Preparation and analysis of JD.com data, a detailed report, and notebook files using Python packages

Modeling and analysis - making predictions and finding useful insights

Executive Summary

JD.com, China's top online retailer, commands a commanding position in the market, with a staggering net revenue of US$67.2 billion in 2018 and a large customer base of over 320 million yearly active users. JD.com, which is well known for its unwavering commitment to quality and authenticity across a wide range of product categories such as fresh foods, apparel, electronics, cosmetics, and more, distinguishes itself through a unique business model. JD.com protects product excellence by combining a first-party strategy that oversees the whole supply chain with a curated marketplace that prioritizes quality over quantity. Vendors make use of a national fulfillment network that covers 99% of China's population.

The examination focuses on building models that accurately predict delivery times for customer orders. Also, to identify potential changes in order volumes, it investigates consumer behavior by examining average item purchases across various order types (1P or 3P). To check the equitability of users across the city tiers, i calculated the spatial distribution of purchases across the city. The order processing delays are investigated by comparing the average time for packages to leave the warehouse for 1P and 3P orders. Also, calculated the total number of packages between the origin distribution center and the destination center. The education level and age level of the majority of users. The price range of most of the orders. Do the typical order volumes for these two kinds (1P and 3P) of orders differ significantly. Finally the sales by day, hour of the day and quantity is measured.

Project Motivation and Background

This data-driven approach is driven by the goal of improving customer happiness and sales operations in an e-commerce environment. By utilizing extensive datasets that include consumer

information, sales transactions, and delivery logistics, the objective is to improve customer satisfaction and optimize the shopping process. The background is that enormous volumes of data must be gathered and analyzed to make wise judgments at every point of the sales funnel.

1. How many items, on average, did customers buy across all order types (1P or 3P) (quantity field)? Do the typical order volumes for these two kinds of orders differ significantly? (orders table)
2. How were customer purchases distributed throughout cities at different levels (city_level)? Are there more users from higher city levels, or are there people from all city levels represented fairly? (users)
3. What is the average duration (ship_out_time) for packages to leave the warehouse once an order is placed? Is there a big difference in processing times between 1P and 3P orders? (delivery table)
4. Predicting the delivery time for customer orders.

The data does have some difficulties with quality. The dataset has a significant number of missing values. The User_Id has only a few missing entries, whereas the dc_id has the most missing data. Because of some inconsistent values in Sku_Id, they must be alphanumeric but include some records with only numeric values. A limited number of records having numerical data can be erased to make the data consistent. Because it isn't utilized to anticipate the results, it's possible to disregard the variable dc_id, which contains most of the missing information. The data collection process begins by gathering sales transaction records capturing details like SKU, date, and quantity sold. Once obtained, the data is filtered to isolate transactions by specific days and SKUs. Concurrently, customer data, including PLUS membership status, might reside in a database, aiding in buyer identification. Delivery priority rules are established, potentially granting preference to PLUS members based on their membership level, order size, or location. Orders are evaluated for adherence to regulations, giving precedence to PLUS members in the fulfillment process. Ultimately, order prioritization occurs based on various factors, such as membership status and order status, ensuring efficient and rule-compliant order processing.

Data Description

Each SKU in the database can be classified as "first-party owned" (1P) or "third-party owned" (3P), depending on who owns the inventory for that SKU.3. JD.com manages all 1P SKUs, including product assortments, inventory replenishments, product pricing, order deliveries, and after-sales customer service. Despite their varied operations, 1P and 3P SKUs compete for sales on the JD.com platform through different pricing tactics and marketing activities.

In general, 1P SKUs are top sellers in the category. JD.com can fully control the entire customer experience by owning these 1P products, ensuring guaranteed quality, fast delivery, and good customer service. Third-party merchants handle all 3P SKUs on the JD marketplace. In particular, the associated merchant can choose between JD Logistics' logistics services and those of other logistics service providers to complete an order for a 3P SKU.4 The JD.com data sets thoroughly understand the operations related to every SKU in a single anonymized consumable category in March of 2018. The tables used in data analysis from JD.com are Order, User, Delivery, Inventor, and Network

