Adidas Synthetic Sales Analysis

Index

Adidas Synthetic Sales Analysis

1. Project Overview

This project analyses adidas customer shopping behaviour using transactional data from purchases across various product categories. The goal is to uncover insights into spending patterns. customer segments, product preferences and subscriptions behaviour to guide strategic business decisions.

2.Dataset Summary

Link - Adidas Sales 2023 - 2025

This dataset is from Kaggle. It contains adidas sales report from January 2023 – September 2025. This dataset represents **synthetic sales and customer transaction data for Adidas products** across multiple regions and store types.

- Rows 1200
- Columns 15

- **Key features:**

Order ID - Unique identifier for each order
Order date - Data when the order was placed
SKU - Stock Keeping Unit
Product Name - names of adidas products
Category - Product Category
Region - Geographic sales region
Store Type - Types of stores
Units Sold - Number of units sold in transactions
Unit Price - Selling price per unit
Discount - Discount Percentage
Revenue - Total revenue after discount
Profit - Estimated profit margin
Customer Age - Age of the customer
Gender - Gender of the customer
Payment - Payment type used

3. Possible Use Cases

Data Visualization - create charts on product performances, future demands and discount impact
Machine Learning - Trains Model for forecasting, Customer segmentation and future predictions
SQL Practice - Write queries for aggregations, filtering and joins
Power BI - Build interactive reports on sales trends, customer demographics and performance

3.1 Main AIMS

To develop predictive models using lasso regression and Prophet to forecast Adidas sales revenue, identify key drivers of sales and create interactive Power BI dashboards for business insights and filtering and sorting using SQL for Business purposes.

1. **Data Preparation**
   - Clean and preprocess Adidas sales data.
   - Handle missing values, outliers and normalizing data.

2. **Random Forest Model**
   -Visualize sales trends, seasonality and correlations.
   - Identify key variables influencing sales revenue.

3. **Ridge Regression Model**
   -Develop a lasso Regression model to predict Sales revenue.
   -Identify important features contributing to sales.
   -Evaluate model performance (RMSE, MAE).

4. **Prophet Model**
   -Develop a Prophet model to forecast sales revenue.
   -Evaluate model performance and compare with Lasso Regression.

5. **Data Extraction using SQL**
   -Extract Adidas sales data from databases using SQL queries.
   -Filtering Data.

6. **Sales performances Analysis**
   -Calculate total sales revenue, growth and trends over time
   - Identify top-selling Products, regions and sales channels.

7. **Customer Insights**
   -Analyse customer demographics, purchase behaviour and loyalty.
   -Identify high-value customer segments.

8. **Product and Category Analysis**
   -Analyse product sales, profitability, and lifecycle.
   -Identify best-selling products and categories.

9. **Insight Generation**
   -share insights for Informed decision-making.
   -Identify Top-selling products.

10. **BI Dashboards Creation**
    -Create interactive dashboards to visualize sales trends and forecasts.
    -Enable stakeholders to explore data and insights.

3.2 OBJECTIVES

- Data preparation and cleaning.
- Checking Null values and replaces
- Predicting Total Profit using Ridge Regression.
- Predicting Total Units Sold using Random Forest Classifier.
- Predicting Next 12 Months Profit using Prophet.
- Analyse gender-based revenue and calculate average unit price per product category in SQL.
- Identify the highest-profit region and compute total units sold by store type using SQL.
- Summarize revenue & profit by payment method and find the product with the highest total revenue.
- Generate an order list by age group and calculate the average discount per region using SQL.
- Identify top-selling products, regions and effective marketing channels using Power BI

4. Exploratory Data Analysis using Python

We began with data preparation and cleaning in python:

*Data Loading: Imported the dataset Using pandas.

*Initial Exploration: Used df.info () to check structure and. describe () for summary statistics

	Order_ID	Units_Sold	Unit_Price	Discount	Revenue	Profit	Customer_Age
count	1200.000000	1200.000000	1200.000000	1200.000000	1200.000000	1200.000000	1200.000000
mean	100599.500000	2.958333	86.078967	5.841667	239.481600	70.891542	29.640833
std	346.554469	1.314548	59.756635	8.557314	209.691529	62.379234	7.844411
min	100000.000000	1.000000	12.000000	0.000000	12.190000	3.030000	16.000000
25%	100299.750000	2.000000	31.967500	0.000000	77.112500	23.157500	24.000000
50%	100599.500000	3.000000	72.700000	5.000000	169.845000	49.715000	29.000000
75%	100899.250000	4.000000	133.655000	10.000000	352.515000	102.832500	35.000000
max	101199.000000	7.000000	252.890000	40.000000	1526.110000	393.250000	54.000000

