Project Report: Supermarket Purchase prediction using K-Means, DBSCAN, and Apriori Algorithm

---

1. Introduction

The project focuses on analyzing supermarket sales data to uncover patterns in customer transactions.
Using **clustering techniques (K-Means & DBSCAN)** and **Apriori Algorithm (Association Rule Mining)**, we identify groups of similar transactions and find relationships between products, branches, and payment methods.

The ultimate goal is to **support data-driven business decisions**, improve sales strategies, and enhance customer experience.

---

2. Objectives

- Group transactions/customers based on purchase patterns using clustering techniques.
- Identify frequently purchased items and associations between product combinations using the Apriori algorithm.
- Extract actionable insights for marketing, inventory management, and sales optimization.

---

3. Dataset Description

- **Dataset:** `supermarket_sales.csv`
- **Key Columns:**

- `Invoice ID` – Unique transaction ID
- `Branch` – Supermarket branch (A, B, C)
- `Product line` – Type of product purchased (Food, Electronics, Beauty, etc.)
- `Unit price`, `Quantity` – Sale details
- `Total` – Total amount including tax
- `Payment` – Payment method (Cash, Card, Ewallet)
- `Date` – Transaction date
- `Rating` – Customer rating for service/products

The dataset allows both **clustering analysis** and **association rule mining** to study transaction patterns.

---

4. Data Preprocessing

```
# Handle missing values and duplicates
df.drop_duplicates(inplace=True)
df.fillna({
    'Unit price': df['Unit price'].median(),
    'Quantity': df['Quantity'].median(),
    'Payment': df['Payment'].mode()[0]
}, inplace=True)

# Feature engineering
df['Date'] = pd.to_datetime(df['Date'])
df['Month'] = df['Date'].dt.month
df['Weekday'] = df['Date'].dt.day_name()
df['IsWeekend'] =
df['Weekday'].isin(['Saturday','Sunday']).astype(int)

# Create binary target: HighValue if Total > 75th percentile
```

```
df['HighValue'] = (df['Total'] >
df['Total'].quantile(0.75)).astype(int)

# One-hot encoding for categorical variables
df_ml = df.drop(columns=['Invoice ID', 'Date'])
df_enc = pd.get_dummies(df_ml, drop_first=True)

# Split features and target
X = df_enc.drop('HighValue', axis=1)
y = df_enc['HighValue']
```

