On-Time Delivery Prediction

Problem statement

We want to predict whether a product will be delivered on time or late, based on factors such as:

- Delivery person rating
- Shipment weight
- Mode of shipment

Why it matters?

Helps logistics companies reduce delays, improve customer satisfaction, and optimize resources.

Dataset Overview

- Source: Kaggle – Customer Analytics Dataset
- Size: ~11,000 records
- Target variable: Reached.on.Time_Y.N (1 = On Time, 0 = Late)
- Important features:
  - Delivery_person_rating
  - Weight_in_gms
  - Mode_of_Shipment

Model Used

Algorithm: Logistic Regression

Benfits:

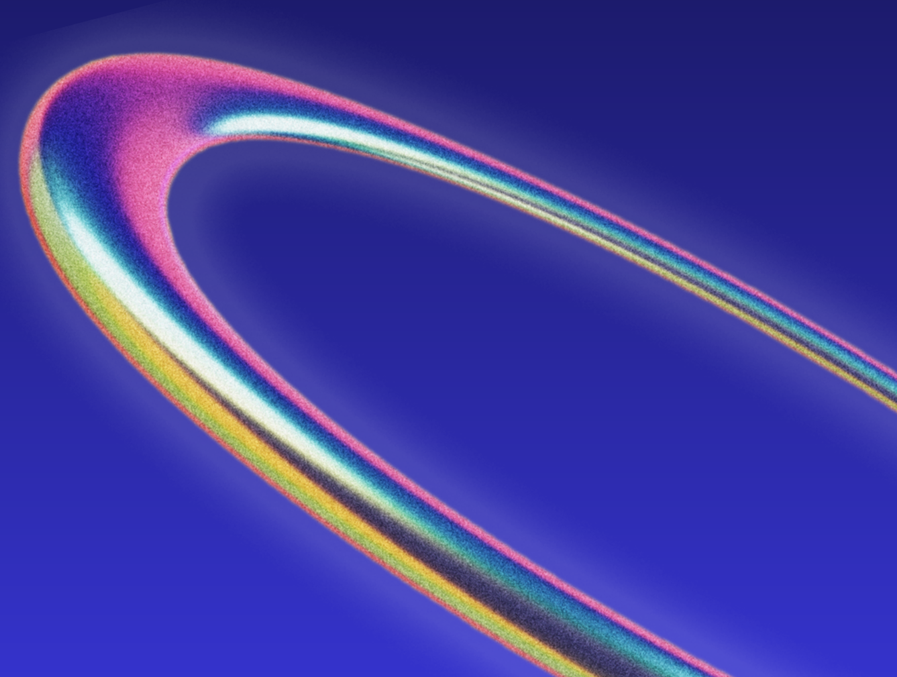
1. Works well for binary classification (on time vs. late)
2. Fast and efficient for structured data

Step 1: Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
```

Step 2: Load the Dataset

```
data = pd.read_csv('Train.csv')
```

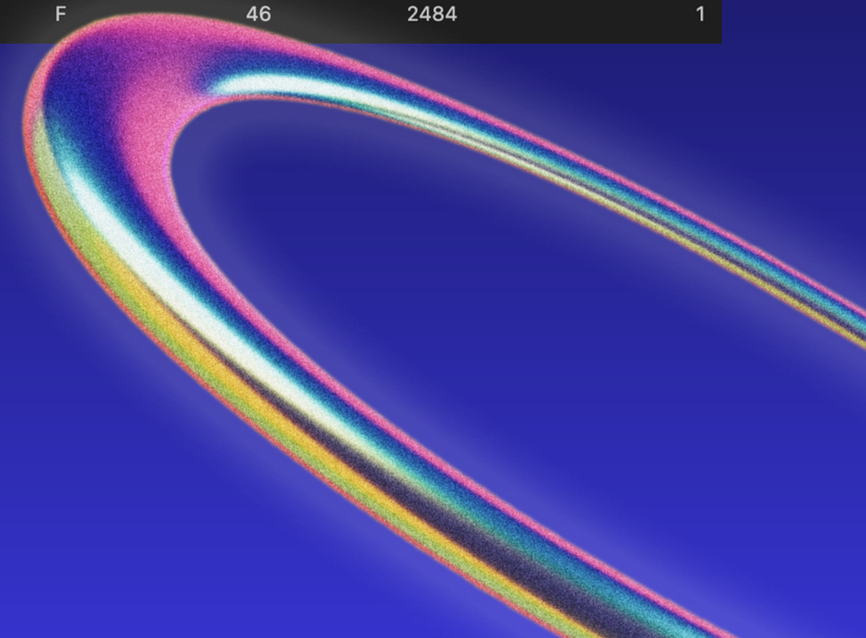


Step 3: Preview the Data