*Checking missing Data: Checked for null values

```
<bound method DataFrame.sum of      Order_ID  Order_Date    SKU  Product_Name
0       False       False  False         False     False   False       False
1       False       False  False         False     False   False       False
2       False       False  False         False     False   False       False
3       False       False  False         False     False   False       False
4       False       False  False         False     False   False       False
...       ...         ...    ...           ...       ...     ...         ...
1195    False       False  False         False     False   False       False
1196    False       False  False         False     False   False       False
1197    False       False  False         False     False   False       False
1198    False       False  False         False     False   False       False
1199    False       False  False         False     False   False       False
```

*No missing values detected

*Changing Cases: Transform upper cases to lower cases.

```
df.columns.str.lower()
```

```
Index(['order_id', 'order_date', 'sku', 'product_name', 'category', 'region',
       'store_type', 'units_sold', 'unit_price', 'discount', 'revenue',
       'profit', 'customer_age', 'gender', 'payment_method'],
      dtype='object')
```

4.1 Predicting Profit Using Ridge Regression

*Importing required features for predicting Ridge regression.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge, RidgeCV, LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

*Reading adidas CSV file and assign X and Y variables.

```
numeric_features = df.select_dtypes(include=[np.number])
x = numeric_features.drop('Profit', axis=1)
y = numeric_features['Profit']
```

*Instantiates a Standard Scaler to standardize the features.
*Scales X and splits it into train/test sets (80 % train, 20 % test)
*Random State = 42

```
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size=0.2, random_state=42)
```

*Defines a list of regularization strength alphas = [0.01, 0.1, 0.5, 1, 5]
*Fits Ridge CV with 2-fold cross-validation on the training data.

```
alphas = [0.01, 0.1, 0.5, 1, 5]
ridge_cv = RidgeCV(alphas=alphas, cv=2)
ridge_cv.fit(x_train, y_train)
```

*Finding Best Alpha & Final model

```
best_alpha = ridge_cv.alpha_
print(f"\nBest alpha from CV: {best_alpha}")
ridge = RidgeCV(alphas=alphas, cv=2)
ridge.fit(x_train, y_train)
```

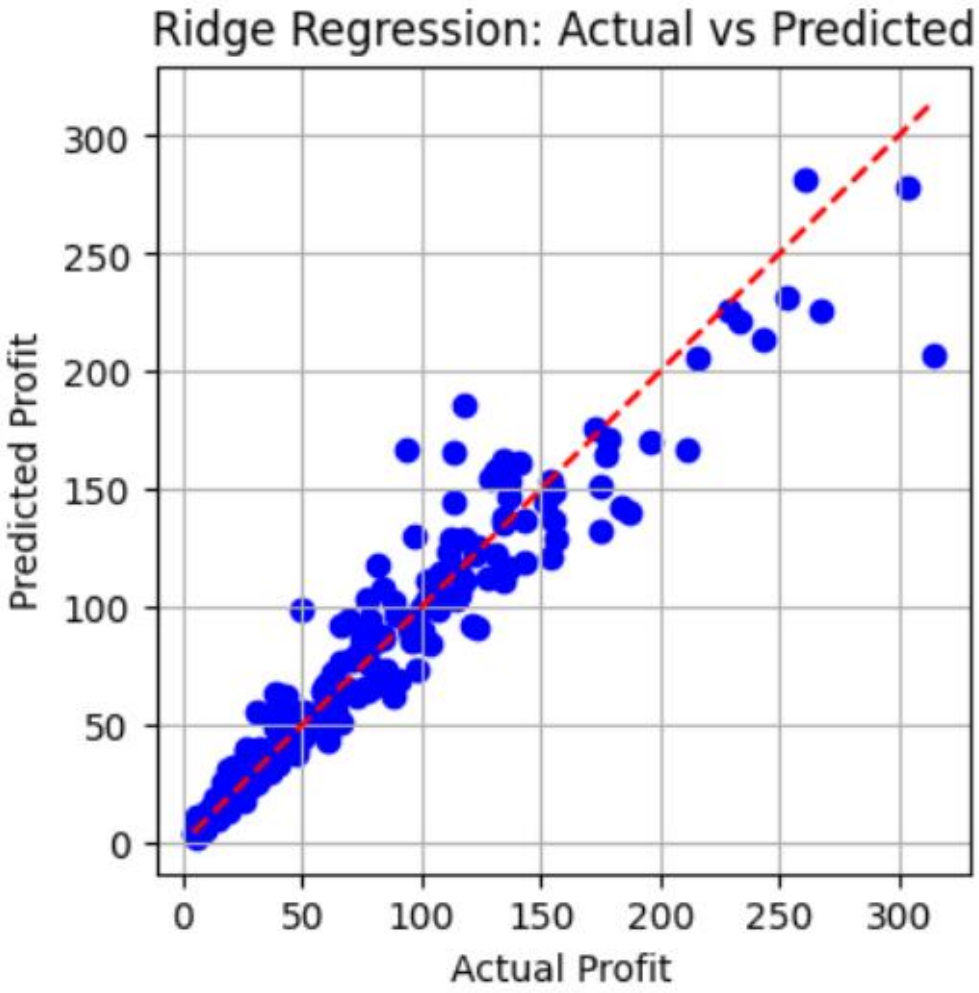
*Predicts on the test set (y pred).
*Computes and prints the Mean Squared Error and R^2 score.
*Prints the regression coefficients.

```
y_pred = ridge.predict(x_test)
print("\nMean Squared Error:", mean_squared_error(y_test, y_pred))
print("R2 Score:", r2_score(y_test, y_pred))
print("Ridge regression coefficients:", ridge.coef_)
```

```
Best alpha from CV: 1.0

Mean Squared Error: 281.5435729771021
R2 Score: 0.922082293957234
Ridge regression coefficients: [-0.50203043  1.45522903  2.74553871 -1.10883287 56.98098213
```

