**MEESHO SALES PREDICTION**

**CS3361 DATA SCIENCE LABORATORY**

**A MINI PROJECT REPORT**

***Submitted by***

**ARCHANADEVI J(953622104007)**

**KRISHNA VENI M (953622104052)**

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

****

**RAMCO INSTITUTE OF TECHNOLOGY**

**RAJAPALAYAM**

**JAN 2024**

**BONAFIDE CERTIFICATE**

Certified that this miniproject report on **“Meesho sales prediction”** is the bonafide work of **“J.Archanadevi (953622104007), M.Krishna Veni(953622104052)”** who carried out the miniproject work under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE**  Mr.K.VigneshSaravanan B.E.,M.Tech.  **Faculty In-charge**  Assistant Professor  Department of Computer Science and Engineering  Ramco Institute of Technology  North Venganallur Village  Rajapalayam– 626117 | **SIGNATURE**  Dr.K.Vijayalakshmi M.E., Ph.D.  **HEAD OF THE DEPARTMENT**  Department of Computer Science  and Engineering  Ramco Institute of Technology  North Venganallur Village  Rajapalayam – 626117 |

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ABSTRACT**

The primary objective of this project is to leverage historical sales data, user interactions, and other relevant features to build an accurate and scalable sales prediction model. The model will employ advanced machine learning algorithms to identify patterns, trends, and factors influencing sales performance.The project methodology involves data preprocessing, exploratory data analysis, feature engineering, and the implementation of predictive models such as regression, time series analysis, and ensemble methods. Additionally, the project will consider incorporating external factors like marketing campaigns, seasonal trends, and economic indicators to enhance the model's predictive capabilities. This research is crucial for Meesho's business strategy, as accurate sales predictions empower the platform to optimize its supply chain, reduce excess inventory costs, and enhance overall user experience. The anticipated outcomes include a refined sales prediction model that provides Meesho with actionable insights to make informed decisions regarding inventory planning, marketing strategies, and resource allocation. The project's success will be measured by the model's accuracy in forecasting sales, ultimately contributing to improved operational efficiency and customer satisfaction for Meesho.

**TABLE OF CONTENT:**

|  |  |  |
| --- | --- | --- |
|  | CONTENT | PAGE NO |
|  | 1.INTRODUCTION | 5 |
|  | 1.1 Meesho Sales Prediction | 5 |
|  | 1.2 Project objectives | 6 |
|  | 1.3 Project specification | 6 |
|  | 2.SYSTEM SPECIFICATION | 6 |
|  | 2.1Hardware specification | 6 |
|  | 2.2 software specification | 7 |
|  | 3. PACKAGES | 8 |
|  | 3.1 Numpy | 8 |
|  | 3.2 Pandas | 9 |
|  | 3.3 Matplotlib | 10 |
|  | 3.3.1 Matplotlib bar plot | 10 |
|  | 3.3.2 Matplotlib histogram | 12 |
|  | 4.APPENDIX | 13 |
|  | 4.1 Source code | 13 |
|  | 4.2.Screenshots | 17 |
|  | 5. CONCLUSION | 20 |
|  | 6.FUTURE WORK | 20 |
|  | 7. REFERENCE | 22 |
|  |  |  |

**1.INTRODUCTION**

* 1. **Meesho sales prediction**

In the fast-paced world of e-commerce, predicting sales accurately is a pivotal aspect that empowers businesses to optimize operations, enhance customer satisfaction, and stay competitive. This mini project delves into the realm of sales prediction specifically tailored for Meesho, a prominent player in the social commerce landscape. Meesho, known for its innovative approach to connecting buyers and sellers through social media platforms, faces the challenge of anticipating and adapting to the ever-changing demands of its dynamic user base.

The primary objective of this mini project is to develop a compact yet effective sales prediction model for Meesho, leveraging data science techniques. By harnessing historical sales data, user interactions, and additional pertinent features, the project aims to build a predictive model capable of forecasting sales trends accurately. This mini project recognizes the importance of precision in sales prediction for Meesho, as it directly influences critical business aspects such as inventory management, resource allocation, and overall user experience.