User table and description

Variables

user_ID	String values in this field serve as each user's unique identifier. For example, "000000f736" designates a particular user in the dataset.
user_level	Integer field that represents the user's tier or level on the platform. '10', for instance, can indicate a particular degree of user interaction or access.
Plus	A numeric field that indicates whether the user is a member of PLUS; values other than '0' may indicate that the user is not a member.
Gender	A string field with the user's estimated gender indicated by letters, like 'F' for female.
Age	A text field with the user's estimated age range is entered. The age range, for example, could be represented by "26–35".
matri_status	A text field that provides an estimate of the user's marital status. M, for instance, may stand for "married."
education	An integer field that may employ numerical codes to represent varying degrees of education to estimate the user's education level.

purchase_power	An integer field that uses probabilities to approximate the user's spending power or purchasing power; various levels of purchasing power are probably represented by numerical values.
city_level	An integer field that encodes the user's address's city level; it may employ numeric values to classify various city tiers.

Observations: A distinct user on JD.com is associated with each observation in this dataset. The observations encompass a range of attributes that describe the demographics, buying patterns, membership status, and other approximations about the characteristics of the users. These insights offer valuable information about the diversity and behavior of the platform's user base.

Table orders and description and Variables

order_ID	This field has string values that function as distinct order IDs. For example, "3b76bfcd3b" denotes a particular order in the dataset.
user_ID	This field has string values that function as special order-placing user identifiers. In the dataset, "3cde601074", for instance, designates a particular user.
sku_ID	String values that serve as special IDs for SKUs (Stock Keeping Units)—specific products inside an order—are contained in this field. For example, a sample SKU_ID linked to an order is "443fd601f0".
order_date	A string field that holds the order placement date.
order_time	A string field that contains the exact moment the order was placed.
Quantity	An integer field that indicates how many units of the SKU were ordered in a specific transaction. '1', for instance, denotes the ordering of a single item.
Type	An integer field designating a 1P (first-party) or 3P (third-party) order, respectively. One could imagine that '1' would stand for a 1P order.
assurance	An integer field that indicates how many days the order should take to arrive. A delivery within two days, for example, would be indicated by the number '2'.
original unit cost	A float field with the original list price of the SKU's single unit. '99.9', for instance, might

Observations: Every observation in this dataset corresponds to a distinct order placed by a JD.com user. The observations offer a wealth of information about the order, including the user, the SKUs purchased, the quantity, pricing data, any applied discounts, the order timing, and the anticipated delivery date. This information provides a thorough understanding of the purchasing behavior of users as well as transactional details.

Delivery table and description

Variables

package_ID	This field has string values that function as distinct package identifiers. This ID matches the order_ID if every SKU in the order is included in the package. For instance, the package "209a005c40" in the dataset denotes a particular package.
order_ID	This field has string values that function as distinct order IDs. It symbolizes the package's associated order and is connected to it. For example, if the package includes every SKU in the order, "209a005c40" might also be used as the order ID.
Type	An integer field informing us of the package's affiliation with a first party (1P) or third-party (3P) order. One could imagine that '1' would stand for a 1P order.

Observations: Every observation in this dataset relates to the current state of a particular package's delivery that is connected to an order. These observations include details regarding the package's journey, such as when it left the warehouse, when it reached a station, and when it was finally delivered to the client. By offering insights into the delivery schedule and logistics, this data makes it possible to analyze metrics related to order fulfillment and shipment effectiveness.

Table inventory and description

Variables

dc_ID	The integer parameter represents the Distribution Center ID. for example, '9' can represent a particular distribution center.
sku_ID	String values that serve as distinctive IDs for SKUs (Stock Keeping Units), or individual products inside the inventory, are contained in this field. As an illustration, the sample SKU_ID for the inventory is "fcc883f713".
Date	A string field that holds the date the inventory status is updated. I used the format 'yyyy-mm-dd'.

Observations: Every observation in this dataset relates to a certain SKU's inventory state at a given distribution facility on a given date. These observations provide details regarding the stock levels of various SKUs at various distribution centers on various dates. By monitoring SKU availability and distribution center inventory over time, this data offers insights into inventory management, and stock levels, and may even help with logistics planning and supply chain optimization

Table Network and description

Variables

region_ID	A field with an integer value that holds the Region ID. One possible use of '2' is to designate a particular area in the network dataset.
dc_ID	An integer field denoting the District ID, which, in this case, is the same as the warehouse ID. In the network dataset, '6', for instance, can represent a particular neighborhood or warehouse.