- **We can assume the Best alpha value is 1.0**
- **R^2 (Goodness of fit) is 92.2 %**
- **The model explains 92.2% of the variance in the target.**
- **Mean Squared Error = 281.54**

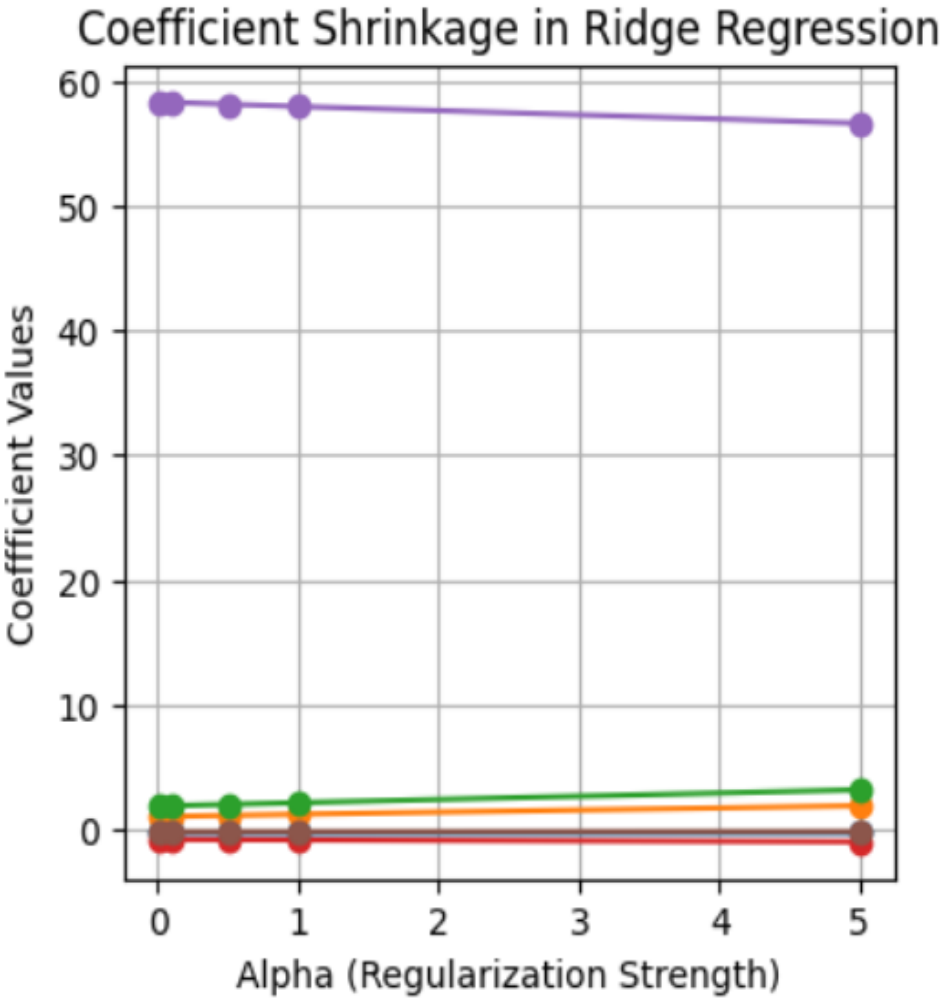
Plotting Actual VS Predicted



- **X-axis represent the true profit numbers**
- **Y-axis represent the predicted profit**
- **Blue dots are data points each pairing an actual profit with corresponding prediction profit**
- **The red dashed line is the perfect-fit line**
- **If dot lies on the line, the prediction matches the actual value exactly**

Visualizing coefficient shrinkage in Ridge

```
coefs = []
for a in alphas:
    ridge_temp = Ridge(alpha=a)
    ridge_temp.fit(x_scaled, y)
    coefs.append(ridge_temp.coef_)
coefs = np.array(coefs)
plt.figure(figsize=(4,4))
plt.plot(alphas, coefs, marker='o')
plt.xlabel('Alpha (Regularization Strength)')
plt.ylabel('Coeffficient Values')
plt.title('Coefficient Shrinkage in Ridge Regression')
plt.grid(True)
plt.show()
```



- **Ridge regression adds a penalty term to loss the function**
- **The shrinking coefficients towards zero to prevent overfitting**
- **The Y-axis (coefficient values) shows the magnitude of the regression for different features**
- **Alpha (regularization Strength) on the X-axis increase from 0 to 5:**
- **The coefficients of most features are slightly reduced but remain relatively stable, indicating moderate shrinkage.**
- **One coefficient (blue line near 60) stays almost unchanged, suggesting the features strong influence persist even with regularization**
- **This plot demonstrate that ridge regression reduces the size of coefficients without setting them exactly to zero.**

4.2 Predicting Next Year Profit Using Prophet Regression

*Importing required features for predicting Prophet.
*Predicting Profit forecast for next 12 months.

```
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt
```

*Reading CSV file and rename Order Date and Profit
*Converts the ds column to datetime format

```
df.rename(columns={"Order_Date": "ds", "Profit": "y"}, inplace=True)
df["ds"] = pd.to_datetime(df["ds"])
```

*Aggregates the data by month (Freq=”ME” means month-end
*sums the profit for each month, creating a monthly time series
*Initialize a Prophet forecasting model
*Fits the model to the monthly aggregated data

```
monthly = df.groupby(pd.Grouper(key="ds", freq="ME")).sum().reset_index()
model = Prophet()
model.fit(monthly)
```

*Create a future data frame for the next 12 months.
*Generates a forecast for those future periods using fitted model.