```
data.head()
```

Python

rehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender	Discount_offered	Weight_in_gms	Reached.on.Time_Y.N
D	Flight	4	2	177	3	low	F	44	1233	1
F	Flight	4	5	216	2	low	M	59	3088	1
A	Flight	2	2	183	4	low	M	48	3374	1
B	Flight	3	3	176	4	medium	M	10	1177	1
C	Flight	2	2	184	3	medium	F	46	2484	1



Step 4: Basic Information

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10999 entries, 0 to 10998
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   10999 non-null  int64
 1   Warehouse_block      10999 non-null  object
 2   Mode_of_Shipment     10999 non-null  object
 3   Customer_care_calls  10999 non-null  int64
 4   Customer_rating      10999 non-null  int64
 5   Cost_of_the_Product  10999 non-null  int64
 6   Prior_purchases      10999 non-null  int64
 7   Product_importance   10999 non-null  object
 8   Gender               10999 non-null  object
 9   Discount_offered     10999 non-null  int64
 10  Weight_in_gms        10999 non-null  int64
 11  Reached.on.Time_Y.N  10999 non-null  int64
dtypes: int64(8), object(4)
memory usage: 1.0+ MB
```

```
data.describe()
```

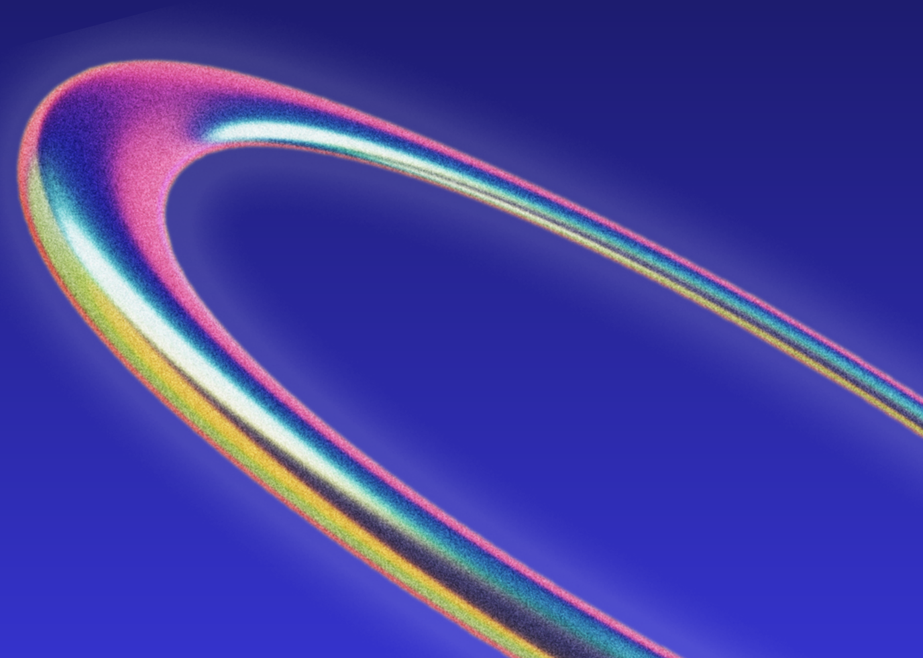
	ID	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Discount_offered	Weight_in_gms	Reached.on.Time_Y.N
count	10999.00000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000
mean	5500.00000	4.054459	2.990545	210.196836	3.567597	13.373216	3634.016729	0.596691
std	3175.28214	1.141490	1.413603	48.063272	1.522860	16.205527	1635.377251	0.490584
min	1.00000	2.000000	1.000000	96.000000	2.000000	1.000000	1001.000000	0.000000
25%	2750.50000	3.000000	2.000000	169.000000	3.000000	4.000000	1839.500000	0.000000
50%	5500.00000	4.000000	3.000000	214.000000	3.000000	7.000000	4149.000000	1.000000
75%	8249.50000	5.000000	4.000000	251.000000	4.000000	10.000000	5050.000000	1.000000
max	10999.00000	7.000000	5.000000	310.000000	10.000000	65.000000	7846.000000	1.000000