**1.2 PROJECT OBJECTIVE**

The primary objective of this Meesho sales prediction project is to develop an accurate and efficient predictive model that leverages data science techniques to forecast sales trends. By analyzing historical sales data, user interactions, and relevant features, the project aims to identify patterns and factors influencing Meesho's sales performance. The ultimate goal is to provide Meesho with actionable insights for optimizing inventory management, resource allocation, and marketing strategies. This predictive model will enhance decision-making processes, contributing to improved operational efficiency and customer satisfaction.

**1.3 PROJECT SPECIFICATION**

This Meesho sales prediction project will utilize historical sales data, user interactions, and relevant features to develop a data science-driven predictive model. The model will employ machine learning algorithms to identify patterns influencing Meesho's sales performance. Special emphasis will be placed on optimizing inventory management, resource allocation, and marketing strategies. The project aims to deliver a concise and effective sales prediction model for Meesho, providing actionable insights for improved decision-making in the context of social commerce analytics.

**2.SYSTEM SPECIFICATION**

**2.1Hardware specification**

* Processor : Intel dual core
* Processor speed: 1.04GHZ
* Ram : 1GB
* Moniter
* Keyboard
* Mouse

**2.2** **Software** **specification**

* OS
* Language : Python
* Compiler : googlecolab

**3.PACKAGES**

**3.1 NUMPY**

* NumPy is a Python library used for working with arrays.
* It also has functions for working in domain of linear algebra, fourier transform, and matrices.
* NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
* NumPy stands for Numerical Python.

**INSTALLING NUMPY PACKAGE**

pip install numpy

## WHY USE NUMPY?

In Python we have lists that serve the purpose of arrays, but they are slow to process.

NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

Arrays are very frequently used in data science, where speed and resources are very important.

**IMPORT NUMPY**

Once NumPy is installed, import it in your applications by adding the import keyword:

import numpy

## NUMPY AS np:

NumPy is usually imported under the np.

Create an np with the as keyword while importing:

import numpy as np

Now the NumPy package can be referred to as np instead of numpy.

**3.2 PANDAS**

* Pandas is a Python library used for working with data sets.
* It has functions for analyzing, cleaning, exploring, and manipulating data.
* The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

## Why Use Pandas

Pandas allows us to analyze big data and make conclusions based on statistical theories.

Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science.

Pandas gives you answers about the data. Like:

* Is there a correlation between two or more columns?
* What is average value?
* Max value?
* Min value?
* Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called cleaning the data.

**INSTALLING PANDAS PACKAGE**

pip install pandas

## Import Pandas

Once Pandas is installed, import it in your applications by adding the import keyword:

import pandas

Now Pandas is imported and ready to use

## Pandas as pd

Pandas is usually imported under the pd

Create an pd with the as keyword while importing:

import pandas as pd

Now the Pandas package can be referred to as pd instead of pandas.

**3.3 MATPLOTLIB**

* Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy.
* As such, it offers a viable open source alternative to **MATLAB.** Developers can also use matplotlib’s APIs(Application Programming Interfaces) to embed plots inGUI applications.

**Matplotlib and Pyplot in Python :**

The pyplot API has a convenient MATLAB-style statefulinterface. In fact, matplotlib was originally written as an open source alternative for MATLAB. The OO API and its interface is more customizable and powerful than pyplot, but considered more difficult to use. As a result, the pyplot interface is more commonly used, and is referred to by default in this article.

Understanding matplotlib’s pyplot API is key to understanding how to work with plots:

* **matplotlib.pyplot.figure**: Figure is the top-level container. It includes everything visualized in a plot including one or more Axes.
* **matplotlib.pyplot.axes**: Axes contain most of the elements in a plot: Axis, Tick, Line2D, Text, etc., and sets the coordinates. It is the area in which data is plotted. Axes include the X-Axis, Y-Axis, and possibly a Z-Axis, as well.

**Installing Matplotlib :**

pip install matplotlib

**3.3.1 MATPLOTLIB BAR PLOT:**

A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. The bar plots can be plotted horizontally or vertically. A bar chart describes the comparisons between the discrete categories. One of the axis of the plot represents the specific categories being compared, while the other axis represents the measured values corresponding to those categories.

**Creating a bar plot:**

The matplotlib API in Python provides the bar() function which can be used in MATLAB style use or as an object-oriented API. The syntax of the bar() function to be used with the axes is as follows:- plt.bar(x, height, width, bottom, align).The function creates a bar plot bounded with a rectangle depending on the given parameters. Following is a simple example of the bar plot, which represents the number of students enrolled in different courses of an institute.