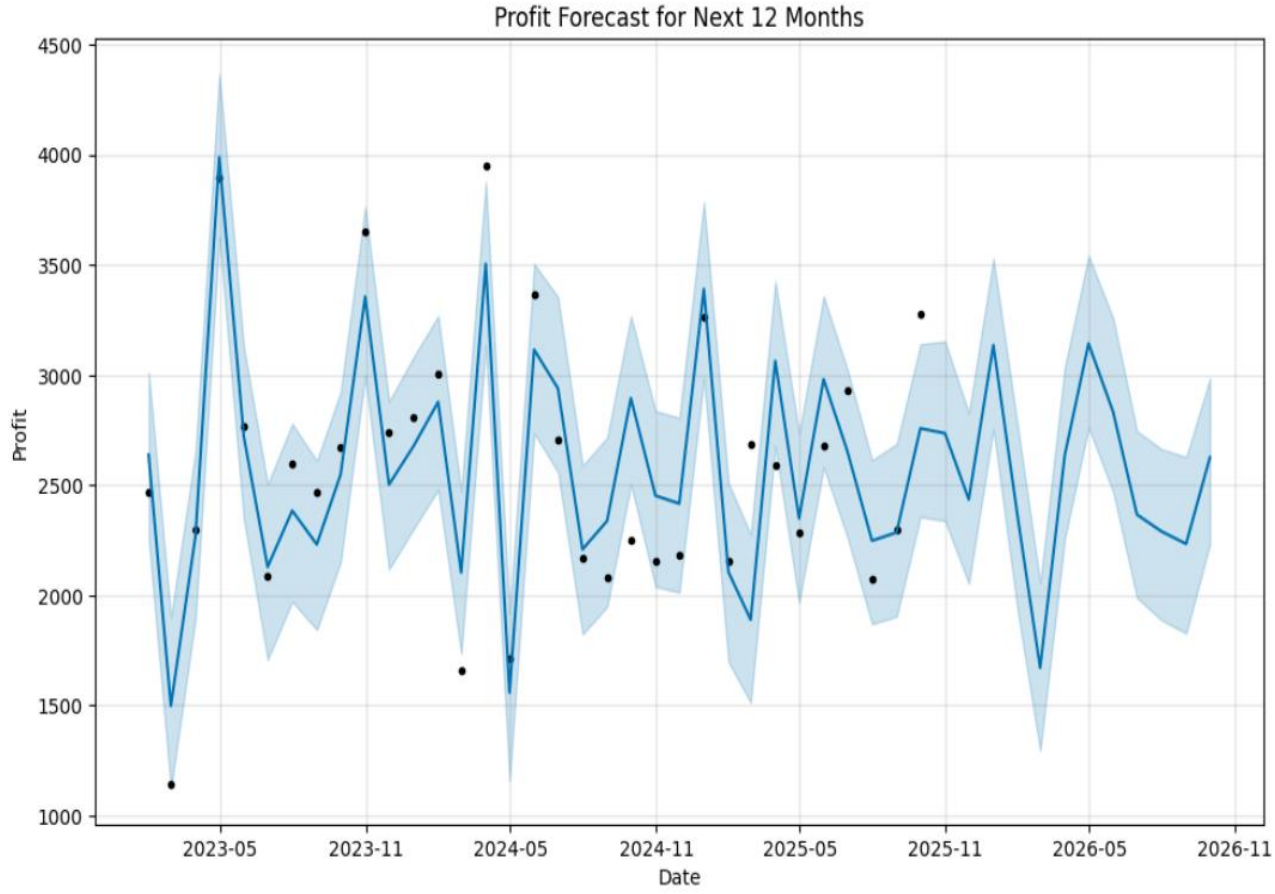
```
future = model.make_future_dataframe(periods=12, freq="ME")
forecast = model.predict(future)
```

*Prints the first few rows of the original Data Frame.
*Print the last 12 rows of the forecast showing predicted Values.
*(Yhat) along with uncertainty intervals (Yhat-lower, Yhat-upper).

```
print("\nForecast Data:")
print(forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(12))
```

```
Forecast Data:
           ds         yhat   yhat_lower   yhat_upper
33 2025-10-31  2735.555615  2338.584125  3153.057449
34 2025-11-30  2435.270954  2053.828685  2827.522389
35 2025-12-31  3135.797765  2754.536178  3529.106189
36 2026-01-31  2347.607459  1920.430270  2759.682051
37 2026-02-28  1671.726458  1295.803729  2056.686553
38 2026-03-31  2635.447213  2257.766265  3036.806979
39 2026-04-30  3143.073053  2762.235815  3541.173059
40 2026-05-31  2832.692398  2469.165055  3258.153574
41 2026-06-30  2366.860613  1989.376993  2746.141329
42 2026-07-31  2290.294148  1887.929702  2664.091005
43 2026-08-31  2233.799856  1827.904019  2628.246139
44 2026-09-30  2626.167897  2233.619119  2985.138784
```

*Plots the forecast results using Prophet's built-in functions.
*Adds labels and title to the plot, then displays it.