Step 5: Check for Missing Values

```
data.isnull().sum()
```

	0
ID	0
Warehouse_block	0
Mode_of_Shipment	0
Customer_care_calls	0
Customer_rating	0
Cost_of_the_Product	0
Prior_purchases	0
Product_importance	0
Gender	0
Discount_offered	0
Weight_in_gms	0
Reached.on.Time_Y.N	0

dtype: int64



Accuracy & Evaluation

- Accuracy: ~64%
- Confusion Matrix, Precision, Recall, F1-Score
- (You can show your printed classification report)
- Visuals: Add heatmap or bar plot

Challenges:

1. The dataset was not very realistic.
2. Choosing the right visualizations was difficult.

Solutions:

1. We accepted that practice datasets can still be useful, even if not perfect.
2. We tried different plots and learned which visuals worked best.

Future Work

1. Use better or real delivery data.
2. Try more models to improve results.
3. Make the results easier to understand for others.
4. Add simple tools to help users see the data clearly.

Step 6: Exploratory Data Analysis

One Type of EDA:
Data Visualization :
Visual exploration is a powerful way to gain insights from data. It helps to identify distributions, correlations, and anomalies quickly and intuitively.

Common Visual Tools in EDA:

- **Histogram: Shows the distribution of a single variable.**
- **Boxplot: Highlights the spread of the data and outliers.**
- **Scatter Plot: Explores the relationship between two continuous variables.**
- **Heatmap: Displays correlations between multiple variables.**
- **Pair Plot: Visualizes pairwise relationships across multiple features.**

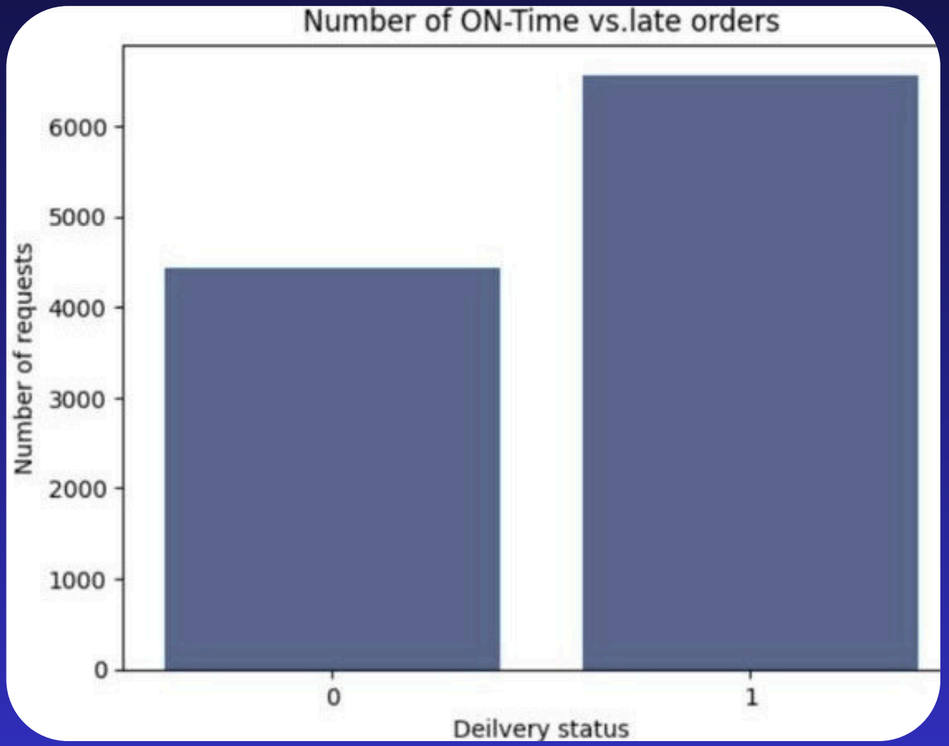
Exploratory Data Analysis : Bar Chart

On-Time vs. Late Deliveries

- **X-axis: Delivery Status**
- **0 = On-Time**
- **1 = Late**
- **Y-axis: Number of Orders**
- **Observation:**

Late deliveries (~6700) are much higher than on-time deliveries (~4400).

There is a delivery performance issue — more orders are arriving late than on time



Exploratory Data Analysis : Box Plot

Delivery Rating vs. Delivery Status

- Ratings for delivery persons are similar whether orders arrive on time (1) or late (0).
- Median rating is ~3 in both cases.