Observations: All the observations in this dataset concern the connections between the network's districts and warehouses and its regions. These observations disclose the mapping or association between various regions and the matching districts or warehouses. The geographic distribution of warehouses across different regions may be shown by this data, which could help with logistics planning and the comprehension of the network structure inside the business's operations.

Data Preparation

After reading the datasets using read.csv. one should understand the data clearly and search for missing values, timestamps, shape, and size of the data. I used info() to get the size, shape, and type of data. shape() to get the number of rows and columns. Changed a datatype using astype()

- ✓ Deciding what variable I might need in our analysis: There are two ways to remove unwanted variables- drop () and del(). drop() is used to remove specific columns from our data frame. I needed to use inplace =True. If True, the original DataFrame is modified; if False (default), a new DataFrame with dropped elements is returned. If del() is used, it deletes elements from the data frame. It is a statement rather than a method. So, it changes the existing data frame.
- ✓ Handling missing values: Checked for any missing or null values in our data frame. I used isnull() to check for null values. Next, I decide whether to drop or assign the null values. one can assign mean or median for numeric values and for categorical, and can replace with mode using fillna().Fig:1 shows the missing value in the promise variable of order table.

order_ID	0
user_ID	0
sku_ID	0
order_date	0
order_time	0
quantity	0
type	0
promise	208583
original_unit_price	0
final_unit_price	0
direct_discount_per_unit	0
quantity_discount_per_unit	0
bundle_discount_per_unit	0
coupon_discount_per_unit	0
gift_item	0
dc_ori	0
dc_des	0
dtype: int64	

```
variables with missing values order
promise    208583
dtype: int64
```

Fig :1 shows the missing values in order table

- ✓ Transform compound variables: working with strings when dealing with strings, transforming compound variables often requires either dividing a single column that contains compounded information into many columns or performing transformations to more accurately reflect the data that is being transformed. I used extract() to separate two compounded columns. To add the extracted column back to the data frame, used expand= True.
- ✓ I created dummy variables to convert categorical variables into dummy variables. The keyword to create dummy variables is str.get_dummies(). They are binary columns. I integrated the dummy variables back into the original data by the index since there is no common column to join them. Next, I needed to remove leading and trailing whitespaces, if any, use str. strip() to remove the white spaces.
- ✓ Working with date and time: Converted the integer timestamps into the pandas timestamp object for easy manipulation of data and analysis of temporal data. To access attributes, one can use dt or can convert the column into a datetimeindex object. For time-series data, one should convert it into a datetimeindex object for filtering and analysis. The datetimeindex is more useful for data that is recorded at regular time intervals.

Exploratory Data Analysis

An inquisitive and visual method of examining datasets is called exploratory data analysis (EDA). The main goals are comprehending the data's structure, summarizing its salient features,

and spot any patterns or trends. Techniques used in EDA include data visualization, statistical summaries, and the investigation of correlations between variables. Prior to conducting further in-depth analyses or modeling, it is an essential stage in the data analysis process that aids in the generation of hypotheses, insights, and well-informed judgments. Because EDA is iterative, as research advances, ideas and insights may be refined. All in all, it offers a basis for significant analysis and more research into the data.

During the initial phase of Exploratory Data Analysis, the CSV files were imported using Numpy and Pandas. Use the code order to retrieve the column names of a Data Frame order and other data frames in Pandas. Columns. Therefore, the Data Frame's Index object, which contains the column names, can be easily accessed and used for iteration and other actions requiring column names. Understanding the Data Frame's structure and enabling additional data processing or analysis are made easier with the help of this information. Subsequently, the percentage of the missing value was calculated, providing a quantitative understanding of incompleteness within datasets. After finding the missing value, merged the order table and delivery table to predict the delivery time for orders as shown in Fig:2

```
merged_data = pd.merge(order, delivery, on='order_ID', how='inner')
print("\nmerged_data")
print(merged_data)
```

Fig: 2 Merging the required tables

The merged table is used to calculate total number of packages between the origin distribution center and the destination center using Fig:3

```
package_counts = merged_data_filtered_2.groupby(['dc_ori', 'dc_des'])['package_ID'].nunique().reset_index()
print(package_counts)
```

```
     dc_ori  dc_des  package_ID
0         1       1         409
1         1      18           1
2         1      31           8
3         1      39          24
4         1      46           1
..      ...     ...         ...
621      65      60           4
622      65      61           2
623      65      64          15
624      66      20         125
625      67      67         405

[626 rows x 3 columns]
```

