A look at your Shopping Cart - Customer Behavior Modelling
Applied Statistics and Experimental Design

DANG MANH CUONG
20214949

TRAN MINH TUAN
20214978

NGUYEN HOANG ANH
20214946

NGUYEN TUAN LONG
20214963

TRAN HOANG ANH
20210023

Hanoi, July 2023

Contents

Abstract

Given that shopping is a daily habit for most of us, it is not surprising that sellers usually have access to a wealth of data from purchasers. Such data can support merchants with understanding their customers in order to improve their services. Given a synthetic set of data, we make use of various machine learning algorithms such as Linear Regression in order to attempt to predict a customer's shopping behavior. Despite some limitations, the project is a good example of how prospective sellers can take advantage of the customer data collected. The project also suggests some future directions to further harness the data.

Acknowledgement

We would like to express our gratitude to Assoc. Prof. Nguyen Linh Giang for giving us an opportunity to work together on a topic that we are interested in the Applied Statistics and Experimental Design course this semester. We would like to express the same gratitude to Assoc. Prof. Than Quang Khoat.The knowledge we gained from his Machine Learning course this semester has helped us in navigating through the problem and delivering the right approach to the problem.

Chapter 1

Introduction

Shopping is an essential activity in many people's lives. Shopping is an activity in which a customer browses the available goods or services presented by one or more retailers with the potential intent to purchase a suitable selection of them. As shopping is targeted at customers, it is important, from the standpoint of a retailer, that they have a good understanding of buyers' needs and demands. With the advent of online shopping, shopping data has become a valuable asset for merchants. The data can shed lights on a customer's shopping behavior, e.g. what they often buy, when they often make purchases,... Based on this wealth of information, it is easier for sellers to be sensitive to changes in market demand and deliver better values for their customers. The race to gain more customers for shops is now not just about conventional approaches, but also about whoever exploits their data best.

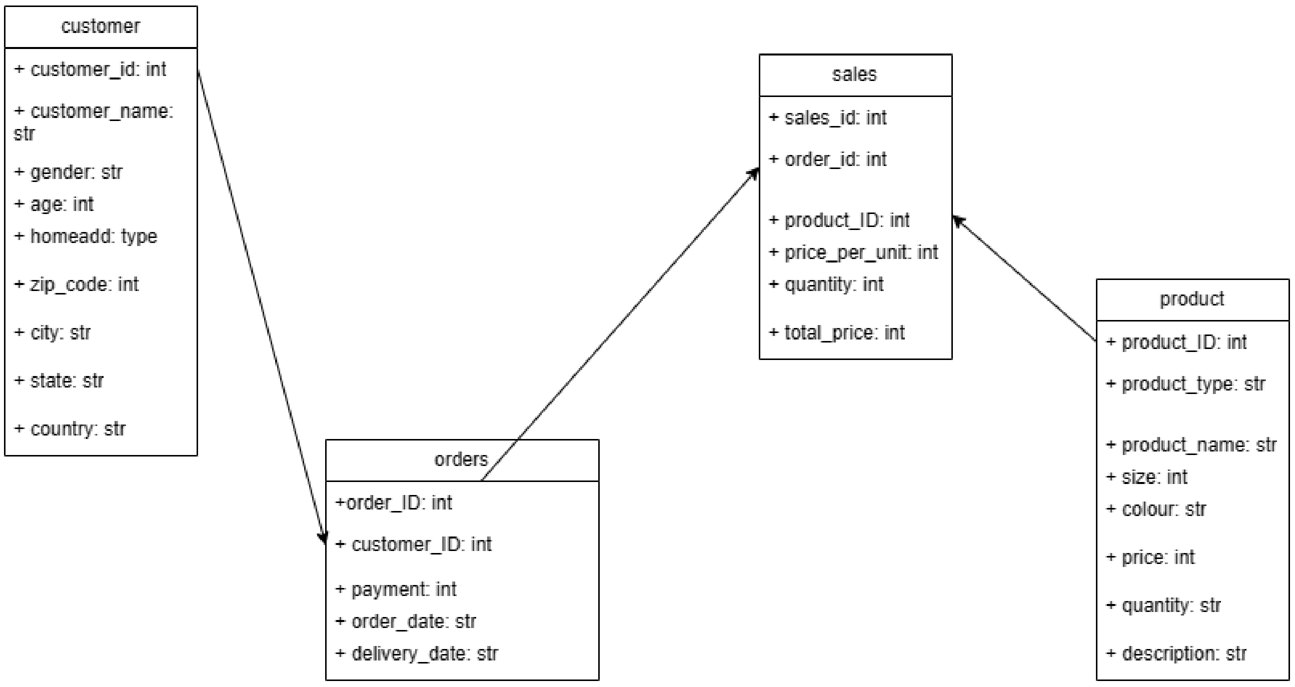
While the data this project explores is synthetic, it is a good representation of the actual information that merchants can have access to. Instead of resorting to usual, we will make use of machine learning (ML) - which is of great interest to every field in recent years. The performance of various ML models has improved immensely - which make it suitable for the purpose of customer behavior prediction.

Chapter 2

Methodology

2.1 Dataset description

The dataset contains synthetic data for e-commerce shopping, generated by one student for one of his courses at Carnegie Mellon University[1]. The dataset focuses on purchases of clothes. Included in the set are customer data, data on their orders at store, a list of products available and sales data. The dataset can be considered as a relational database; a UML class diagram representing the dataset structure can be found here:



For clarification, each customer with their own information creates some orders; each order belongs to one or many sales. However, a single sale can only be associated to one order. A sale involves selling products, with multiple information surrounding them.

Attributes

Below is the list of attributes represented in the diagram, explained:

customer_id	Uniquely identifies each customer
customer_name	Customer name
gender	Gender
age	Age
homeadd	Customer home address
zip_code	ZIP code
city	City
state	State
country	Country
order_ID	Uniquely identifies each order
payment	Total amount of payment for each order
order_date	Date of order
delivery_date	Date of delivery
sales_id	Uniquely identifies each sale
product_ID	Uniquely identifies each product
price_per_unit	Price per product unit
quantity	Total number of products in a sale
total_price	Total price of a sale
product_type	Product type
size	Size of product
colour	Colour
price	Price
quantity	Number of available units
description	Description

2.2 Data preprocessing

Shopping data comes in myriad shapes and forms; it is necessary to process them to a standardized form before any further operations can be done.