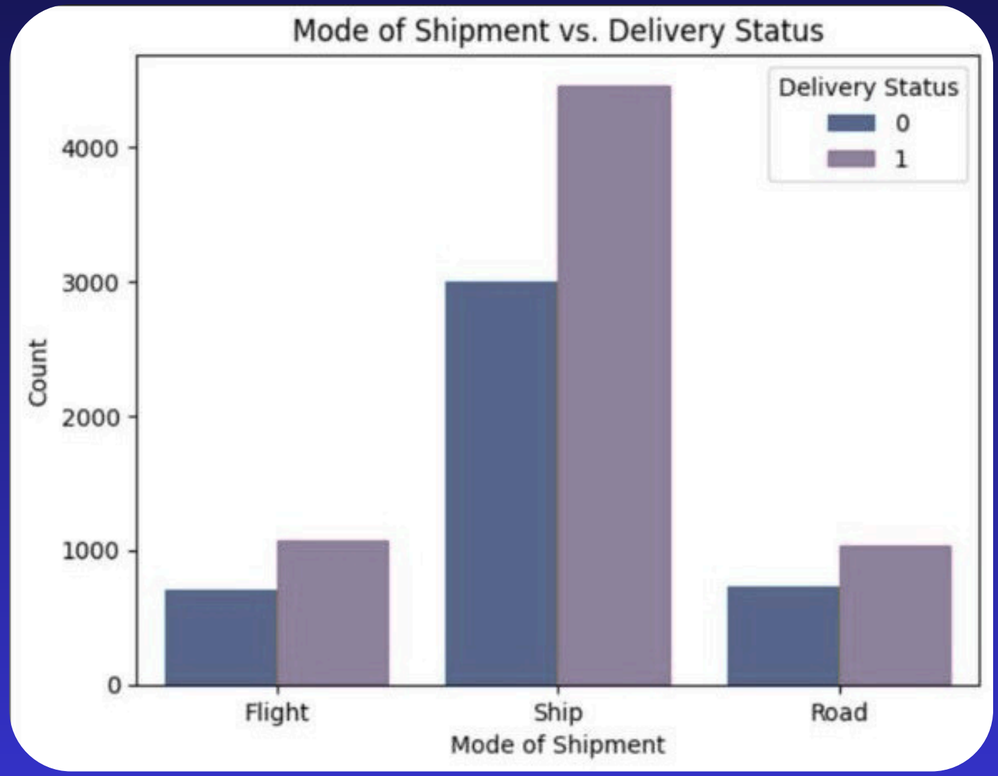
Delivery timing does not strongly impact customer ratings of delivery personnel.

Exploratory Data Analysis : Grouped Bar Chart

Shipment Mode vs. Delivery Status

- Most orders were shipped by sea (Ship).
- Shipments by sea had the highest on-time deliveries.
- Flight and road shipments showed smaller differences between late and on-time deliveries.

Sea shipping is both the most used and most reliable mode in terms of delivery timing.