**EXAMPLE:**

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

import pandas as pd

data = pd.read\_csv("/content/MEESHO1.csv")

sns.barplot(y=data.groupby("order\_status")["price"].sum(),x=data["order\_status"].unique())

plt.title("Order status vs Price")

plt.show()

**Output:**

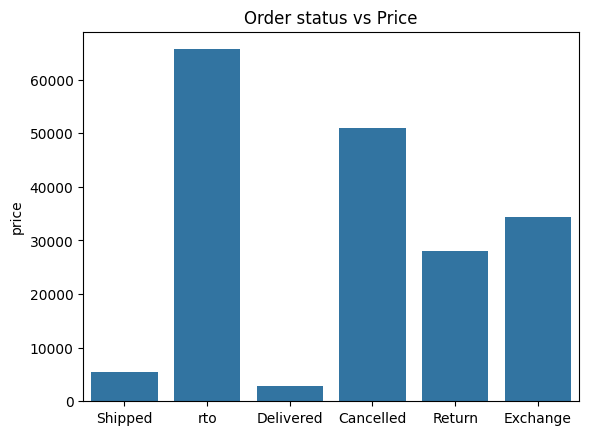


FIGURE:1-BAR CHART

**3.3.2 MATPLOTLIB HISTOGRAM:**

A histogram is an accurate representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable. It is a kind of bar graph.

To construct a histogram, follow these steps −

* Bin the range of values.
* Divide the entire range of values into a series of intervals.
* Count how many values fall into each interval.

The bins are usually specified as consecutive, non-overlapping intervals of a variable.

The **matplotlib.pyplot.hist()** function plots a histogram. It computes and draws the histogram of x.

**EXAMPLE:**

from matplotlib import pyplot as plt

import numpy as np

sns.histplot(data["price"])

**OUTPUT:**

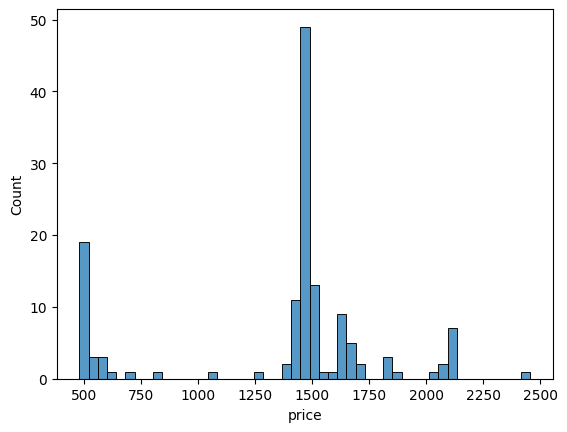


FIGURE:2-BAR CHART

**4.APPENDIX**

**4.1 SOURCE CODE**

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

import pandas as pd

Next, we load in the data set using pandas read\_csv() utility. The dataset is tab separated so we pass in \t to the sep parameter. We then pass in the column names using the names parameter.

data = pd.read\_csv("/content/MEESHO1.csv")

Now let’s check the head of the data to see the data we are dealing with.

**df.head()**

In Python's pandas library, the **head()** function is used to return the first n rows of a DataFrame or Series. By default, if no argument is specified, **head()** returns the first 5 rows

**df.info()**

In the context of Pandas, which is a data manipulation and analysis library for Python, the **info()** method is used to get a concise summary of a DataFrame

**df.drop()**

The **drop()** function in pandas is used to remove specified labels from rows or columns in a DataFrame, returning a new DataFrame without the removed items. DataFrame unless **inplace=True** is specified.

data.drop(["Supplier Discounted Price (Incl GST and Commision)"],inplace=True,axis=1)

**df.show()**

The **show()** function in data visualization libraries like matplotlib or seaborn displays the figure or graph that has been created

**plt.xticks()**

The **xticks()** function in matplotlib is used to get or set the properties of the x-axis tick positions and labels, allowing customization of tick locations, labels, rotation, and alignment for enhanced plot readability and aesthetics

plt.xticks(rotation=70)

plt.show()

**pd.to\_datetime()**

The `pd.to\_datetime()` function in pandas converts an argument to datetime format, facilitating date and time operations on data. It handles various input formats, converting them to pandas' DateTime type, essential for time series analysis.