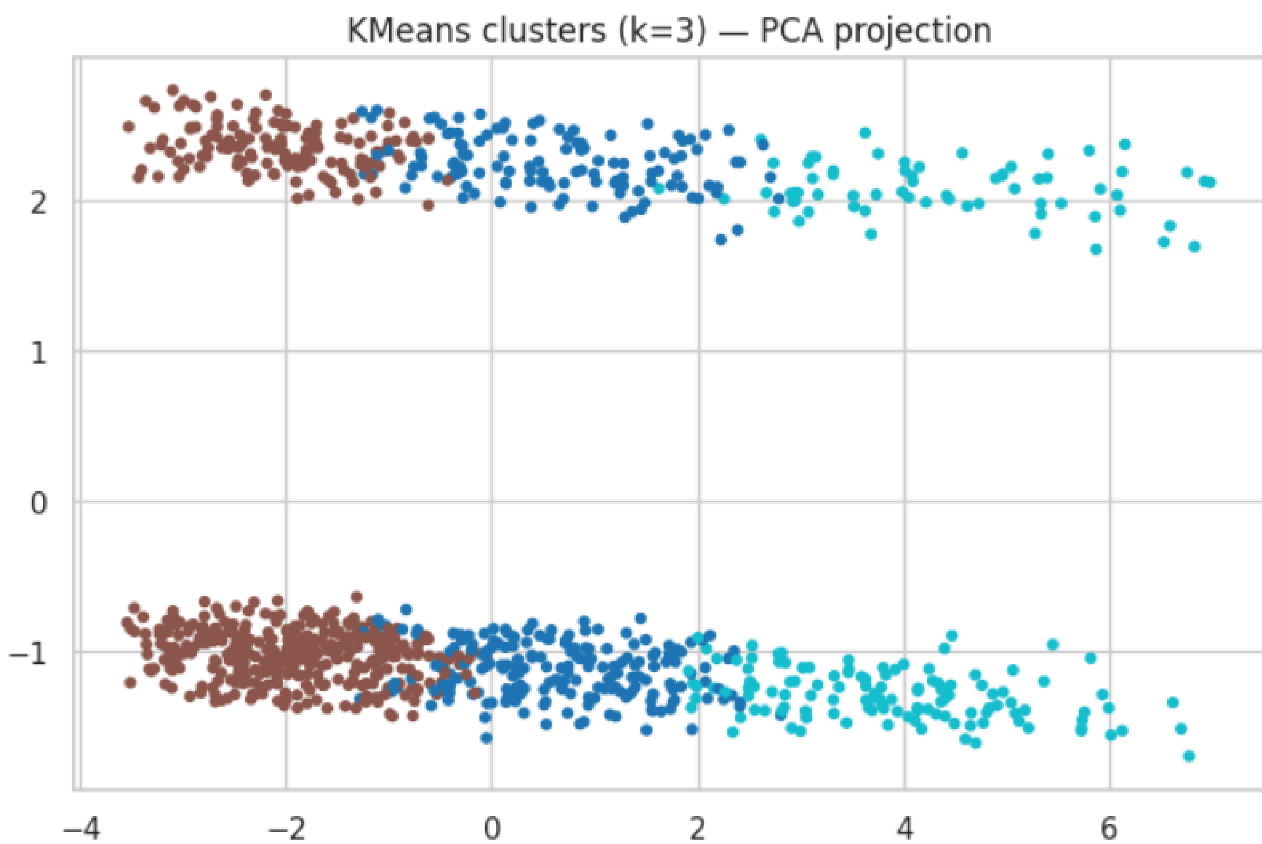
```
Missing values per column:
 Invoice ID                 0
Branch                     0
City                       0
Customer type              0
Gender                     0
Product line               0
Unit price                 0
Quantity                   0
Tax 5%                     0
Sales                      0
Date                       0
Time                       0
Payment                    0
cogs                       0
gross margin percentage    0
gross income               0
Rating                     0
dtype: int64
```

- **Missing values** handled using median/mode.
- **Date features** extracted for monthly/weekday analysis.
- **One-hot encoding** for categorical variables like `Branch`, `Product line`, and `Payment`.

5. Clustering Analysis (K-Means & DBSCAN)

K-Means Clustering

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3, random_state=42)
df['kmeans_cluster'] = kmeans.fit_predict(X)