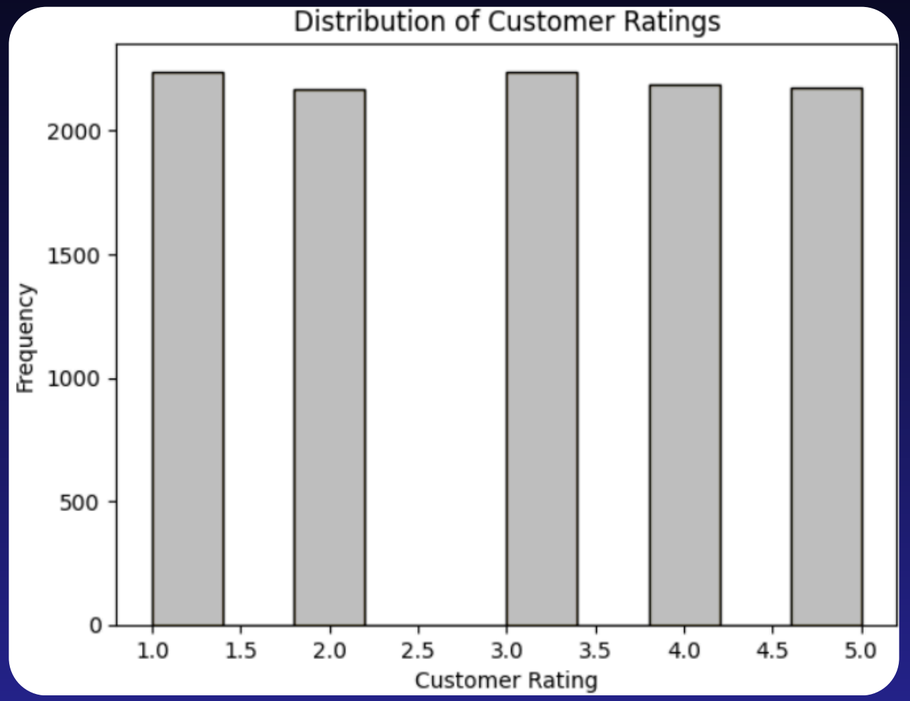


Discount Offered vs Delivery Status

More discount → On time delivery!

```
sns.boxplot(x='Reached.on.Time_Y.N', y='Discount_offered', data=data)
plt.title("Discount Offered vs Delivery Status")
plt.xlabel("Reached on Time (1 = Yes, 0 = No)")
plt.ylabel("Discount Offered")
plt.tight_layout()
plt.show()
```

Distribution of Customer Ratings

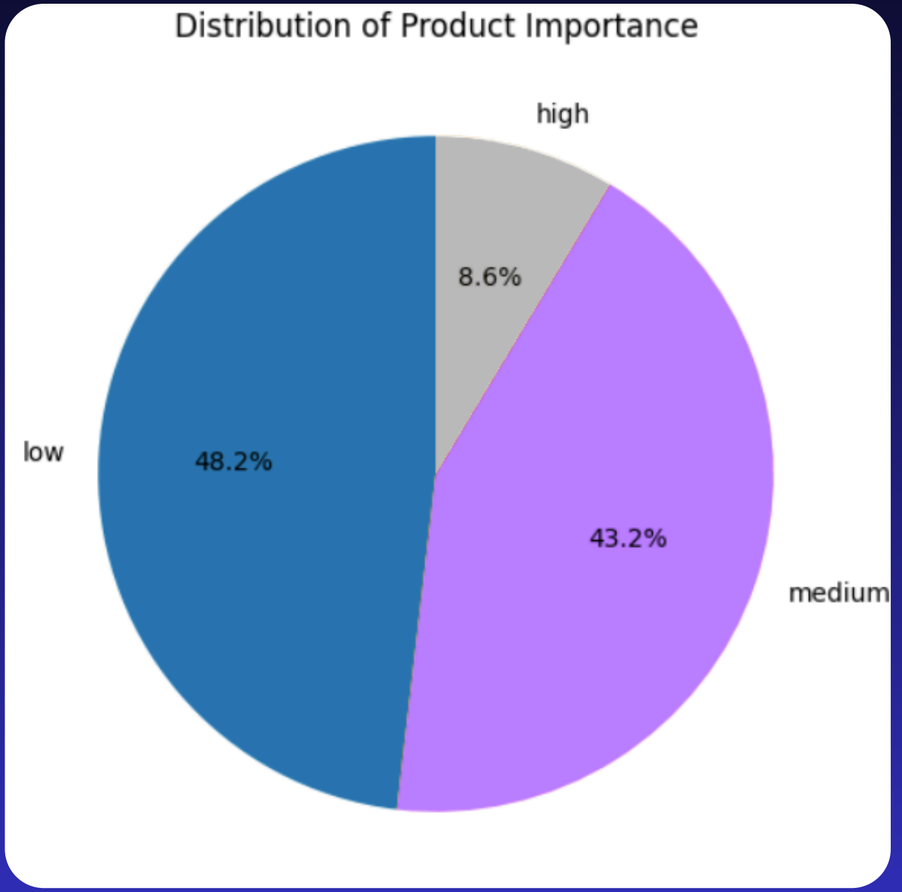



```
sns.histplot(data['Customer_rating'], bins=10 )
plt.title('Distribution of Customer Ratings')
plt.xlabel('Customer Rating')
plt.ylabel('Frequency')
plt.show()
```

Distribution of Product Importance

Pie chart showing how product importance is categorized as low, medium, or high

```
product_counts = data['Product_importance'].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(product_counts, labels=product_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Distribution of Product Importance')
plt.show()
```