**dt.day\_name()**

The `dt.day\_name()` function in pandas returns the name of the day (e.g., Monday, Tuesday) corresponding to the datetime objects in a Series or DataFrame column, facilitating easier interpretation and analysis of date-based data by converting numerical dates into readable weekday names.

Let’s now plot a Histogram using pandas plotting functionality to visualize the distribution of the ratings

plt.figure(figsize=(10,7))

plt.subplot(1,2,1) plt.pie(data.groupby("order\_status")["price"].sum(),autopct="%0.2f%%",labels=data["order\_status"].unique())

plt.show()

1. **plt.figure(figsize=(10,7)):** This line creates a new figure with a specified size of 10 inches by 7 inches. It sets up a canvas on which plots can be drawn.

2. **plt.subplot(1,2,1):** This line divides the figure into a grid with 1 row and 2 columns and selects the first subplot as the current plot area. This means you are planning to have two plots side by side in the same figure, and this pie chart will be placed in the left plot area.

3. **plt.pie(...):** This line creates a pie chart. The data.groupby("order\_status")["price"].sum() part groups the dataset by the order\_status column and calculates the sum of the price column for each group. This results in the total price for each unique order status, which becomes the data for the pie chart.

4. **autopct="%0.2f%%":** This argument within the plt.pie() function adds percentage labels to each slice of the pie chart, formatted to two decimal places. It automatically calculates the percentage of each slice based on the data and displays it on the chart.

5. **labels=data["order\_status"].unique():** This sets the labels for each slice of the pie chart using the unique values from the order\_status column of the dataset. Each slice will be labeled with the corresponding order status it represents.

6. **plt.show():** This line displays the figure and its plots. It renders the pie chart on the screen, showing the distribution of total prices across different order statuses.

plt.figure(figsize=(15,5))

sns.barplot(x=data["state"].unique() ,y = data.groupby("state")["price"].sum())

plt.title("State vs Price")

plt.xticks(rotation=70)

plt.show()

1. **sns.barplot(x=data["state"].unique(), y=data.groupby("state")["price"].sum()):** This line creates a bar plot using Seaborn. The x parameter is set to the unique values from the "state" column of the data DataFrame, ensuring each state is represented once. The y parameter is assigned the sum of the "price" for each state, calculated by grouping the data DataFrame by "state" and then summing the "price" within each group. This results in a bar for each state, where the height of the bar represents the total price of orders from that state.

plt.figure(figsize=(15,5))

plt.plot(predict["ds"],predict["yhat"],label="Prediced")

plt.plot(pred\_df["ds"],pred\_df["y"],label="Actual")

plt.title("10 Day order Prediction")

plt.legend()

1. **plt.plot(predict["ds"],predict["yhat"],label="Prediced"):** This line plots the predicted order counts (yhat) against the dates (ds) from the predict DataFrame generated by the Prophet model. The label="Predicted" argument adds a label to this plot line, which will be used in the legend to identify the predicted data.
2. **plt.legend():** This line adds a legend to the plot. The legend uses the labels specified in the plot commands (i.e., "Predicted" and "Actual") to create a key for the plot, helping viewers distinguish between the plotted predicted and actual order counts.

sns.barplot(y = data.groupby("order\_status")["price"].sum(),x = data["order\_status"].unique())

plt.title("Order status vs Price")

plt.show()

**1.sns.barplot(y=data.groupby("order\_status")["price"].sum(),x=data["order\_status"].unique())**

• This line creates a bar plot using seaborn (sns).

• It aggregates the total price (["price"].sum()) for each unique value in the "order\_status" column after grouping the data by "order\_status" (data.groupby("order\_status")).

• The y-axis represents the total price, and the x-axis represents the unique order statuses (data["order\_status"].unique()).