Firstly, we take a look at general information about the data from 4 relations:

```
customer.info()

RangeIndex: 1000 entries, 0 to 999
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   customer_id    1000 non-null   int64
 1   customer_name  1000 non-null   object
 2   gender         1000 non-null   object
 3   age            1000 non-null   int64
 4   home_address   1000 non-null   object
 5   zip_code       1000 non-null   int64
 6   city           1000 non-null   object
 7   state          1000 non-null   object
 8   country        1000 non-null   object
dtypes: int64(3), object(6)
memory usage: 70.4+ KB

order.info()

RangeIndex: 1000 entries, 0 to 999
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   order_id       1000 non-null   int64
 1   customer_id    1000 non-null   int64
 2   payment        1000 non-null   int64
 3   order_date     1000 non-null   object
 4   delivery_date  1000 non-null   object
dtypes: int64(3), object(2)
memory usage: 39.2+ KB

product.info()

RangeIndex: 1260 entries, 0 to 1259
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   product_ID    1260 non-null   int64
 1   product_type  1260 non-null   object
 2   product_name  1260 non-null   object
 3   size          1260 non-null   object
 4   colour        1260 non-null   object
 5   price         1260 non-null   int64
 6   quantity      1260 non-null   int64
 7   description   1260 non-null   object
dtypes: int64(3), object(5)
memory usage: 78.9+ KB

sales.info()
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   sales_id        5000 non-null   int64
 1   order_id        5000 non-null   int64
 2   product_id      5000 non-null   int64
 3   price_per_unit  5000 non-null   int64
 4   quantity        5000 non-null   int64
 5   total_price     5000 non-null   int64
dtypes: int64(6)
memory usage: 234.5 KB
```

Examining the information, we can see there are 1000 entries each for customer and order data; 1260 entries for product data and 5000 entries for sales data. No null values are detected.

To make it easier to work on the data, we will perform some join operations on the relations. Overall, customer and order data are joined on *customer_id* common attribute; which produces a joined relation (we will call it relation A for simplicity). After that, relation A is joined with the sales table based on the *order_id* attribute. This task results in a new relation (relation B). Finally, the product table is joined with B based on the common *product_id* attribute. We have our final relation which will be used for later analysis and modelling. Below is the information about the merged relation:

```
df.info()

Int64Index: 5000 entries, 0 to 4999
Data columns (total 25 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   product_id      5000 non-null   int64
 1   product_type    5000 non-null   object
 2   product_name    5000 non-null   object
 3   size            5000 non-null   object
 4   colour          5000 non-null   object
 5   price           5000 non-null   int64
 6   quantity_x      5000 non-null   int64
 7   description     5000 non-null   object
 8   customer_id     5000 non-null   int64
 9   customer_name   5000 non-null   object
 10  gender          5000 non-null   object
 11  age             5000 non-null   int64
 12  home_address    5000 non-null   object
 13  zip_code        5000 non-null   int64
 14  city            5000 non-null   object
 15  state           5000 non-null   object
 16  country         5000 non-null   object
 17  order_id        5000 non-null   int64
 18  payment         5000 non-null   int64
 19  order_date      5000 non-null   object
 20  delivery_date   5000 non-null   object
 21  sales_id        5000 non-null   int64
 22  price_per_unit  5000 non-null   int64
 23  quantity_y      5000 non-null   int64
 24  total_price     5000 non-null   int64
dtypes: int64(12), object(13)
memory usage: 1015.6+ KB
```

As shown above, we ended up with 5000 non-null entries. There are two attributes whose names are not quite right: *quantity_x* and *quantity_y*. We will perform a rename operation to change their names into *quantity_available_product* and *quantity_available_sales*, respectively:

```
    df=df.rename(columns = {'quantity_x':'quantity_available_product'})

    df=df.rename(columns = {'quantity_y':'quantity_product_sales'})
```

That concludes the initial part of data preprocessing.

Data Cleaning

Now, we will check for any data miss or duplication:

```
df.isnull().sum()
df.duplicated().sum()

product_id                    0
product_type                  0
product_name                  0
size                          0
colour                        0
price                         0
quantity_available_product    0
description                   0
customer_id                   0
customer_name                 0
gender                        0
age                           0
home_address                  0
zip_code                      0
city                          0
state                         0
country                       0
order_id                      0
payment                       0
order_date                    0
delivery_date                 0
```

```
sales_id                    0
price_per_unit              0
quantity_product_sales      0
total_price                 0
dtype: int64
```

No duplicated or missing values are detected - which is a good news as we do not have to do any data cleaning.

Deriving attributes

We add some attributes derived from the original dataset:
Minimum discount per product (in %):

$$[1 - (price_per_unit \div price)] * 100\%$$

Profit per product (in %)(*note*: this figure must not be smaller than 0)

$$[(price_per_unit \div price) - 1] * 100\%$$

Profit per product (in $)

$$price_per_unit - price$$

Total profit (in $):

$$profit_per_product(\$) * quantity_product_sales$$

2.3 Feature engineering & selection

Feature Engineering

While the previous steps have produced a significant number of features to be utilized during the training phase, it is clear that we need to explore other features that may be useful to improve the performance of the model. For starters, a feature is an individual measurable property or characteristic of a phenomenon.[2] Choosing informative, discriminating and independent features is a crucial element of effective algorithms in pattern recognition, classification and regression.

Season times

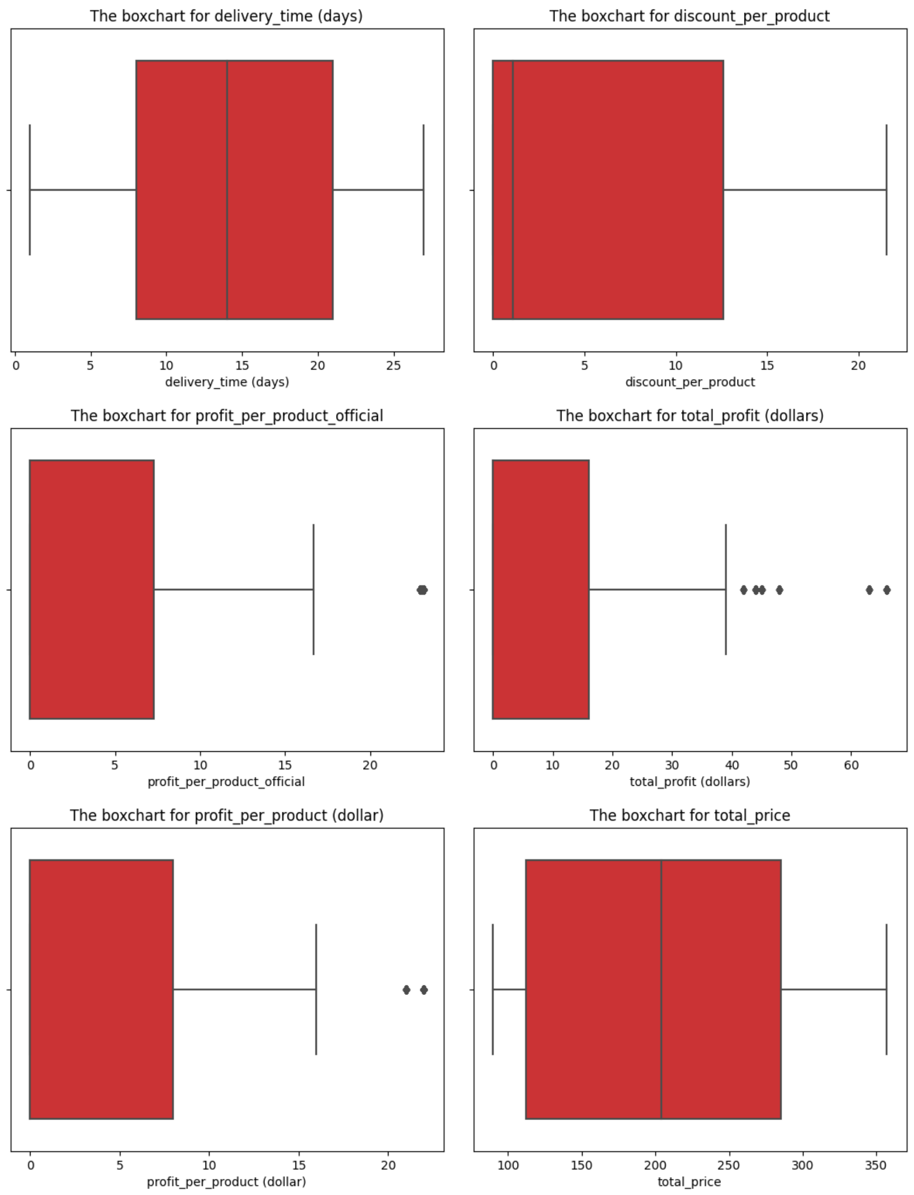
Seasonality may have a significant impact on shopping habits. As the geography of the data is solely based in Australia, it is reasonable to choose season times based on that in Australia, which are as follow [3]:

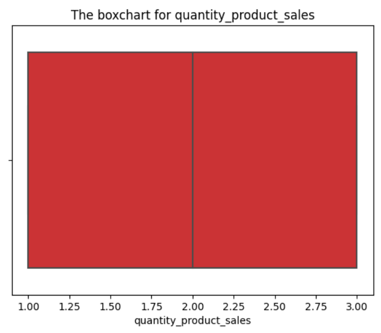
- Spring: September to November
- Summer: December to February
- Autumn: March to May
- Winter: June to August

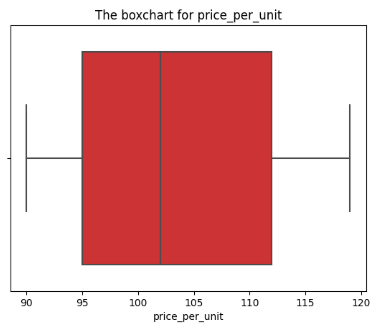
Outlier analysis

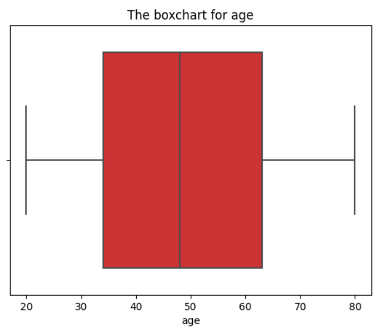
Outliers are those data points that are significantly different from the rest of the dataset. They are often abnormal observations that skew the data distribution, and arise due to inconsistent data entry, or erroneous observations. To ensure that the trained model generalizes well to the valid range of test inputs, it's important to detect and remove outliers. Therefore, outlier detection is an important step in tasks around data.

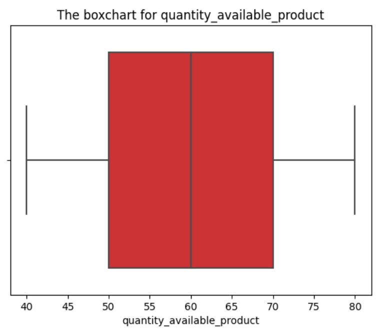
Here we plot boxcharts which visualize data belonging to various features of our interests, so that we can quickly spot any anomaly:

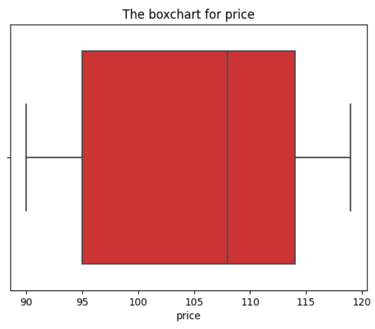










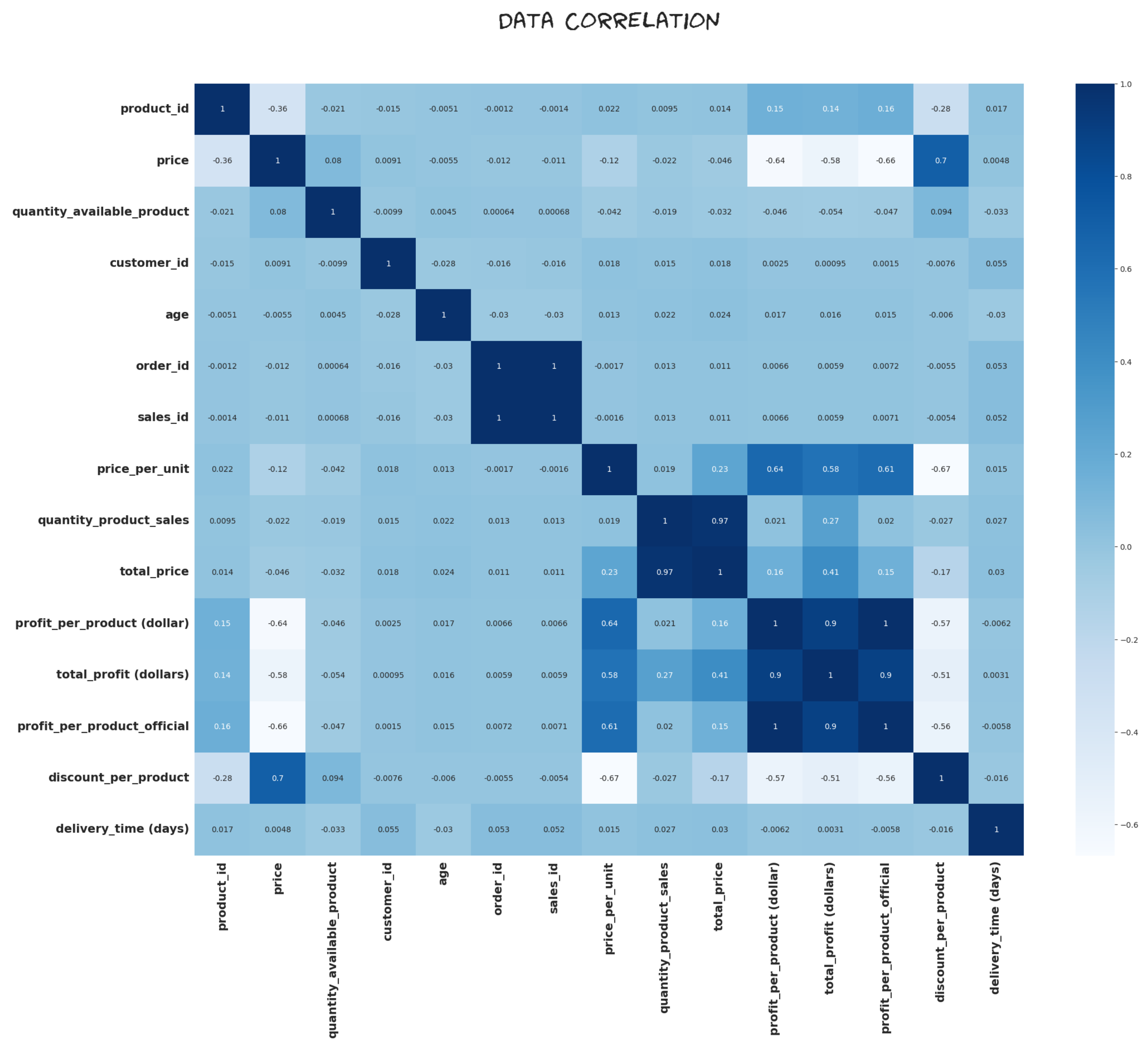


Boxplot visualisation shows outliers in the 'profit per product (dollar)', 'total profit (dollar)', and 'profit_per_product'* columns. We decide not to remove outliers data here because data detected as outliers are still reasonable and not invalid data, invalid means the data is corrupted, or incorrect. The outliers here occur in the data less frequently. Removing outliers here means drop all the non-zero entries, so column would be constant, less variable than it is in reality, and will lose a lot of valuable information, hence useless. Moreover, the infrequent data might happen again and may be very valuable in terms of the unique information they bring to the data. So there will be no cleaning of the ouliers data so that all the price profit values can be further analysed.

Feature Selection

Correlation is a measure of the linear relationship between 2 or more variables. Through correlation, we can predict one variable from the other. The logic behind using correlation for feature selection is that good variables correlate highly with the target. Furthermore, variables should be correlated with the target but uncorrelated among themselves.

If two variables are correlated, we can predict one from the other. Therefore, if two features are correlated, the model only needs one, as the second does not add additional information.

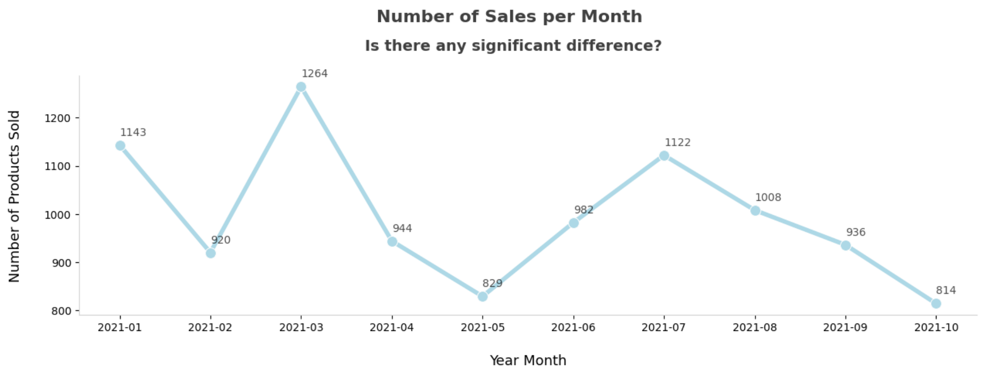


The above figure shows the Correlation coefficients between various features in the dataset. The values are, while not good, acceptable for usage in later stages. In conclusion, only these features will be retained for analysis: *'product_type', 'product_name', 'size', 'colour', 'description', 'gender', 'state', 'order_date', 'delivery_date', 'season'.*

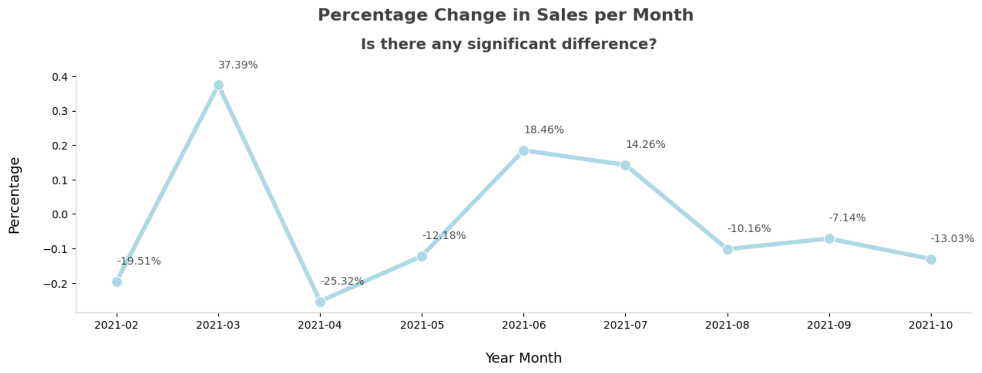
2.4 Exploratory data analysis

Exploratory data analysis (EDA) is a key step in any problems working with datasets. It allows us to detect any potential noise and errors, visualize the data to gauge various information about the dataset patterns. Here, we will illustrate the findings that we obtained from analyzing the dataset in hand.

Sales per Month

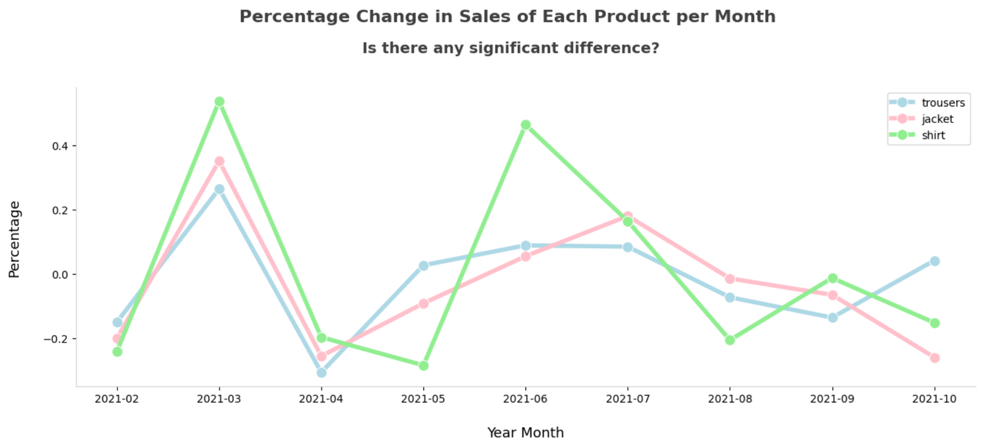


It is obvious that the sales are not consistent throughout the year in question. The numbers fluctuate heavily, with two halves of the year corresponding to two clear shopping cycles. The peaks are in March and July, while the drops concentrate on the end of each cycle.



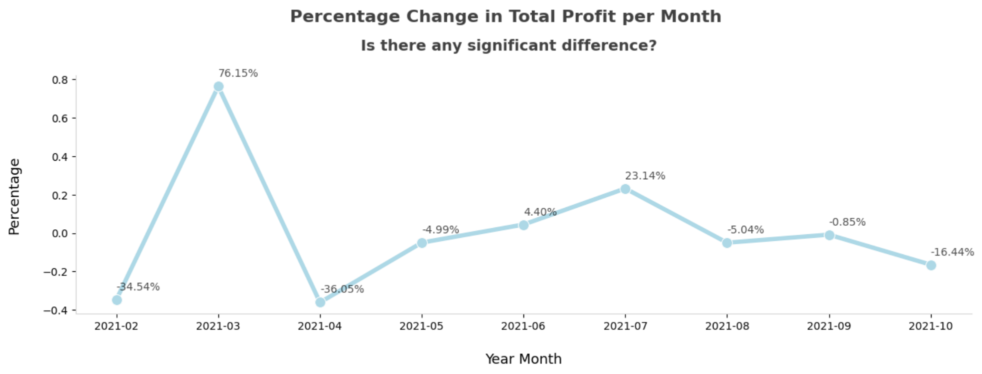
This figure also tells the same story: sales experienced a significant increase in March and June which is the beginning of the Autumn and Winter season, then decreased in the following month.

