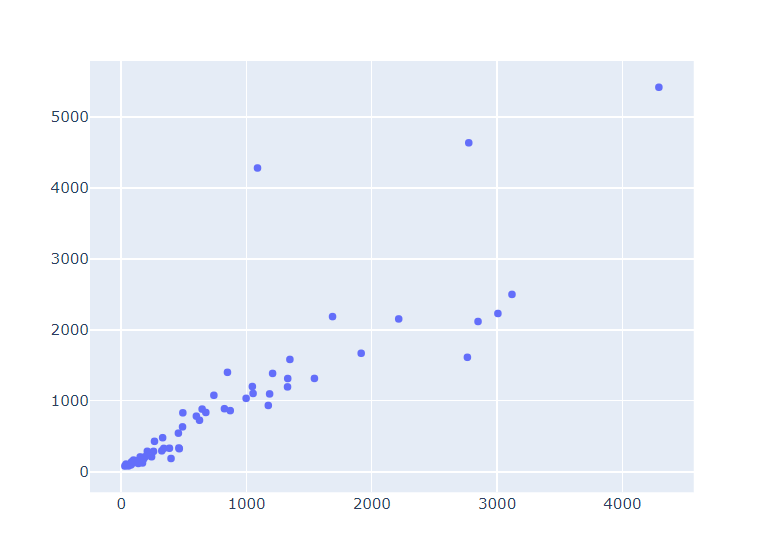
x**=**y\_test**.**flatten(),

y**=**y\_pred,

mode**=**'markers',

name**=**'Actual vs Predicted'

))



fig**.**add\_trace(go**.**Scatter(

x**=**[min(y\_test**.**flatten()), max(y\_test**.**flatten())],

y**=**[min(y\_test**.**flatten()), max(y\_test**.**flatten())],

mode**=**'lines',

name**=**'Ideal',

line**=**dict(color**=**'red', dash**=**'dash')

))

fig**.**update\_layout(

title**=**'Actual vs Predicted Values',

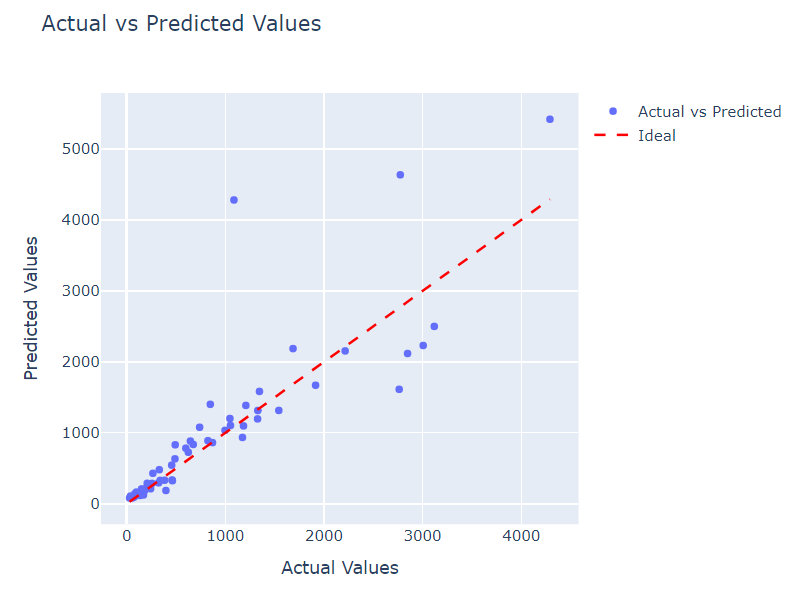
xaxis\_title**=**'Actual Values',

yaxis\_title**=**'Predicted Values',

showlegend**=True**,

)

fig**.**show()