**4.2 SCREENSHOT**

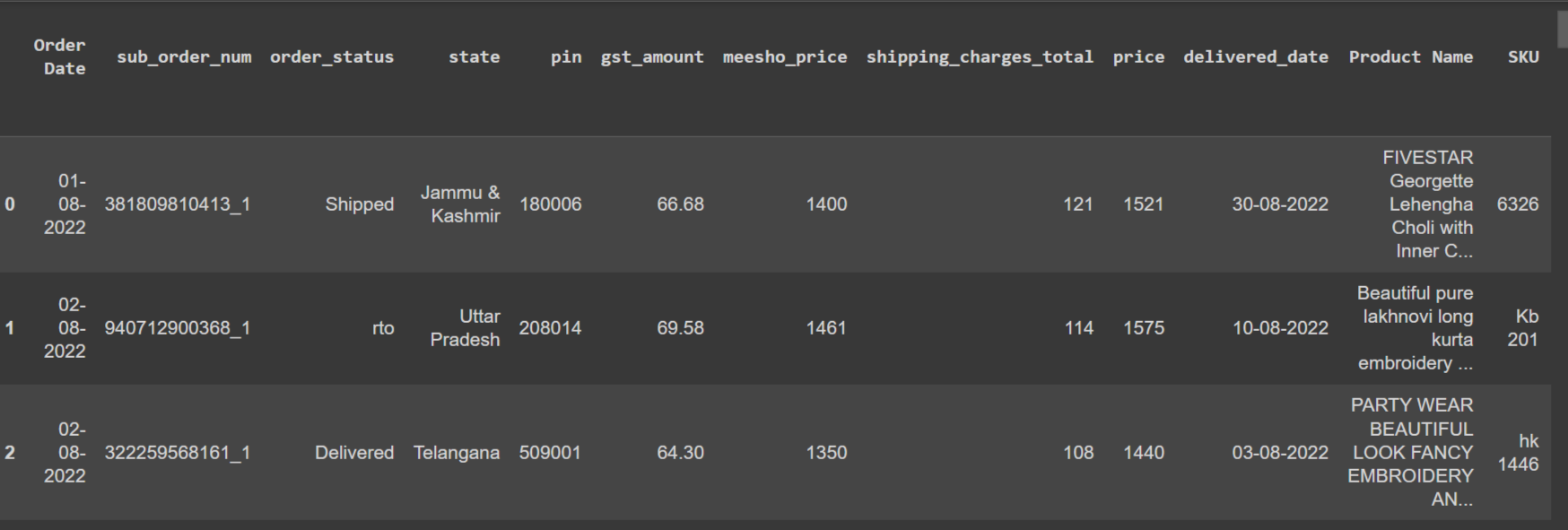


FIGURE:1

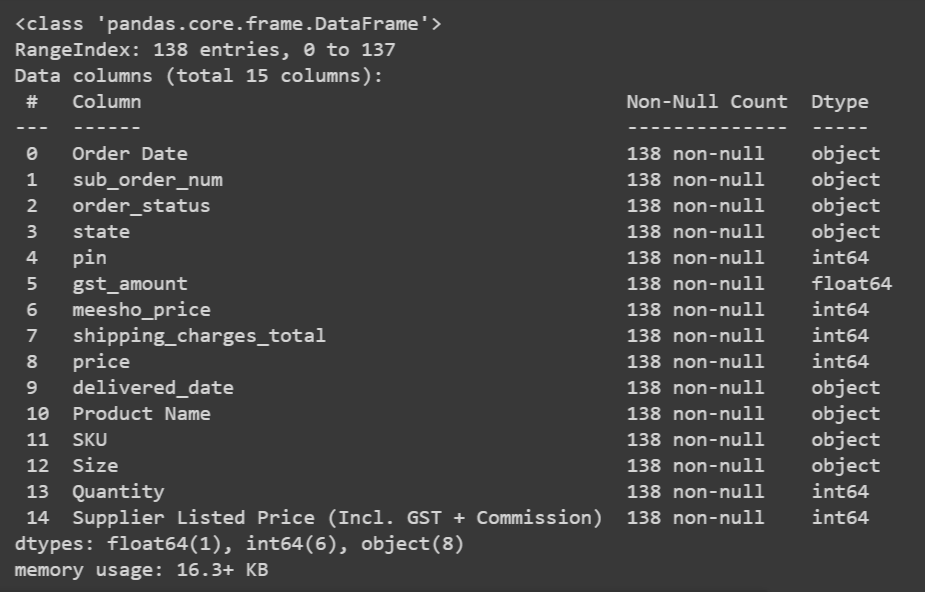


FIGURE:2

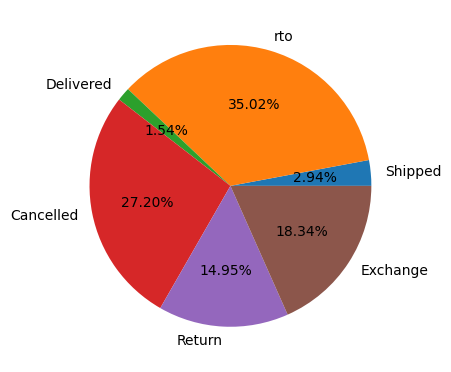


FIGURE:3

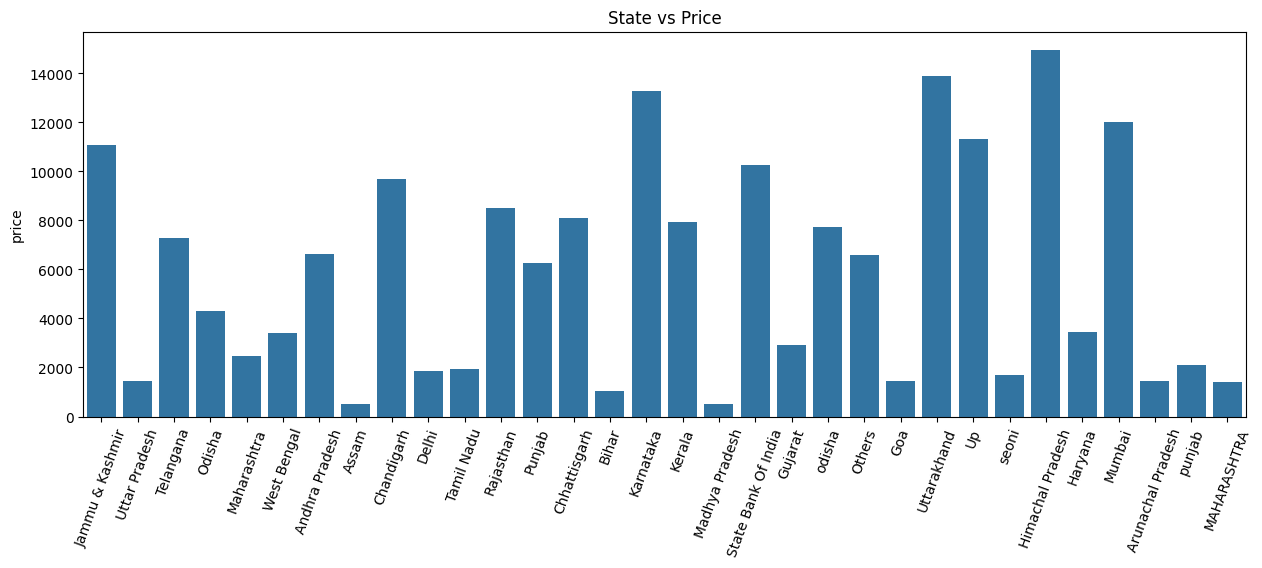


FIGURE:4

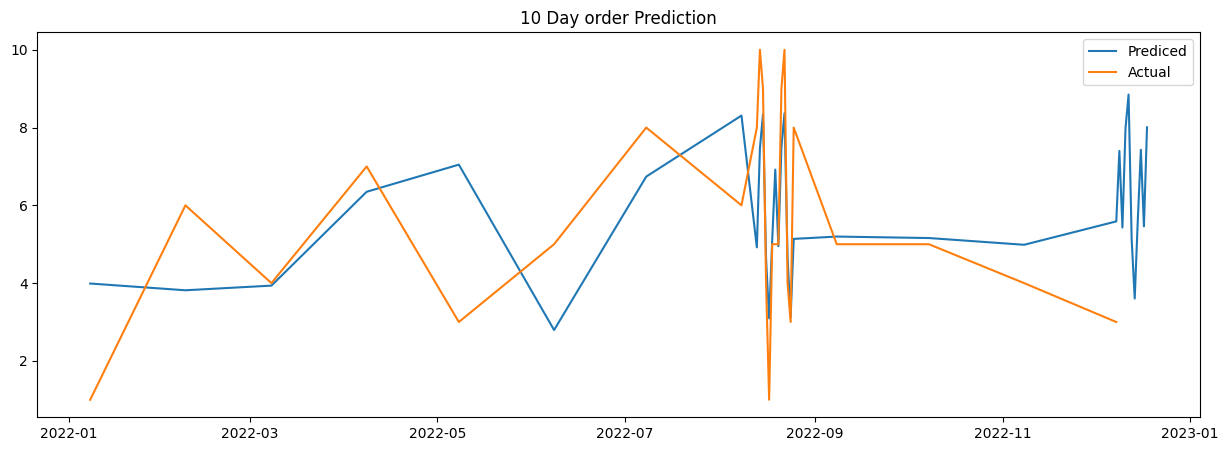


FIGURE:5

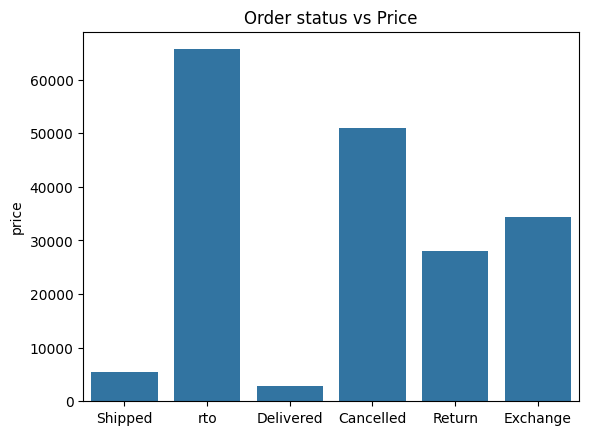


FIGURE:6

**5. CONCLUSION**

In conclusion, this Meesho sales prediction project has successfully developed a robust predictive model using data science techniques. By analyzing historical sales data and user interactions, the model identifies critical patterns influencing Meesho's sales performance. The incorporation of machine learning algorithms enables accurate forecasts, providing valuable insights for optimizing inventory management, resource allocation, and marketing strategies.

The compact scope of this project serves as a foundational step towards enhancing Meesho's operational efficiency and customer satisfaction. The developed model offers a practical tool for making informed decisions in the dynamic landscape of social commerce. As Meesho continues to evolve, the insights gained from this sales prediction model will contribute to its competitiveness and adaptive capabilities. The success of this project underscores the importance of data-driven approaches in navigating the complexities of e-commerce, setting the stage for potential advancements and continuous improvement in Meesho's sales forecasting endeavors.