Now that we have completed the general analysis, let's take a closer look at each group of products:



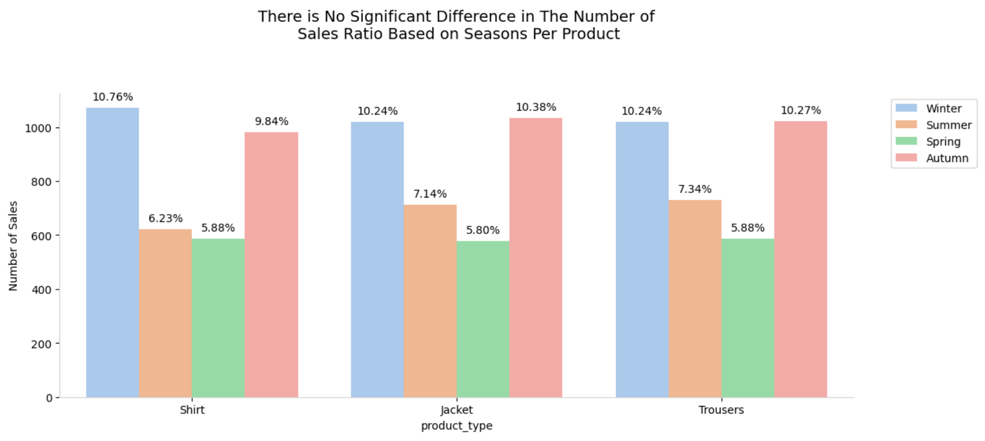
It is clear that the trends for each product roughly align with the general trends. Each product experienced a significant increase in sales in March, June, and July which is the beginning of the Autumn and Winter season, then decreased significantly in the following month. Overall, sales of all products in 2021 did not experience significant movements except at the beginning of the Autumn and Winter seasons.

Total Profit per Month



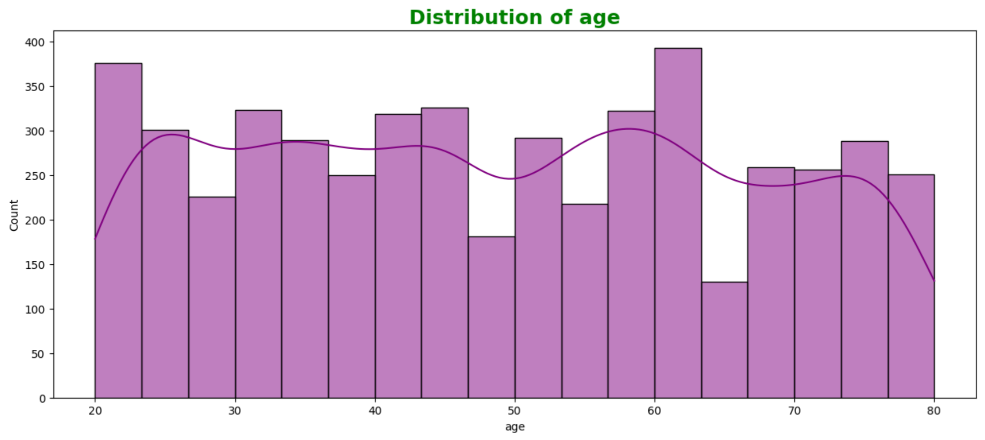
The graph of the total profit per month has the same pattern as the number of sales per month (3.2). The biggest profit was obtained in March and July which is the beginning of the Autumn and Winter season, then decreased in the following month.

Number of Sales per Product



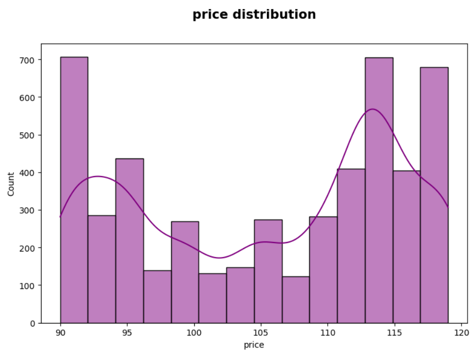
All products have a high number of sales in Winter and Autumn, which is in line with previous observations above. It is worth noting that the difference between seasons are not huge: about 5% for each of the product type.

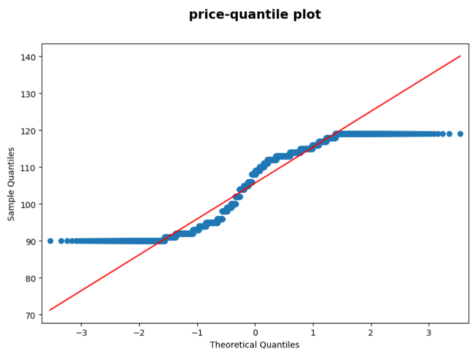
Visualization of age data



From this visualization, we can see that age is uniformly distributed, which the average centering on 50.