Step 7: Data Preprocessing

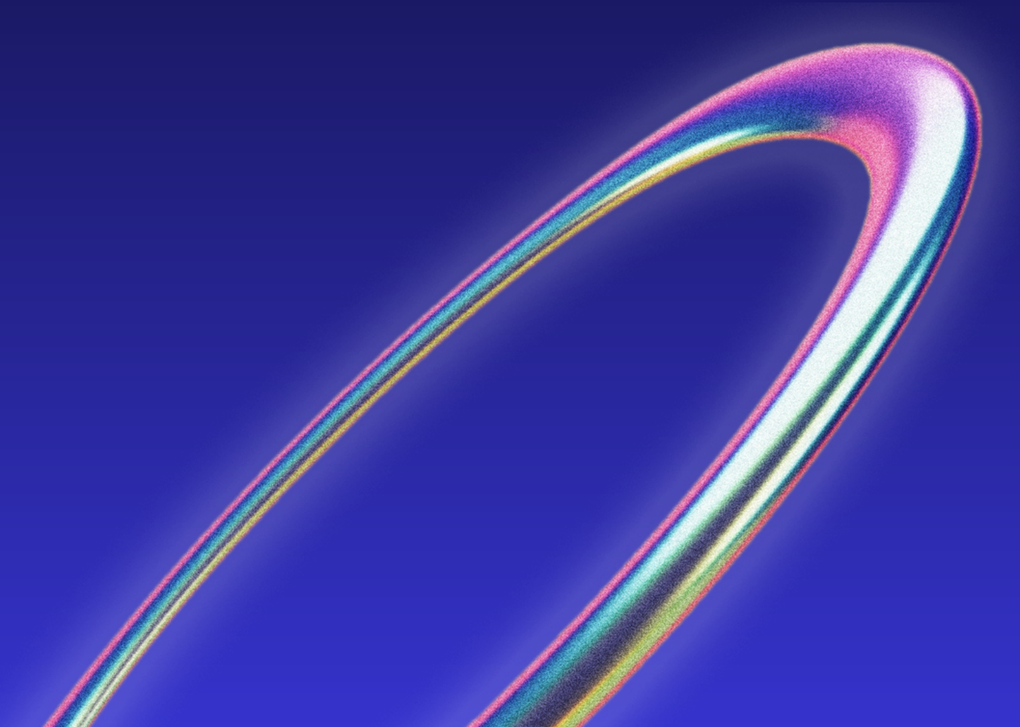
Dropped ID column (not useful for prediction)

```
data.drop('ID', axis=1, inplace=True)
```

Applied One-Hot Encoding using pd.get_dummies()

Set drop_first=True to avoid multicollinearity

```
data = pd.get_dummies(data, drop_first=True)
```

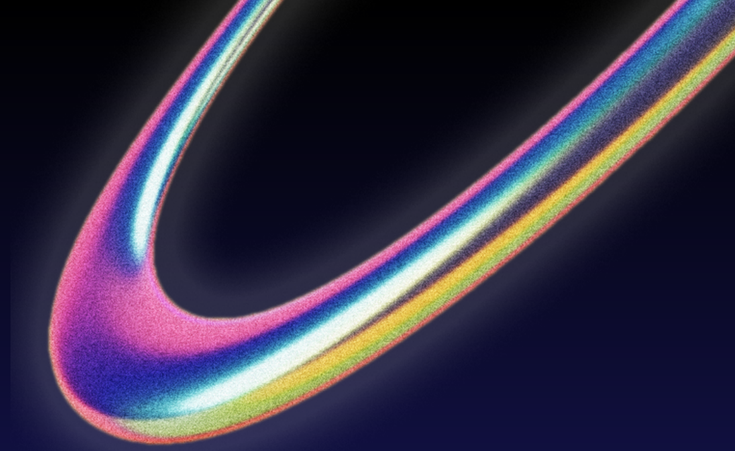


Step 8: Define Features and Target

- Defined features (X) and target (y)
- X = All columns except 'Reached.on.Time_Y.N'
- y = Delivery status (0 or 1)

```
X = data.drop('Reached.on.Time_Y.N', axis=1)
y = data['Reached.on.Time_Y.N']
```

Steps 9–10: Split Data & Train the Model



- Used train_test_split() (70% train, 30% test)