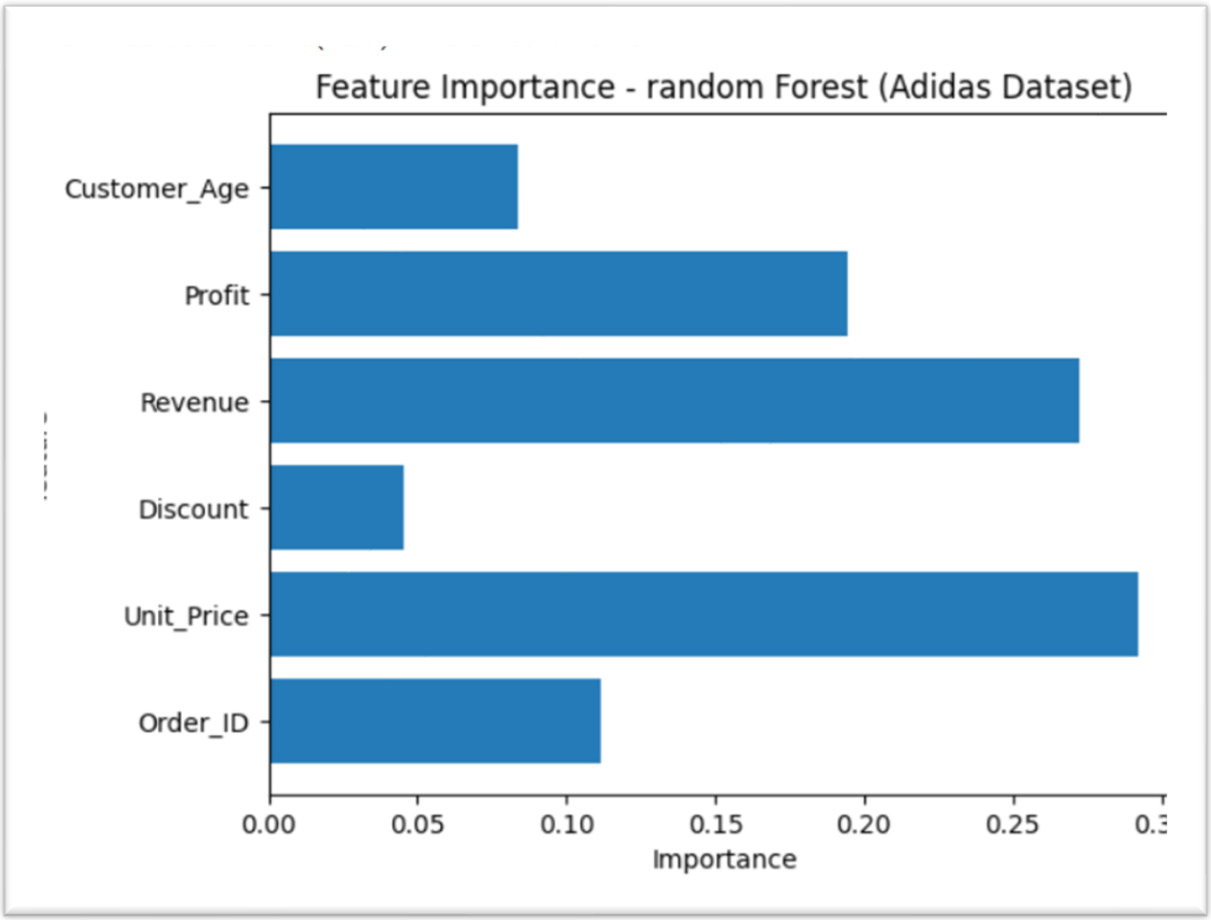


- Blue line – It's the predicted profit trend over time. Wiggles up and down showing expected profit.
- Shaded area – That's the confidence interval around the forecast. Wilder shading means the model is less sure about the periods profit.
- Black dots – These represent the actual observed profit data points used to fit the forecast.
- X-axis (Date)– The timeline from mid-2023 to late 2026, split into monthly ticks.
- Y-axis (Profit) – The profit values raging from about 1000 to 4500 units.
- Overall, the model expects profit to stay in 2000-3000 range with occasional spikes.

4.3 Predicting Total Units Sold Using Random Forest Classifier

```
Accuracy: 0.8166666666666667

Confusion Matrix:
 [[48  0  0  0  0  0  0]
 [ 3 93  4  0  0  0  0]
 [ 0  8 87  5  0  0  0]
 [ 0  1 14 48  5  0  0]
 [ 0  0  0 12 16  0  0]
 [ 0  0  0  2  7  2  0]
 [ 0  0  0  4  1  0  0]]

classification Report:
              precision    recall  f1-score   support

           1       0.94      1.00      0.97        48
           2       0.91      0.93      0.92       100
           3       0.83      0.87      0.85       100
           4       0.68      0.71      0.69        68
           5       0.55      0.57      0.56        28
           6       1.00      0.18      0.31        11
           7       0.00      0.00      0.00         5

    accuracy                           0.82       360
   macro avg       0.70      0.61      0.61       360
weighted avg       0.81      0.82      0.80       360

OOB Score: 0.7523809523809524
```

- Accuracy is 0.81 % this is the overall predictions out of all predictions made by the model.
- Out Of Bag score indicating the model's performance on bootstrap samples.
- Marco Avg: averages metrics across classes without weighting.
- Weighted Avg: class-size weighted average.
- Class 7: Zero precision, recall & F1-acoew; the model
- fails completely on this class.



- The plot shows feature importance from a random forest model on the adidas dataset.
- Unit price has the highest importance, meaning it is the strongest predictor among the features.
- Discount has lower importance implying its less influential than revenue or discount.