Fig:3. Number of packages between distribution and destination center

Further statistical calculations are also included in the study, including the determination of the mean, median, minimum, and maximum values for the 'original_unit_price' column. These metrics offer a more thorough comprehension of the values' distribution, range, and central tendency inside the designated column, as shown in Fig: 4.

```
statistics = merged_data_filtered_2['original_unit_price'].describe()

# Extract min, max, and median
min_value = statistics['min']
max_value = statistics['max']
median_value = statistics['50%']  # 50% corresponds to the median

# Print the results
print(f"Min: {min_value}")
print(f"Max: {max_value}")
print(f"Median: {median_value}")
```

```
Min: 4.0
Max: 7130.0
Median: 89.0
```

Fig: 4 Statistical calculations

To ascertain the mean and median of 'original_unit_price' in the Data Frame **merged_data_filtered_2**, the **describe()** function is employed. This function calculates

descriptive statistics, encompassing metrics such as count, mean, standard deviation, minimum, 25th percentile (1st quartile), median (50th percentile), 75th percentile (3rd quartile), and maximum. The resulting statistics object provides a comprehensive overview of the dataset's distribution and central tendency of the 'original_unit_price' column.

Moreover, Data visualization is crucial because it may aid in our understanding of the data. The data is displayed in this column using the Matplotlib and Seaborn libraries. In order to show the distribution of univariate values, it creates bar charts by iterating over certain columns and counting the frequency of unique values. The code is made to effectively investigate and learn about the categorical variables in the dataset, which will help in deciphering their distributions. The education level and age level of the majority of users are plotted using a bar graph, as all features in the user table are categorical.

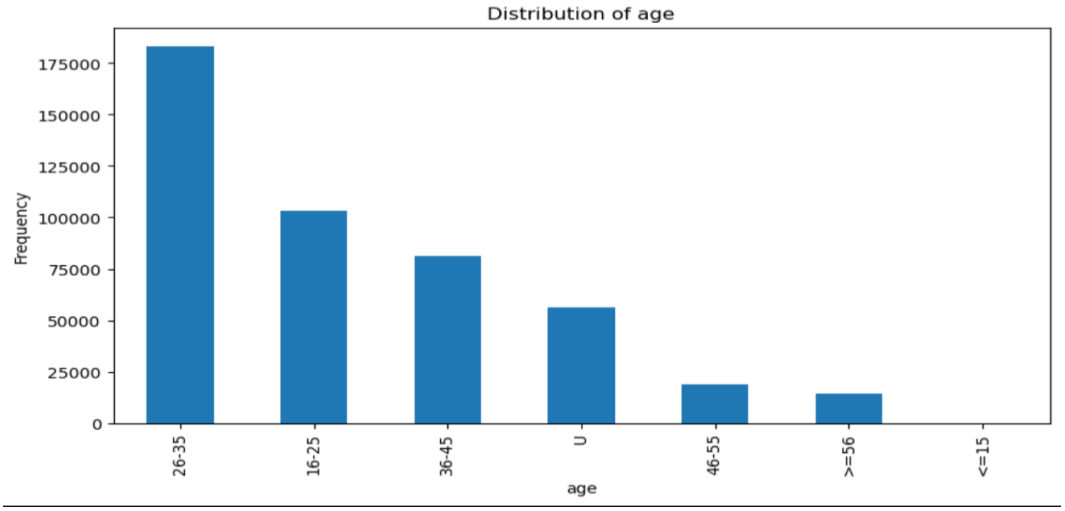

Fig: 5 Distribution of age for users

Considering the age distribution, it is evident that the age range of 26 to 35 years old has the greatest number of users, followed by 16 to 25 years old, which has over half of the higher user base. It is also evident that the age range of 56 and older has the fewest users, with no users under the age of 15.

The majority's education level in the distribution is 3, with -1, -2, and 1 being the lowest education levels. This is noted using the distribution of education level from Fig:6.