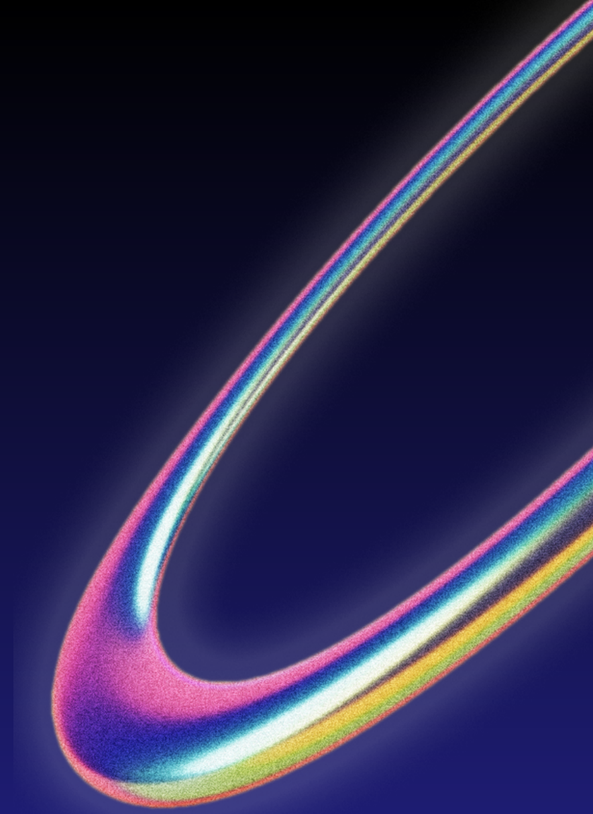
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

- Trained Logistic Regression model on training data

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
```

Step 11: Make Predictions

- Used .predict() to make predictions on test data
- Stored predictions in a variable