5.Data Analysis using SQL

```
import pandas as pd
from sqlalchemy import create_engine
username = "postgres"
password = "appuappu"
host = "localhost"
port = "5432"
database = "adidas"

engine = create_engine(f"postgresql+psycopg2://{username}:{password}@{host}:{port}/{database}")
table_name = "adidas"
df = pd.read_csv(r"C:\Users\appuv\Desktop\b\adidas_dataset_1200rows.csv")
df.to_sql(table_name, engine, if_exists="replace", index=False)

print(f"Data sucessfully loaded into table '{table_name}' in database '{database}'.")
```

```
Data sucessfully loaded into table 'adidas' in database 'adidas'.
```

We performed structured analysis in PostgresSQL to answer key business questions.

```sql
TRUNCATE TABLE adidas RESTART IDENTITY;
DROP TABLE IF EXISTS adidas
CREATE TABLE adidas (
      Order_ID SERIAL PRIMARY KEY,
      Order_Date VARCHAR(100),
      SKU VARCHAR(100),
      Product_Name VARCHAR(100),
      Category VARCHAR(100),
      Region VARCHAR(100),
      Store_Type VARCHAR(100),
      Units_Sold INT,
      Unit_Price VARCHAR(100),
      Discount INT,
      Revenue DECIMAL(10,2),
      Profit DECIMAL(10,2),
      Customer_Age INT,
      Gender VARCHAR(100),
      Payment_Method VARCHAR(100)
);

SELECT*FROM adidas
```

- Create a database called ‘Adidas’ in PostSQL and Import file.
- Truncate removes all rows from adidas table.
- Restart Identity resets any auto-incrementing sequences back.
- Deletes the table if already exists in database.
- Creates a new table named adidas with specified columns.
- VARCHAR is used for storing strings with a maximum length defined, (e.g. Varchar (100) allows up to 100 characters).
- Queries all rows and columns (*) from the newly created adidas table (likely to verify the set up or inspect data).
- In short, the script resets the adidas table, rebuilds it with given schema, and then selects everything from it for inspection.

1.**Revenue by Gender** – compared total revenue generated by male vs female vs others

```
select gender, SUM(revenue) AS total_revenue
from adidas
group by gender;
```

	gender character varying (100)	total_revenue numeric
1	Other	4048.91
2	Male	158775.67
3	Female	124553.34

2.**Average units per category** – average unit price for each category

```
select category, AVG(unit_price::numeric) AS avg_unit_price
from adidas
group by category;
```

	category character varying (100)	avg_unit_price numeric
1	Accessories	32.0019827586206897
2	Footwear	128.6223306772908367
3	Apparel	78.8276000000000000

3.Highest Profit by Region - Which region generates highest Profit.

```
select store_type, SUM(units_sold) AS total_units_sold
from adidas
group by store_type;
```

	region character varying (100)	total_profit numeric
1	Asia-Pacific	25519.31

4.Total Units Sold for each Store Type

```
select store_type, SUM(units_sold) AS total_units_sold
from adidas
group by store_type;
```

	store_type character varying (100)	total_units_sold bigint
1	Wholesale	259
2	Retail	1263
3	Outlet	427
4	Online	1601

5.Total Revenue By each Payment Method

```
select payment_method, SUM(revenue) AS total_revenue
from adidas
group by payment_method;
```

	payment_method character varying (100)	total_revenue numeric
1	UPI	35369.94
2	Google Pay	28480.27
3	Apple Pay	38598.59
4	PayPal	42348.68
5	Debit Card	38181.17
6	NetBanking	39117.77
7	Credit Card	32106.24
8	Cash	33175.26

6.Product With Highest Total Revenue

```
select product_name, SUM(revenue) AS total_revenue
from adidas
group by product_name
order by total_revenue desc
limit 1;
```

	product_name character varying (100)	total_revenue numeric
1	Predator Freak	31599.08