****

**from** sklearn.metrics **import** mean\_absolute\_error, mean\_squared\_error, r2\_score

y\_pred **=** model**.**predict(x\_test)

mae **=** mean\_absolute\_error(y\_test, y\_pred)

mse **=** mean\_squared\_error(y\_test, y\_pred)

r2 **=** r2\_score(y\_test, y\_pred)

OUTPUT:

Mean Absolute Error (MAE): 259.9664692542472

Mean Squared Error (MSE): 320094.9289621058

R-squared Score (R2): 0.6328757510276888

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.preprocessing **import** StandardScaler, OneHotEncoder

**from** sklearn.compose **import** ColumnTransformer

**from** sklearn.pipeline **import** Pipeline

**import** pandas **as** pd

X **=** data**.**drop('Historical\_Price', axis**=**1)

y **=** data['Historical\_Price']

categorical\_features **=** ['Product\_Category', 'Customer\_Loyalty\_Status', 'Time\_of\_Purchase']

numerical\_features **=** [col **for** col **in** X**.**columns **if** col **not** **in** categorical\_features]

numerical\_transformer **=** StandardScaler()

categorical\_transformer **=** OneHotEncoder(drop**=**'first')

preprocessor **=** ColumnTransformer(

transformers**=**[

('num', numerical\_transformer, numerical\_features),

('cat', categorical\_transformer, categorical\_features)

])

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.2, random\_state**=**42)

X\_train\_preprocessed **=** preprocessor**.**fit\_transform(X\_train)

X\_test\_preprocessed **=** preprocessor**.**transform(X\_test)

**from** sklearn.model\_selection **import** GridSearchCV

**from** sklearn.ensemble **import** RandomForestRegressor

param\_grid **=** {

'n\_estimators': [50, 100, 200],

'max\_depth': [**None**, 10, 20, 30],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4]

}

rf\_model **=** RandomForestRegressor(random\_state**=**42)

grid\_search **=** GridSearchCV(estimator**=**rf\_model, param\_grid**=**param\_grid, cv**=**5, n\_jobs**=-**1, verbose**=**2)

grid\_search**.**fit(X\_train\_preprocessed, y\_train)

best\_params **=** grid\_search**.**best\_params\_

print("Best parameters:", best\_params)

Fitting 5 folds for each of 108 candidates, totalling 540 fits

Best parameters: {'max\_depth': 10, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 200}

best\_rf\_model **=** RandomForestRegressor(**\*\***best\_params, random\_state**=**42)

best\_rf\_model**.**fit(X\_train\_preprocessed, y\_train)

y\_pred **=** best\_rf\_model**.**predict(X\_test\_preprocessed)

**from** sklearn.metrics **import** mean\_absolute\_error, mean\_squared\_error, r2\_score

mae **=** mean\_absolute\_error(y\_test, y\_pred)

mse **=** mean\_squared\_error(y\_test, y\_pred)

r2 **=** r2\_score(y\_test, y\_pred)

print("Mean Absolute Error (MAE):", mae)

print("Mean Squared Error (MSE):", mse)

print("R-squared Score (R2):", r2)

**OUTPUT**

Mean Absolute Error (MAE): 51.126310015651114

Mean Squared Error (MSE): 5743.99861858873

R-squared Score (R2): 0.9383067825136141

**Overall Metrics**

* **Mean Absolute Error (MAE): 51.13**

Interpretation: The average absolute difference between predicted and actual values is 51.13. Lower MAE indicates better accuracy.

* **Mean Squared Error (MSE): 5743.99**

Interpretation: The average of the squared differences between predicted and actual values is 5743.99. This metric penalizes larger errors more heavily, so lower MSE is better.

* **R-squared Score (R²): 0.94**

Interpretation: The model explains 94% of the variance in the test data. A higher R² indicates a better fit.

**REQUIREMENTS**

**Google Account:**

Access Jupyter Notebook with a Google account or any local environment.

**Pre-installed Libraries:**

Ensure the following libraries are available:

* Pandas
* NumPy
* Scikit-learn
* Matplotlib
* Seaborn
* Plotly

**Additional Libraries:**

Install any missing libraries using ‘!pip install <library\_name>’.