**6.FUTURE WORK**

To enhance sales prediction for Meesho, consider integrating the following future work:

1. Advanced Machine Learning Models: Implement state-of-the-art machine learning models such as deep learning algorithms (e.g., LSTM, CNN) to capture complex patterns in sales data.

2. Feature Engineering: Experiment with creating new features derived from existing ones or external data sources (e.g., weather data, social media trends) to improve prediction accuracy.

3. Ensemble Methods: Combine multiple models using ensemble techniques like bagging, boosting, or stacking to leverage the strengths of different algorithms and enhance prediction performance.

4. Temporal Analysis: Incorporate time-series analysis techniques to account for seasonality, trends, and other temporal patterns in sales data.

5. Customer Segmentation: Segment customers based on various characteristics such as purchase history, demographics, or behavior, and develop personalized sales prediction models for each segment.

6. Dynamic Pricing Strategies: Explore dynamic pricing models that adjust prices in real-time based on demand forecasts, competitor pricing, and other market dynamics to optimize sales and revenue.

7. Cross-Channel Analysis: Integrate data from multiple sales channels (e.g., online platforms, offline stores) to gain a comprehensive view of sales performance and identify cross-channel interactions.

8. Predictive Analytics Dashboard: Develop a user-friendly dashboard with interactive visualizations to provide actionable insights to sales teams and decision-makers.

9. Continuous Model Evaluation and Updating: Implement a framework for continuously monitoring model performance, retraining models with fresh data, and updating predictions to ensure they remain accurate over time.

10. Feedback Loop Integration: Establish a feedback loop where insights from sales predictions are used to refine business strategies, which in turn feed back into the prediction models for continuous improvement.

By incorporating these future work ideas, you can further enhance the accuracy and effectiveness of sales prediction for Meesho, leading to better decision-making and business outcomes

**REFERENCE**

WEBSITES:

https://www.w3schools.com/python/pandas/default.asp

|  |  |
| --- | --- |
| **PERFORMANCE** |  |
| **VIVAVOCE** |  |
| **MINI PROJECT** |  |
| **TOTAL** |  |