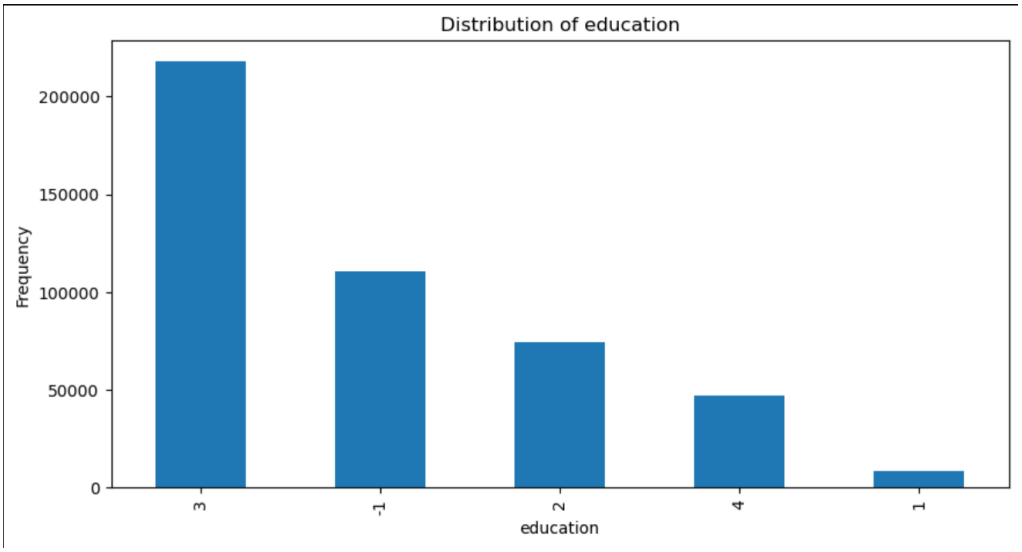

Fig: 6 distribution of education level

Fig: 7 Order quantity over day

Fig:7 shows the Customer Order Trend over the course of a day, that the ordered quantity saw a sharp increase starting at around the eighth hour and peaking at the tenth hour. From that point on until the twentieth hour, there was a rise and fall in the quantity ordered. The highest peak was reached at the tenth hour of the day.

To check the purchase level by city level of the users, the user table is grouped using a city_level variable, and the variable purchase power is added and depicted in Fig: 8. It can be observed that the purchasing power of users in city-level 2 is higher followed by the users in city level 1, 3, and 4 respectively.