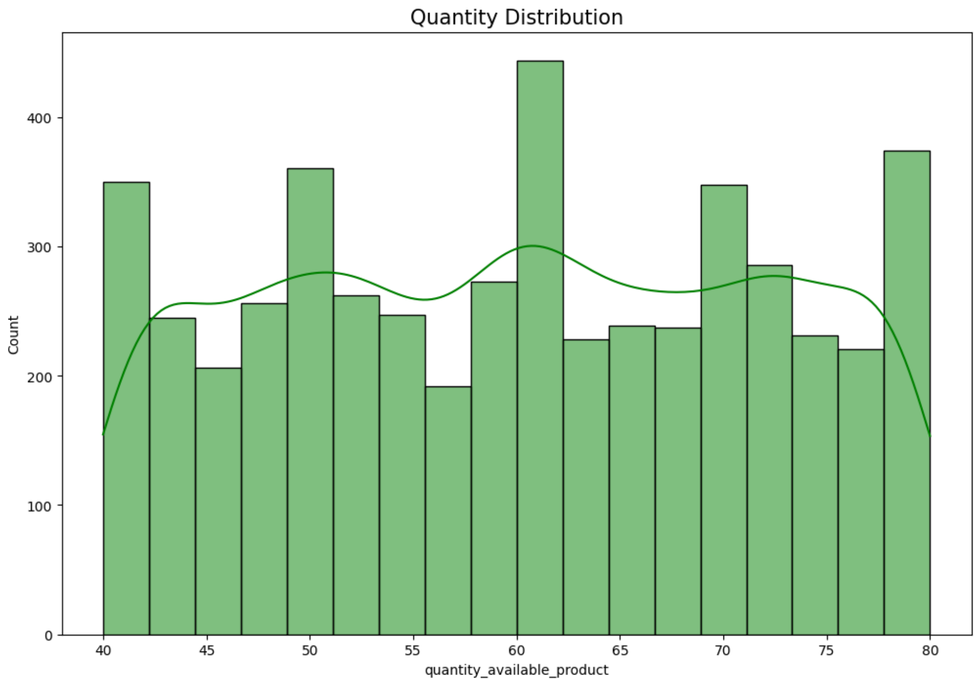
Price distribution





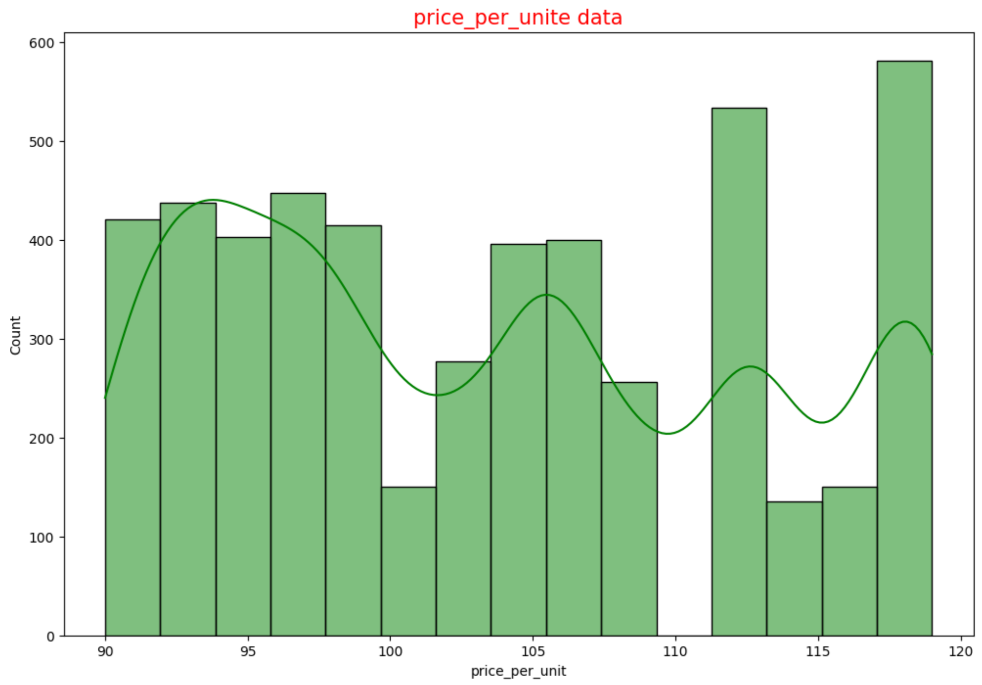
At least 75% of Shopping Cart population in the product price database in Australia has a price range from 89 795 to 126 395 (Australian Dollars)

Distribution of Quantity of products



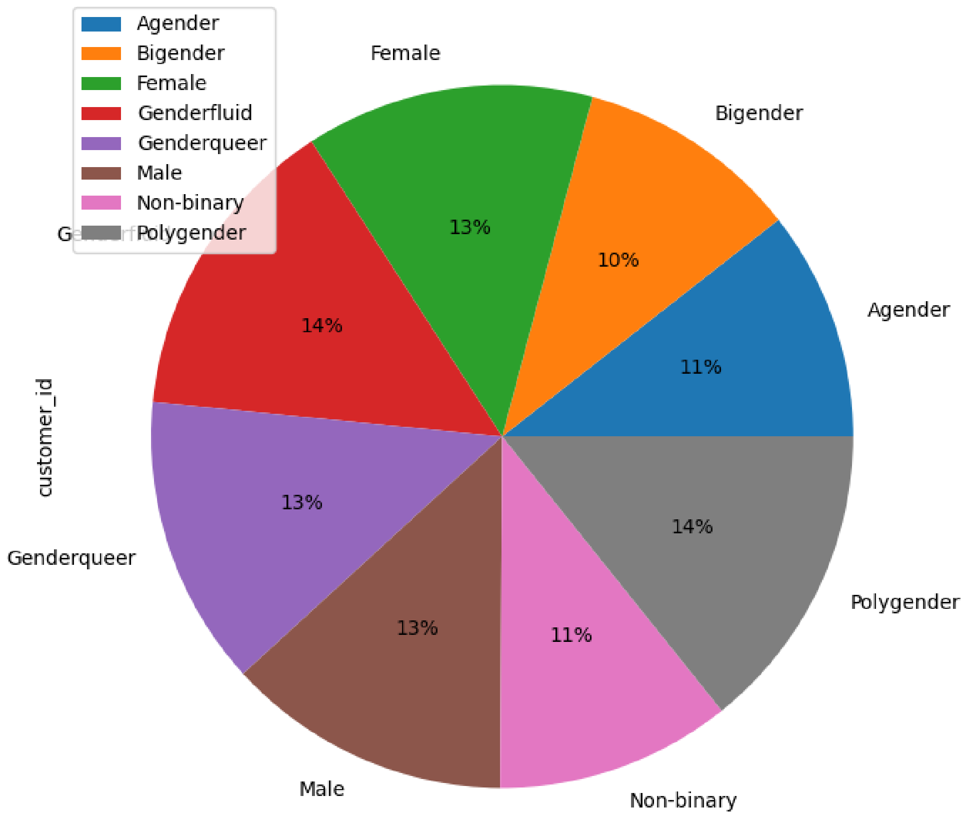
The quantity order of the Australian shopping cart in question ranges between 40-80.

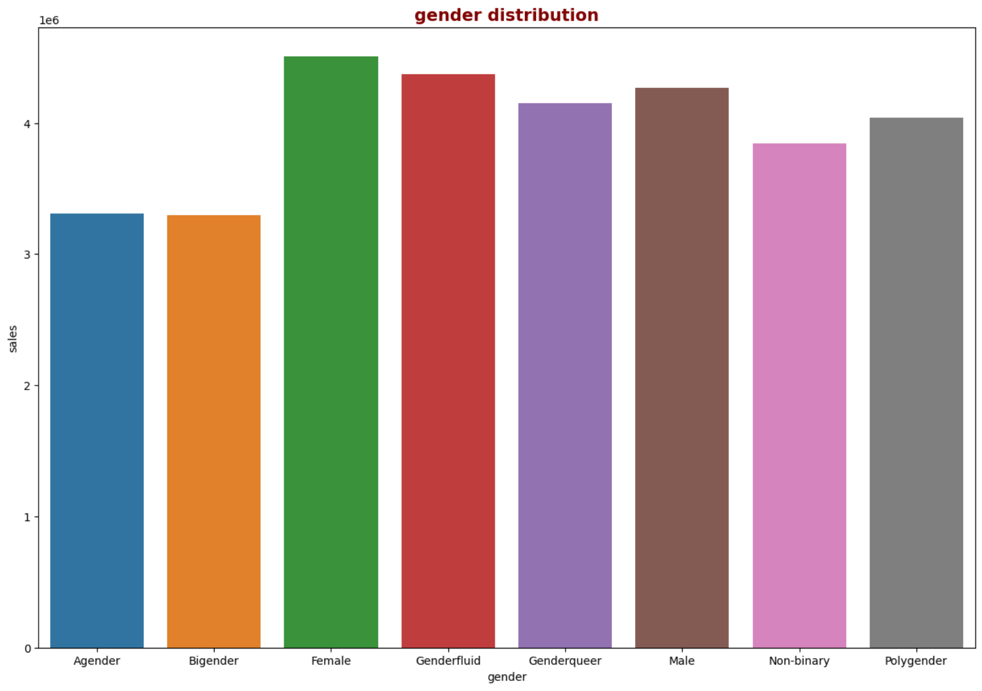
Price per Unit



For, at least 75% of the products in the Shopping Cart Database, the per unit price range is in between 90 to 120 (Australian Dollars).