**Optional GPU/TPU:**

For Jupyter Notebook running locally, you can configure GPU acceleration if supported by your hardware. For cloud-based notebooks, enable GPU or TPU acceleration if available.

**CONCLUSION**

In this project, we successfully developed a dynamic pricing model and conducted a comprehensive predictive analysis for ecommerce products. Initial data exploration and visualization highlighted significant trends and correlations, setting the stage for the implementation of a dynamic pricing strategy. This strategy, grounded in demand and supply multipliers, enabled us to distinguish between profitable and loss-making products effectively. The initial RandomForestRegressor model provided a foundation, but after hyperparameter tuning with GridSearchCV, the model's performance markedly improved, achieving an impressive R² score of 0.94, alongside a MAE of 51.13 and an MSE of 5744.00. This optimized model not only enhances pricing accuracy but also offers actionable insights for refining pricing strategies, ultimately driving better profitability and more informed decision-making for ecommerce businesses.

FUTURE SCOPE

The dynamic pricing strategy project has laid the foundation for implementing data-driven pricing models in e-commerce platforms, but there is significant potential for future enhancements. One area of expansion is the integration of real-time data streams, such as live customer behavior, competitor pricing, and seasonal trends, into the pricing model. By incorporating machine learning algorithms that continuously learn from these data streams, the system could adjust prices in real-time, offering a more personalized and competitive pricing strategy. Additionally, expanding the model to include other influencing factors like supply chain disruptions, marketing campaigns, and macroeconomic indicators could further refine price adjustments, improving profitability and customer satisfaction.

Another promising avenue is the development of predictive analytics to forecast long-term trends and customer lifetime value (CLV). By integrating CLV predictions with dynamic pricing, the model could prioritize high-value customers, offering them personalized discounts or loyalty rewards that enhance retention and lifetime profitability. Moreover, incorporating advanced techniques like reinforcement learning could enable the pricing model to learn optimal strategies through continuous interaction with the market, leading to more sophisticated and effective pricing decisions over time.

BIBLIOGRAPHY

[1] https://github.com/Mohshaikh23/Dynamic-Pricing-Strategy

[2] https://medium.com/@mehulved1503/dynamic-pricing-in-e-commerce

[3] https://medium.com/@baabak/dynamic-pricing-using-machine-learning-5e882282effe

[4] https://ieeexplore.ieee.org/document/9285592

Appendix A

|  |  |  |
| --- | --- | --- |
| **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY**  Department of Information Technology  **B.Tech 3rd year 2nd Sem IT-‘C’ GROUP PROJECT** | | |
| **Roll Number** | **Name** | **Title** |
| 21311A12E2 | **B. Adithya Reddy** | **Dynamic Pricing Optimization in E-Commerce Website** |
| 21311A12J0 | **K. Shashi Vardhan** |
| 21311A12K0 | **N. Shiva Sai** |

Abstract

This project focuses on developing a dynamic pricing model for ecommerce products, aimed at optimizing pricing strategies to enhance profitability. We began by analyzing a dataset containing key variables such as visitor numbers, purchases, product categories, and historical pricing. Through exploratory data analysis and visualization, we identified significant patterns and correlations. Leveraging these insights, we implemented a dynamic pricing strategy that adjusts prices based on demand and supply multipliers derived from percentile calculations.

To predict the adjusted prices, we trained a RandomForestRegressor model. Initial results were promising, but we further refined the model using hyperparameter tuning with GridSearchCV, which significantly improved its performance. The final model demonstrated high accuracy with an R² score of 0.94, alongside a Mean Absolute Error of 51.13 and a Mean Squared Error of 5744.00. This refined model not only provides accurate price predictions but also facilitates a more strategic approach to pricing, ultimately contributing to enhanced profitability and more effective decision-making in ecommerce.