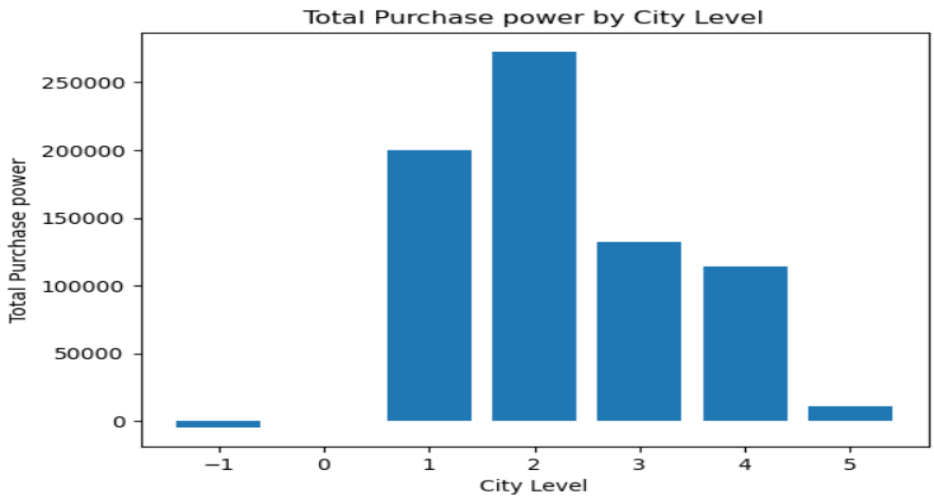

Fig: 8 Total purchase power by city level

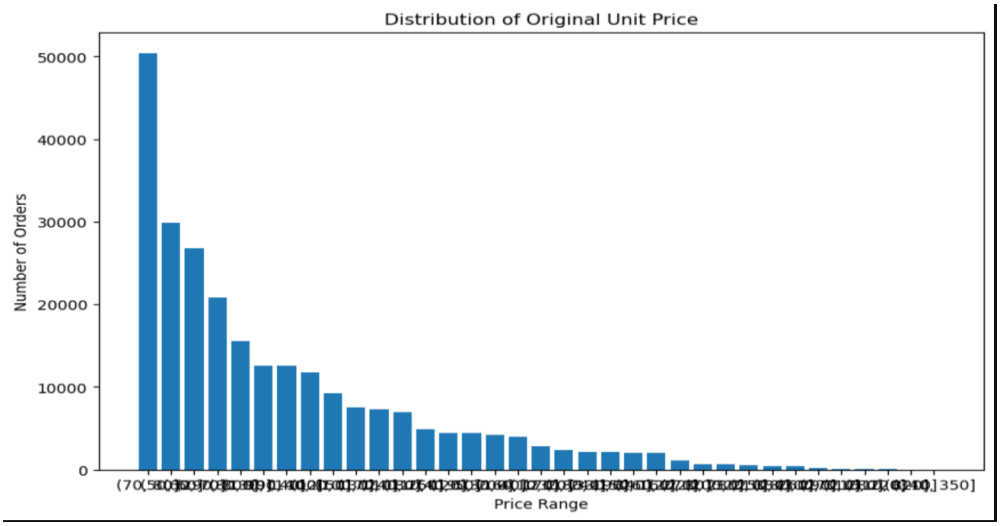

Fig: 9 Distribution of original unit price

Fig: 9 shows the distribution of the original unit price among the various orders and pricing ranges, which is noteworthy. The price range of 70–80 has the greatest number of orders, followed by 50, with almost 20,000 differences. It is evident that there hasn't been much of a difference in orders after the 50-price range, which has reduced the variation in the total number of orders.

Checking for typical order volumes for two kinds, 1P and 3P, of order differences. To ascertain whether there is a statistically significant difference between the means of the two groups, the t-test is utilized. Experiments and observational studies frequently use it to compare two distinct groups. T-test is performed in Python, as shown in Fig:10

```
# Perform a t-test
t_stat, p_value = stats.ttest_ind(quantity_1P, quantity_3P, equal_var=False)

print(f"T-Statistic: {t_stat}, P-value: {p_value}")

# Interpretation of results
if p_value < 0.05:
    print("There is a significant difference in order volumes between 1P and 3P orders.")
else:
    print("There is no significant difference in order volumes between 1P and 3P orders.")
```

Fig: 10 T-test and P-value determination

The t-statistic is then used to get the p-value. If there is no difference between the groups (null hypothesis), the p-value indicates the likelihood of getting test findings that are at least as dramatic as the actual results observed. A result is deemed statistically significant if the p-value is less than a predefined threshold, usually 0.05, indicating that it is unlikely that the observed difference happened by accident. It was found that the there is no significant difference in order volumes between 1P and 3P orders.

Models and Analysis

The methodology of predictive modeling involves the creation of a mathematical model or algorithm that, employing patterns encountered in current or historical data, can forecast or make predictions. Prediction models possess an assortment of configurations and are utilized to data analysis throughout numerous domains.

The order and user table were merged to calculate delivery time by subtracting the delivery time of a merged table and reference time. The delivery time for the orders were predicted using the Decision tree model. It has the ability to capture certain facets of the intricate relationships found in the data. A decision tree is a well-liked and simple machine learning model often used for both regression and classification applications. Every internal node in the structure resembles a choice made based on the value of a certain characteristic; every branch shows the result of that decision, and every leaf node shows the final prediction or classification. The structure is similar to a flowchart. Using decision trees, complex patterns and principles of decision-making were found in the data. The **Scikit-Learn** library is used to implement the decision tree model.

Before modeling the training and testing datasets are prepared with a test size of 0.33, followed by training of the decision tree model as shown in Fig: 11

```
regressorT = DecisionTreeRegressor(max_depth=20)
regressorT.fit(X_train, y_train)
```

Fig: 11 Train the decision tree model

The code implements a decision tree regressor in Python by using the scikit-learn module. A collection of features (X_train) and the target values that correspond to them (y_train) are used to train a decision tree regressor with a maximum depth of 20. Through the limitation of the decision tree's depth, the max_depth parameter regulates the model's complexity. RegressorT, the trained model that is produced as a consequence, may be used to forecast fresh data. To avoid overfitting, it is crucial to properly choose hyperparameters, such max_depth, depending on the unique features of the dataset and task.