Customer data distribution





Gender, state Female have contributed most in sales. Meanwhile people in the southern part of Australia spend more money than any other states.

2.5 Model selection and training

2.5.1 Linear Regression

Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.

This form of analysis estimates the coefficients of the linear equation, involving one or more independent variables that best predict the value of the dependent variable. Linear regression fits a straight line or surface that minimizes the discrepancies between predicted and actual output values. There are simple linear regression calculators that use a "least squares" method to discover the best-fit line for a set of paired data. You then estimate the value of X (dependent variable) from Y (independent variable).

Evaluation is based on three metrics: mean absolute error, root mean square error and R2 score.

The mean absolute error (MAE) is a measure of errors between paired observations expressing the same phenomenon. Examples of Y versus X include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement. MAE is calculated as the sum of absolute errors divided by the sample size:

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}.$$

The root mean square error (RMSE) measures the average difference between a statistical model's predicted values and the actual values. Mathematically, it is the standard deviation of the residuals. Residuals represent the distance between the regression line and the data points.

$$RSME = \sqrt{\frac{\sum(\mathrm{y}_i - \hat{\mathrm{y}}_i)^2}{N - P}}$$

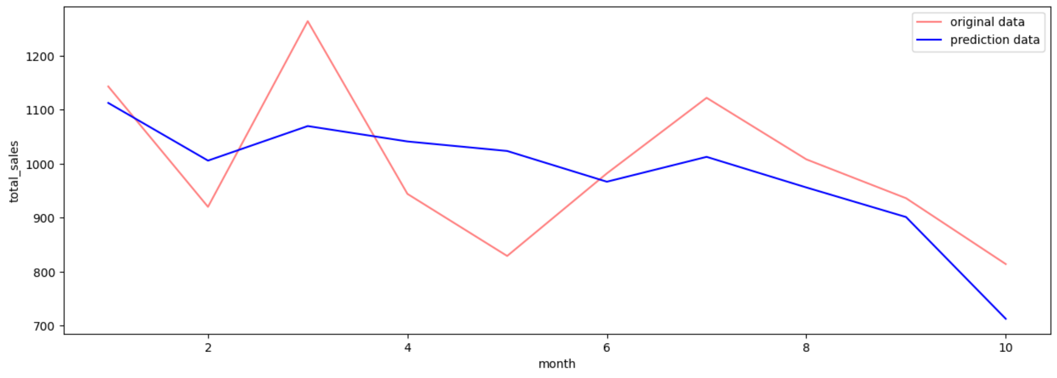
R-Squared (R^2) is a statistical measure used to determine the proportion of variance in a dependent variable that can be predicted or explained by an independent variable. In other words, R-Squared shows how well a regression model (independent variable) predicts the outcome of observed data (dependent variable).

Here, we attempt to predict the number of sales given a certain timeframe in the year. First, we will do the experiment on the full range of products, then test on every product category.

For training and testing, 291 data points are randomly splitted into two sets of testing and training data by the ratio of 20 80.

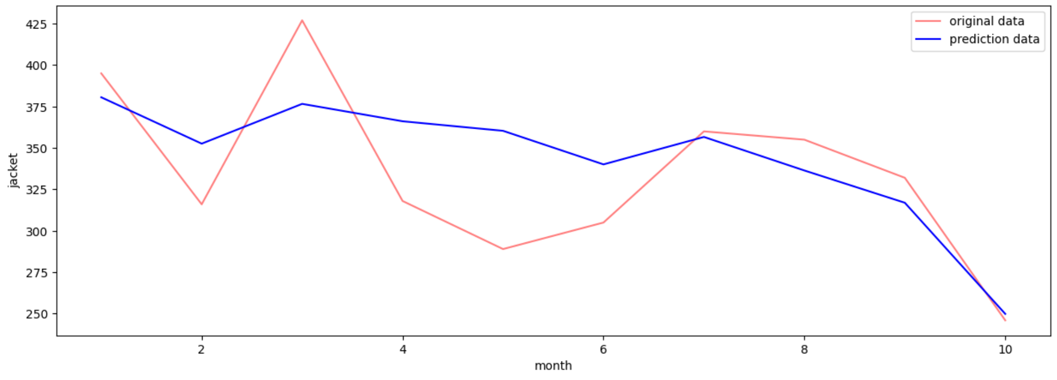
Results

Sales Prediction



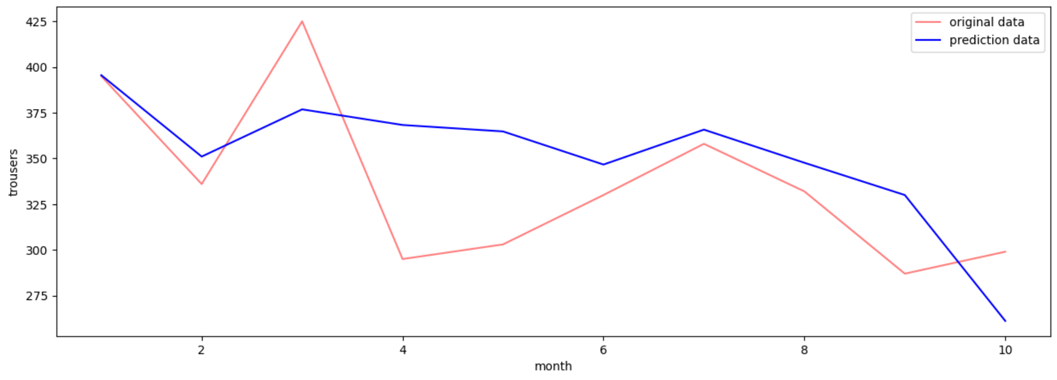
```
MAE: 18.38
RMSE: 22.18
R2 score: -0.10
```