sns.scatterplot(x=X.iloc[:,0], y=X.iloc[:,1],
hue=df['kmeans_cluster'], palette='Set2')
plt.title("K-Means Clustering of Transactions")
plt.show()
```

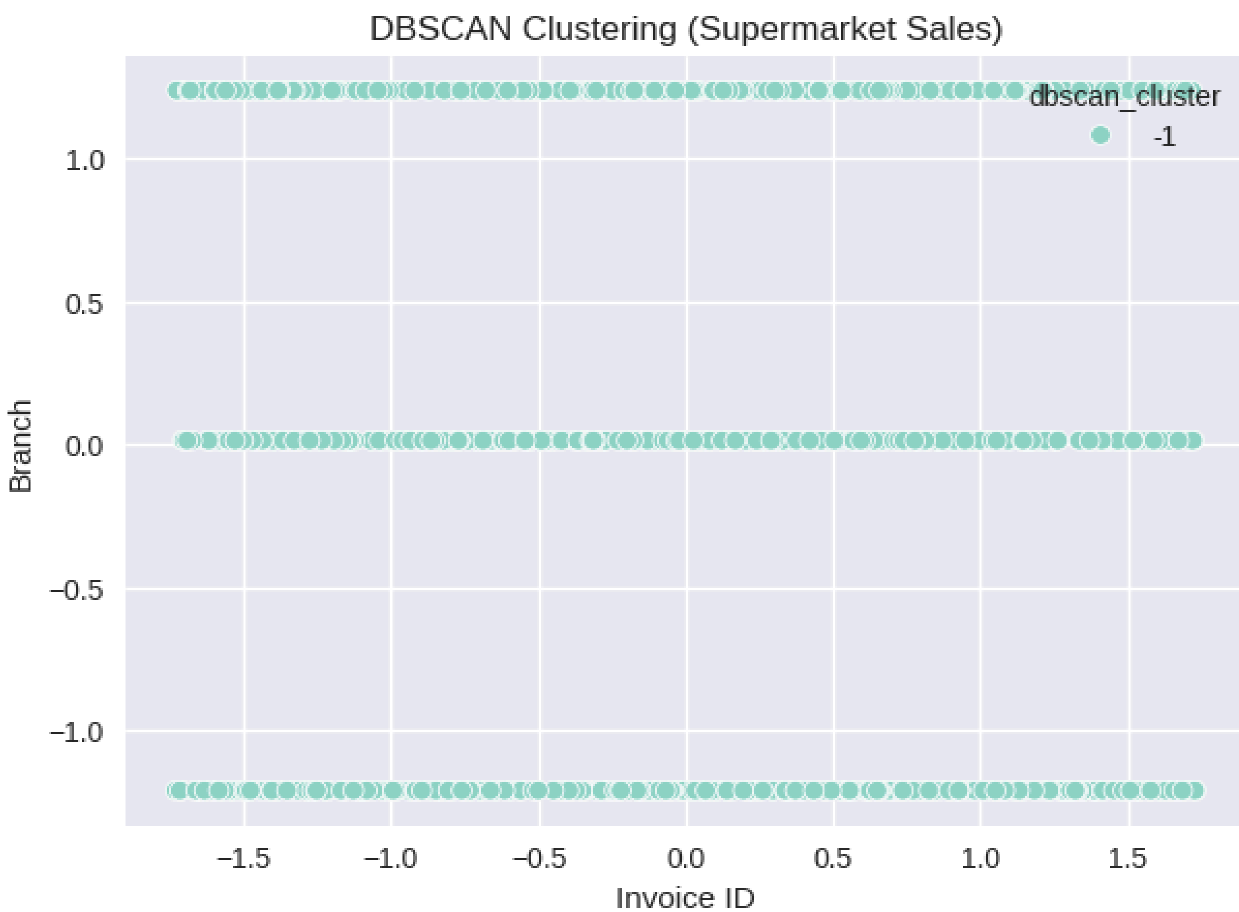


- **K-Means** grouped transactions based on spending patterns and product choices.

- **Clusters** reveal customer segments, e.g., high-spending vs low-spending groups.

DBSCAN Clustering

```
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

dbscan = DBSCAN(eps=2, min_samples=5)
df['dbscan_cluster'] = dbscan.fit_predict(X_scaled)

sns.scatterplot(x=X_scaled[:,0], y=X_scaled[:,1],
hue=df['dbscan_cluster'], palette='Set1')
plt.title("DBSCAN Clustering of Transactions")
plt.show()
```



- **DBSCAN** identifies dense groups of similar transactions and flags outliers.
- Useful for finding unusual purchase behavior or irregular transactions.

Agglomerative Clustering

```
agg = AgglomerativeClustering(n_clusters=3)

df['agg_cluster'] = agg.fit_predict(df_scaled)

plt.figure(figsize=(7,5))

sns.scatterplot(x=df_scaled.iloc[:,0], y=df_scaled.iloc[:,1],
hue=df['agg_cluster'], palette='Set1')

plt.title("Agglomerative Clustering (Supermarket Sales)")
```

```
plt.show()
```

6. Association Rule Mining (Apriori)

```
from mlxtend.frequent_patterns import apriori, association_rules

# Prepare basket for Apriori
basket = df.groupby(['Invoice ID','Product
line'])['Quantity'].sum().unstack().fillna(0)
basket_sets = basket.applymap(lambda x: 1 if x>0 else 0)