7.List of Orders placed by Age Group

```
select customer_age, COUNT(order_id) AS order_count
from adidas
group by customer_age
limit 5;
```

	customer_age integer	order_count bigint
1	42	22
2	54	1
3	29	56
4	34	55
5	41	22

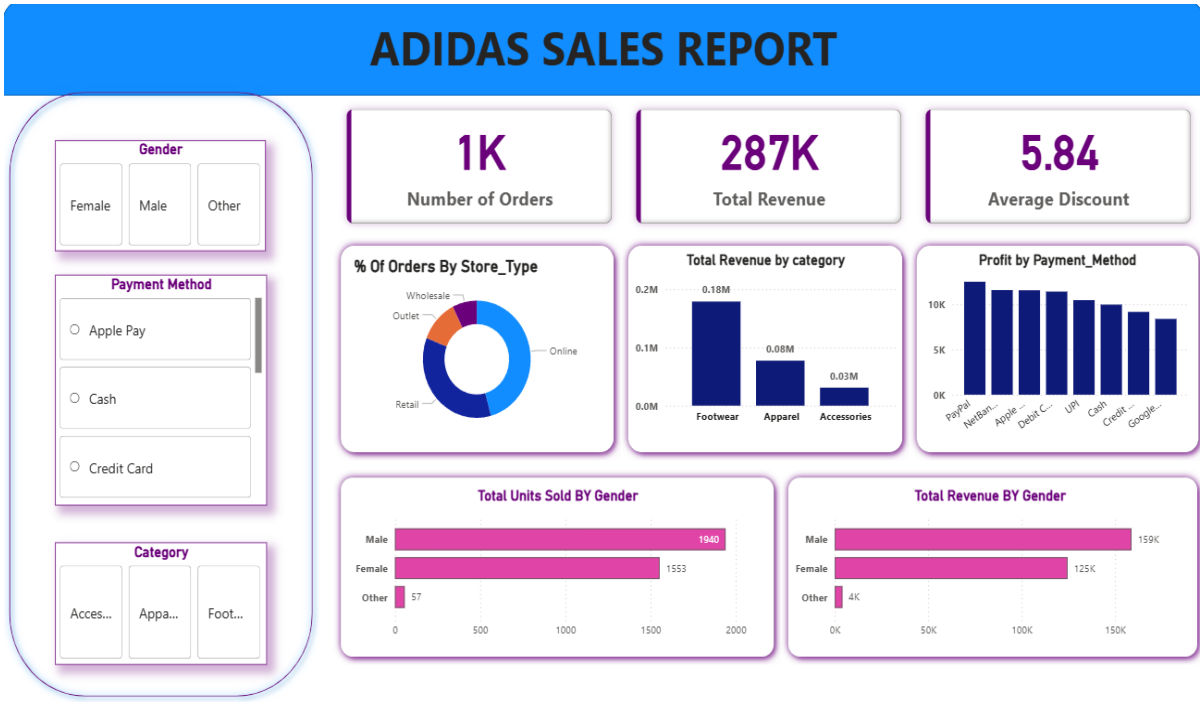
8. Average Discount Given Per Region

```
select region, AVG(discount) AS avg_discount
from adidas
group by region;
```

	region character varying (100)	avg_discount numeric
1	Middle East & Africa	5.7407407407407407
2	Latin America	6.3008130081300813
3	North America	6.0512129380053908
4	Europe	5.3287197231833910
5	Asia-Pacific	5.9077380952380952

6. Data Presentation Using Power BI

- We performed structured analysis using POWER BI to answer key business questions.
- Importing data from Postgres SQL
- Creating a new measure by finding the discount of average.



- 1K orders were placed, generating 287K in total revenue with an average discount of 5.84.
- Most sales come from Retail stores, followed by Online and Wholesale Outlet.
- This suggests retail locations drive the bulk of order value.
- Footwear brings in the highest revenue, followed by Apparel and then Accessories.
- Focus on footwear could boost overall sales.
- Apple Pay shows the highest profit among payment options, while methods like UPI, Cash and google Pay have lower profit bars.
- Promotion of high-profit payment methods may improve margins.

7. CONCLUSION

The dataset captures 1K orders that generated 287K In total revenue with an average discount of 5.84%. This indicates a healthy sales volume with effective promotional pricing.

- The Ridge regression predict profit accuracy of 92.2%.
- Random Forest classifier gives an accuracy for Total Units sold of 81%.
- Revenue is heavily driven by Footwear (0.18M), followed by apparel and accessories.
- Inventory and promotions should prioritize footwear to maximum sales.
- After predicting the next 12 months profit using Prophet regression, Profit decreasing and increasing with average value.

8. Business Recommendations

- **Boost Subscriptions** – Promote exclusive benefits for subscribers.
- **Customer Loyalty Program** – reward repeat buyers to more them in to "Loyal segments".
- **Review Discount Policy** – Balance sales boost with margin control.
- **Product Positioning** – Highlight top-rated and best-selling products in campaigns.
- **Targeted Marketing** – Focus efforts on high-revenue age groups and express-shipping users.