```
predictions = model.predict(X_test)
```

step 12 - Model Evaluation

```
# @title Default title text
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, classification_report
print("Accuracy:", round(accuracy_score(y_test, y_pred)*100))
print("Precision:", round(precision_score(y_test, y_pred)*100))
print("Recall:", round(recall_score(y_test, y_pred)*100))
print("F1 Score:", round(f1_score(y_test, y_pred)*100))
```

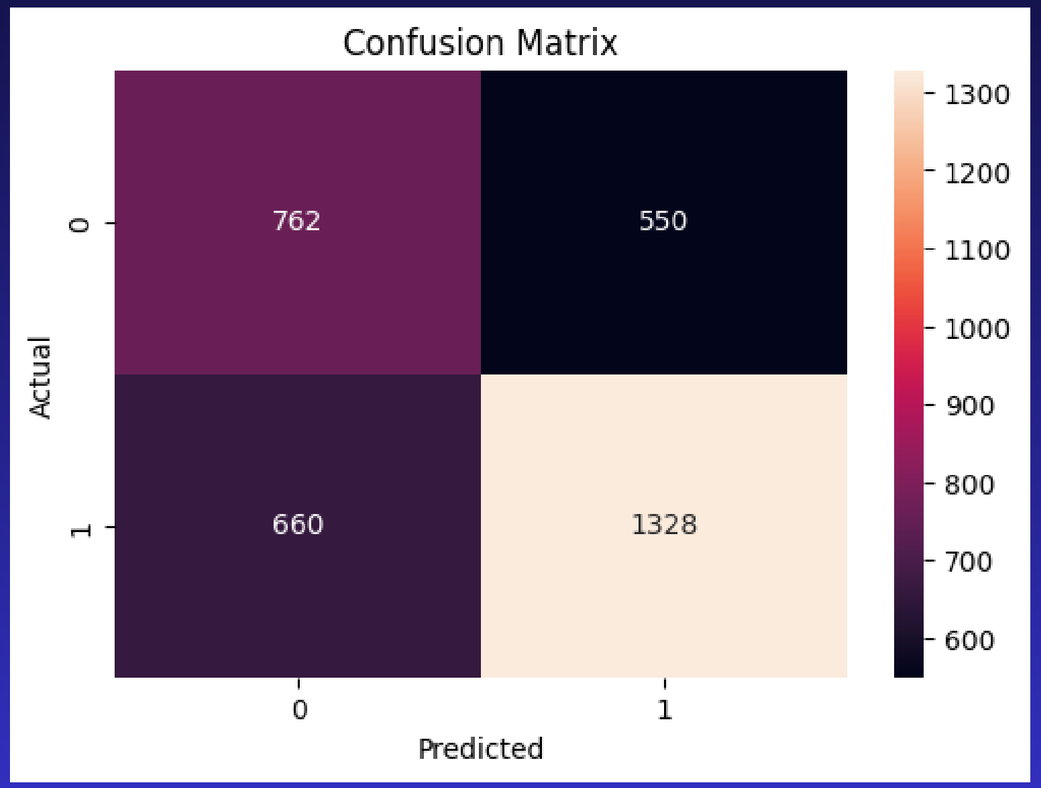
Code Highlights:

- Evaluates model using:
- confusion_matrix()
- accuracy_score()
- classification_report()
- Converts confusion matrix to readable table.
- Prints all results clearly.y Score: measures how many .

Model Performance Summary

- Accuracy: 63%
- Precision: 71%
- Recall: 67%
- F1 Score: 69%

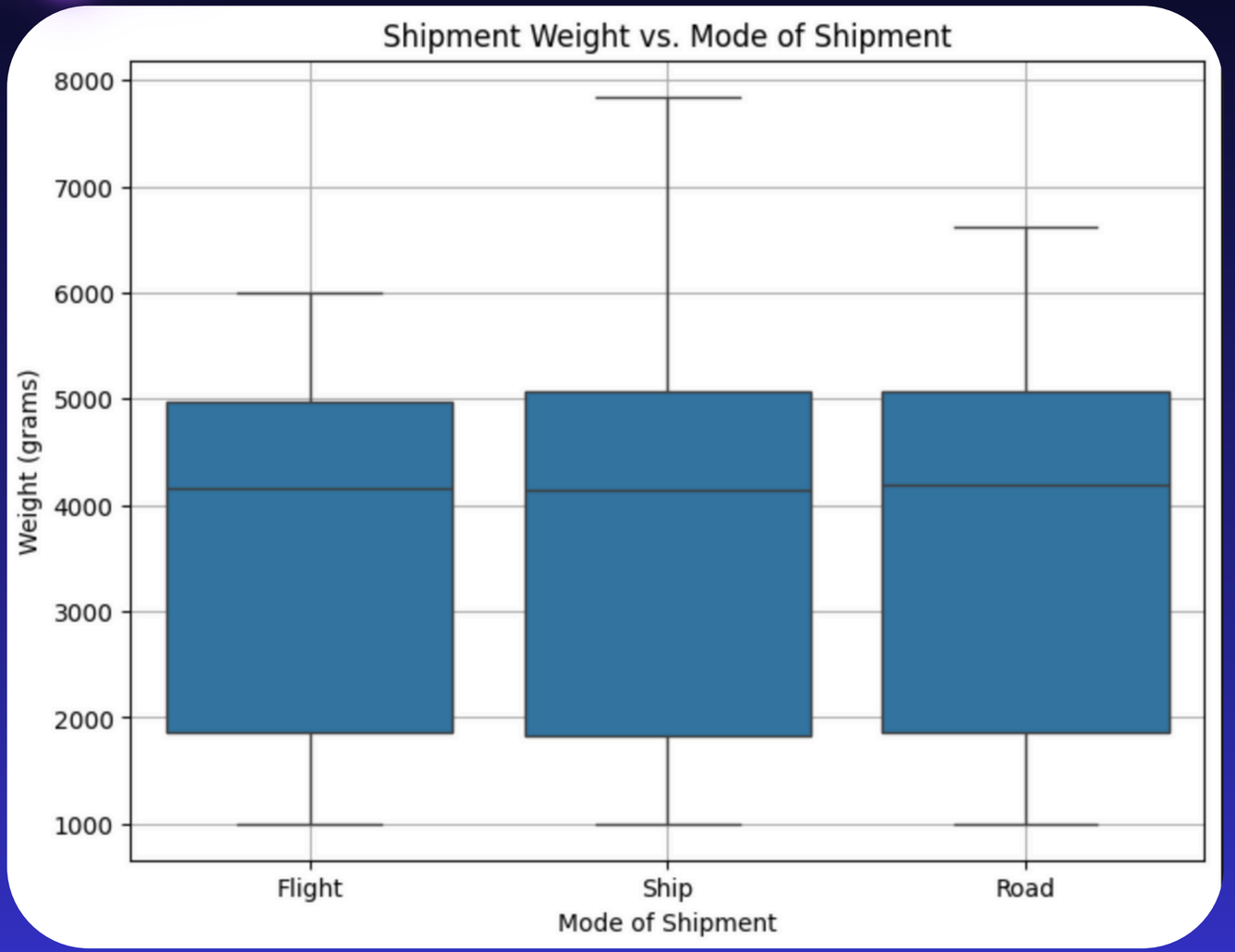
Step 13: Confusion Matrix Heatmap
Visualize the confusion matrix using a heatmap.



- Used seaborn.heatmap() to display predictions clearly.
- X-axis: Predicted | Y-axis: Actual
- Helps visualize correct and incorrect predictions.

Additional Step: Shipment Mode vs. Weight

Box plot showing shipment weight across Flight, Ship, and Road.



Group 4 Members:

- Danah Allehyani
- Khawla Allehyani
- Dana Alamery
- Nourh AL dheam
- Amerah Alhomadhi
- Alaa


THANK
YOU