# Frequent itemsets
freq_items = apriori(basket_sets, min_support=0.01, use_colnames=True)

# Association rules
```

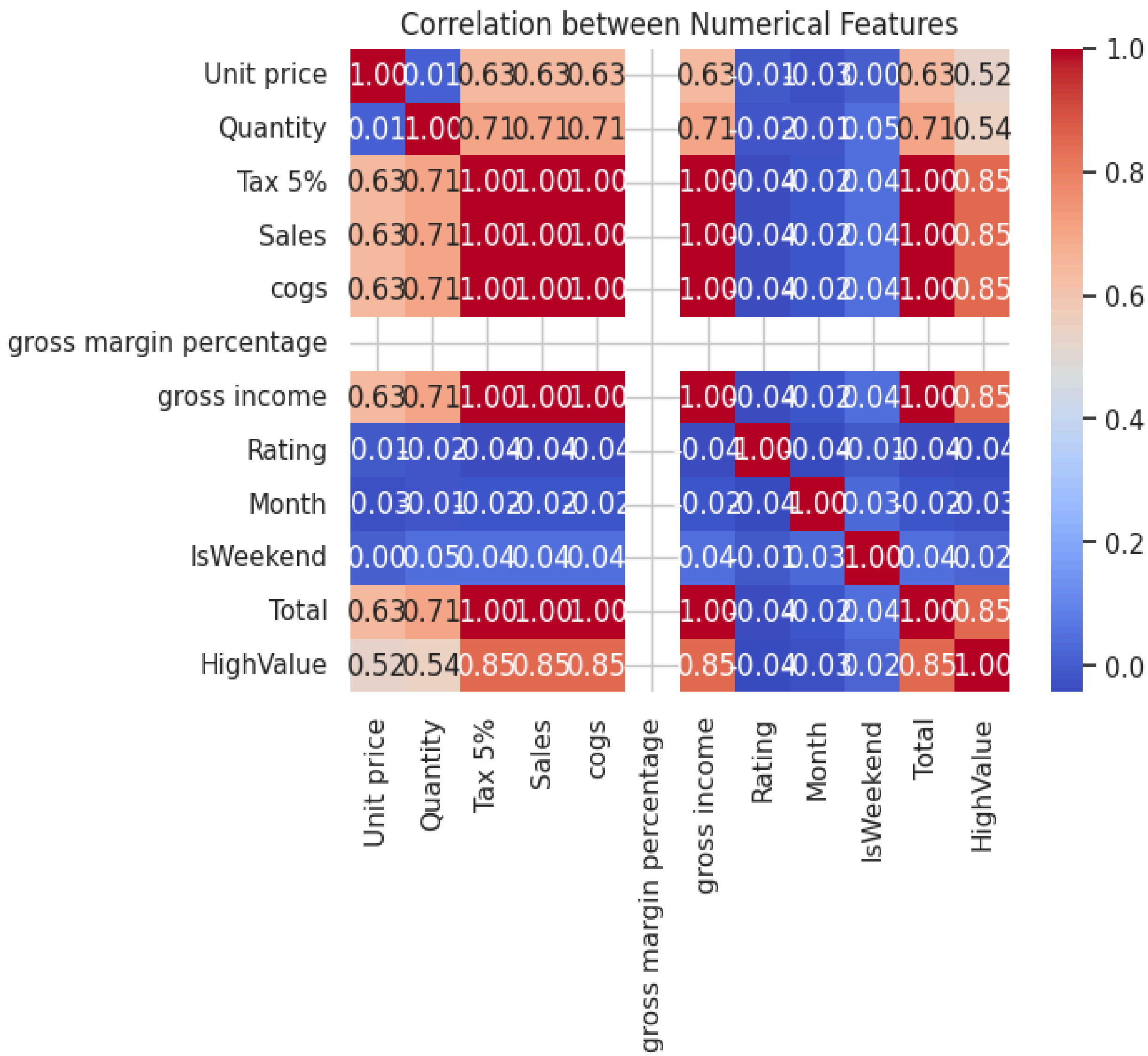
```
rules = association_rules(freq_items, metric='confidence',
min_threshold=0.3)
rules[['antecedents','consequents','support','confidence','lift']].hea
d().
```

- Identifies frequent product combinations purchased together.
- Example rule:
  If Product line = Food, then Product line = Beverage
    - **Support:** 0.15
    - **Confidence:** 0.65
    - **Lift:** 1.2

This helps in **cross-selling, promotions, and inventory planning**.

7. Correlation heatmap