Jacket Sales Prediction



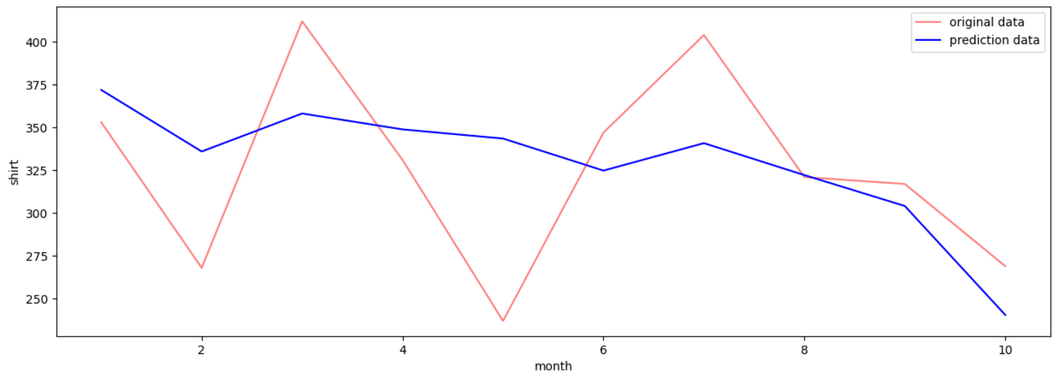
```
MAE: 7.14
RMSE: 9.22
R2 score: -0.07
```

Trousers Sales Prediction



```
MAE: 5.97
RMSE: 7.47
R2 score: -0.08
```

Shirt Sales Prediction



```
MAE: 6.09
RMSE: 8.20
R2 score: -0.13
```

2.5.2 ARMA

Autoregressive–moving-average (ARMA) models provide a parsimonious description of a (weakly) stationary stochastic process in terms of two polynomials, one for the autoregression (AR) and the second for the moving average (MA).

$$y_t = c + \sum_{n=1}^{p} \alpha_n y_{t-n} + \epsilon_t$$

Figure 2.1: AR Formula

To perform evaluation, the root mean square error (RMSE) is used, with the same formula as that in the linear regression model above.

In this case, the set is splitted into two parts: the sales data before June is the training set, and the rest is the testing set.

Dickey-Fuller Test for stationary series

A stationary series is one whose statistical properties such as mean, variance, covariance, and standard deviation do not vary with time, or these stats properties are not a function of time. Stationary series is easier for statistical models to predict effectively and precisely.

Dickey-Fuller test uses an autoregressive model and optimizes an information criterion across multiple different lag values. A Dickey-Fuller test is a unit root test that tests the null hypothesis that alpha =1 in the following model equation. alpha is the coefficient of the first lag on Y.

Null Hypothesis (H0): alpha=1

where,

y(t-1) = lag 1 of time series delta Y(t-1) = first difference of the series at the time (t-1) Fundamentally, it has a similar null hypothesis as the unit root test. That is, the coefficient of Y(t-1) is 1, implying the presence of a unit root. If not rejected, the series is taken to be non-stationary.

The Augmented Dickey-Fuller test evolved based on the above equation and is one of the most common forms of the Unit Root Test.

As the name suggests, the ADF test is an 'augmented' version of the Dickey-Fuller test. The ADF test expands the Dickey-Fuller test equation to include a high-order regressive process in the model.

$$y_t = c + \beta t + \alpha y_{t-1} + \phi_1 \Delta Y_{t-1} + \phi_2 \Delta Y_{t-2} .. + \phi_p \Delta Y_{t-p} + e_t$$

Figure 2.2: ADF Formula

A key point to remember here is: Since the null hypothesis assumes the presence of unit root, that is alpha=1, the p-value obtained should be less than the significance level (say 0.05) in order to reject the null hypothesis. Thereby inferring that the series is stationary.

```
1. ADF :  -29.25278641636878
2. P-Value :  0.0
3. Num Of Lags :  3
4. Num Of Observations Used For ADF Regression and Critical Values Calculation : 6
5. Critical Values :
```

```
1% :  -5.354256481481482
5% :  -3.6462381481481483
10% :  -2.901197777777778
```

For this dataset, the p-value obtained is smaller than the significance level of 0.05, and the ADF statistic is higher than any of the critical values. So, the time series is, in fact, stationary.

Here's the RMSE for the model applied for the dataset:

```
RMSE:  190.31318646889005
```

It is clear that the error is higher than that in linear regression model.

2.5.3 ARIMA

$$Y_t = \beta_2 + \omega_1\varepsilon_{t-1} + \omega_2\varepsilon_{t-2} + \ldots + \omega_q\varepsilon_{t-q} + \varepsilon_t$$

Figure 2.3: ARIMA Formula

An autoregressive integrated moving average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model. ARIMA models are applied in some cases where data show evidence of non-stationarity in the sense of mean (but not variance/autocovariance), where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity of the mean function (i.e., the trend). When the seasonality shows in a time series, the seasonal-differencing could be applied to eliminate the seasonal component.

Here's the RMSE for the model applied for the dataset:

```
RMSE:  211.20816449749827
```

The result improves with AutoARIMA used. The autoARIMA process automatically discover the optimal order for an ARIMA model. It seeks to identify the most optimal parameters for an ARIMA model, settling on a single fitted ARIMA model.

Here's the RMSE for the model applied for the dataset:

```
RMSE:  189.15403589685783
```

2.5.4 SARIMA

$$y_t = c + \sum_{n=1}^{p} \alpha_n y_{t-n} + \sum_{n=1}^{q} \theta_n \epsilon_{t-n} + \sum_{n=1}^{P} \phi_n y_{t-sn} + \sum_{n=1}^{Q} \eta_n \epsilon_{t-sn} + \epsilon_t$$

Figure 2.4: SARIMA Formula

Seasonal Autoregressive Integrated Moving Average, SARIMA or Seasonal ARIMA, is an extension of ARIMA that explicitly supports univariate time series data with a seasonal component.

It adds three new hyperparameters to specify the autoregression (AR), differencing (I) and moving average (MA) for the seasonal component of the series, as well as an additional parameter for the period of the seasonality.

Here's the RMSE for the model applied for the dataset:

```
RMSE:  185.78912777662742
```

With best fit AIC -pdq/s by grid search, surprisingly, the error increases:

```
RMSE:  325.68483012381733
```

Chapter 3

Conclusion

Customers are starving for a better, tailored shopping experience. There is no better way to meet that need than directly taking advantage of their purchasing decisions. It is clear that exploiting sales data is the way forward for sellers to adapt to the ever-changing market demands. This project has provided us with a good opportunity to thoroughly analyze a set of shopping cart data to look for any shopping patterns that may support merchants with their business decisions. Some findings are quite surprising. The analysis shows that there is no sufficient evidence to conclude that there is a significant correlation between changing seasons and total sales. The model evaluation focuses on ARMA and linear regression, with linear regression, a simple model, showing better performance than ARMA.

Future research will focus on feature engineering to seek for features that more accurately reflect customer patterns. In addition, we would like to do the research on different datasets to validate the results.

Appendix A

Reference

[1] Shopping Cart Database, Ruchi Bhatia, Kaggle, 2021:
https://www.kaggle.com/datasets/ruchi798/shopping-cart-database

[2] Bishop, Christopher (2006). *Pattern recognition and machine learning.* Berlin: Springer. ISBN 0-387-31073-8.

[3] Australia seasons, Tourism Australia, 2023:
https://www.australia.com/en/facts-and-planning/when-to-go/australias-seasons.html

[4] scikit-learn: machine learning in Python: *https://scikit-learn.org*