The model was evaluated using root mean squared error (RMSE). The Root Mean Squared Error (RMSE) between the actual target values (y_test_numeric) and the projected values (y_predT) derived from a regression model is computed using the given Python code. A popular metric for assessing a regression model's performance is the root mean square error (RMSE), which calculates the average size of the prediction mistakes. The RMSE is obtained by taking the square root of the mean squared error (MSE) between the actual and predicted values, which is calculated using the sci-kit-learn mean_squared_error function. Reduced RMSE values indicate higher prediction accuracy. The resultant RMSE value offers a quantitative assessment of the regression model's performance on the test data. The calculated RMSE is printed by the print command for interpretation. The RMSE of the model is 6.61 as per Fig: 12

```
rmse = sqrt(mean_squared_error(y_test_numeric, y_predT))
print("Root Mean Squared Error:", rmse)
```

```
Root Mean Squared Error: 6.613093397123416e-09
```

Fig: 12 model evaluation using RMSE

Findings and Managerial Implications

The Findings of analysis of JD.com data are as follows

- The users with an age distribution of 26 – 35 years have the highest number of orders compared to others age groups
- The users with a higher level of education have the highest number of orders, and the gender of users also has an effect on number of orders. Females placed more number of orders.
- The number of orders during the days of the month is plotted using a line graph. It was observed that the orders placed in the month's first and second weeks were more than in the other weeks.
- The line graph is plotted to observe the order quantity for the hour of the day. It was observed that the number of orders placed increases from 5:00 am to 10:00 am, gradually decreases till 8:00 pm, and then increases till 10:00 pm.
- The most number of orders placed with the price range of 70 – 80.
- The purchasing power of the users also depends on the city level. The users in city level 2 placed the highest number of orders than other city levels.
- There is no significant difference between the order volumes in 1P and 3P orders
- The delivery time for orders is predicted using the decision tree model and is evaluated using RSME with the RSME value of 6.6

Managerial Implications

Strategic managerial decisions can be made using the valuable insights into customer behavior and preferences obtained by JD.com's data analysis observations. Based on each observation, the following are the managerial implications.

- Marketing and product tactics should be adapted to appeal to people in the 26–35 age range, as they have the highest order volume. This could entail choosing goods, services, and marketing initiatives that complement their hobbies and way of life.
- The pattern of increased orders during the first two weeks of the month points to customers' possible post-payday purchasing habits. JD.com might take advantage of this to further increase sales by planning deals, promotions, or the release of new products around certain periods.

- There appear to be two peak shopping hours based on the rise in orders from 5:00 am to 10:00 am and again in the evening. JD.com has the capacity to efficiently manage the spike in orders during these hours by optimizing its workforce, customer service, and logistics. Flash sales or exclusive offers could also be scheduled to fall during these busy periods to optimize income.
- The predominance of products priced between 70 and 80 suggests that consumers are price-sensitive. To encourage larger order quantities, JD.com can concentrate on providing competitive prices within this range, combine products to match this price point, or give discounts.
- Given that city level 2 consumers place the most orders, JD.com ought to think about boosting its marketing and infrastructure for deliveries in these regions. This could entail expanding the number of distribution facilities, forming alliances with nearby companies, or adjusting the product mix to suit local consumers' tastes.
- An RSME value of 6.6 indicates potential improvement in logistics, and the delivery time may be predicted using a decision tree model. To improve customer happiness, JD.com can concentrate on lowering this error margin. Introducing predictive analytics for improved stock placement, streamlining the delivery process, and optimizing routes may be beneficial.

Conclusion

In conclusion, JD.com's order patterns and consumer behavior data analysis provide important new information for strategic business choices. The most important lessons learned include the value of focusing on the 26–35 age group, satisfying the needs of well-educated female consumers, and utilizing temporal sales patterns to maximize marketing and promotions. The research also emphasizes the importance of regional targeting, especially in city-level 2 locations, and shows the necessity of price-sensitive methods in several product ranges. In order to improve customer happiness, the results also highlight how crucial it is to optimize delivery schedules and logistics. By strategically using these insights, JD.com can customize its strategy to satisfy customer requests, enhance operational effectiveness, and propel growth and profitability within a competitive market.

References

Shen, M. Tang, S.C. Wu, D. Yuan, R. Zhou, W. (2019),” JD.com: Transactional data for the 2020 MSOM Data-Driven Research Challenge”, *MSOM Research Challenge Data Paper*, JD.Com