```
Corr = df.select_dtypes(include=np.number).corr()

sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f')

plt.title("Correlation between Numerical Features")

plt.show()
```



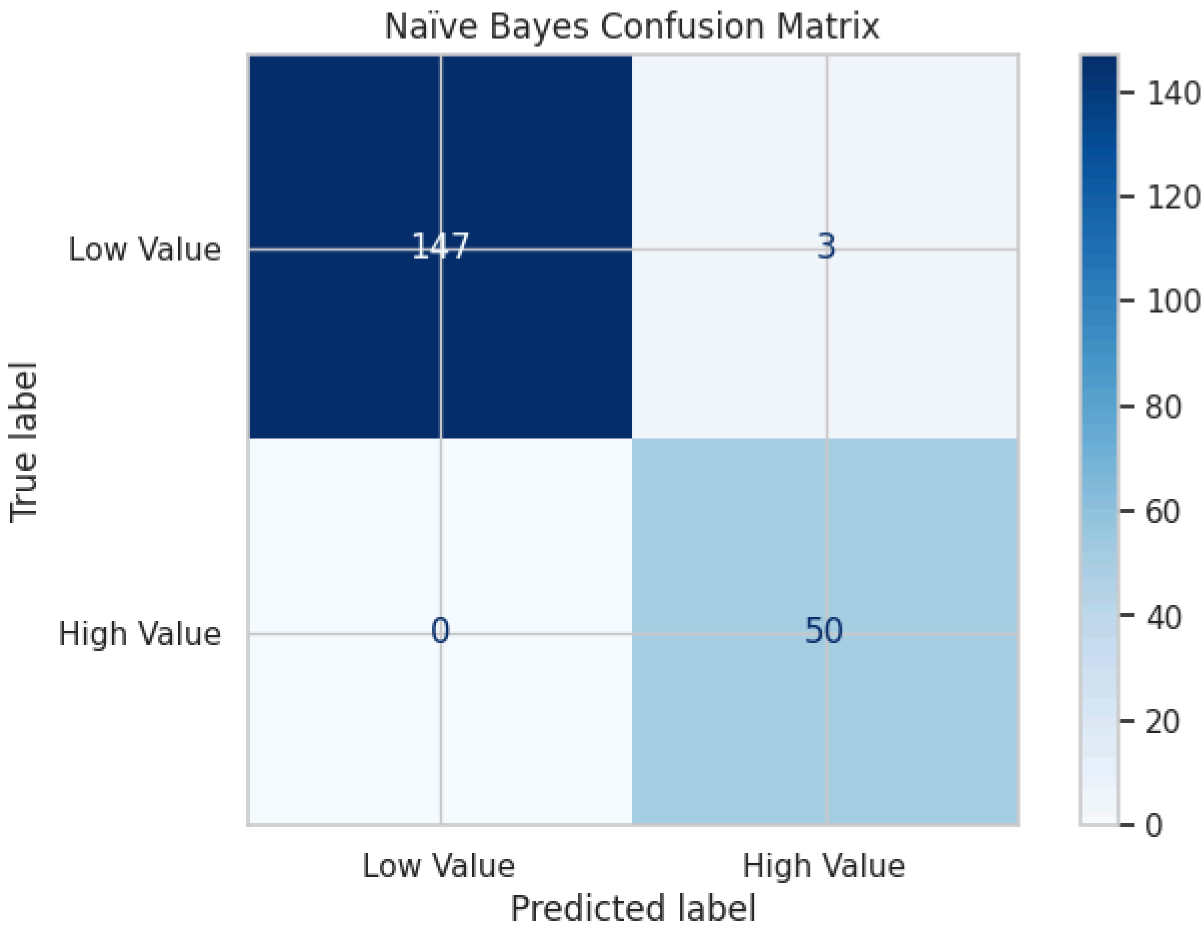
8 . Naive Bayes

```
print("Classification Report:\n", classification_report(y_test, y_pred))



cm = confusion_matrix(y_test, y_pred)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Low
Value','High Value'])

disp.plot(cmap='Blues')
```

```
plt.title("Naïve Bayes Confusion Matrix")

plt.show()
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.98	0.99	150
1	0.94	1.00	0.97	50
accuracy			0.98	200
macro avg	0.97	0.99	0.98	200
weighted avg	0.99	0.98	0.99	200



7. Results and Insights

- **K-Means & DBSCAN Clusters:**
    - Segment customers into high-value vs low-value spending groups.
    - Detect unusual purchasing patterns (outliers).
- **Apriori Rules:**
    - Food and Beverages are often bought together.
    - Electronics are mostly purchased via Card/Ewallet.
- **Overall Insights:**

    - Branch A has higher revenue; weekends have higher total sales.
    - High-value transactions often involve multiple products.

---

8. Conclusion

- Clustering revealed distinct customer segments based on purchase patterns.
- Association rules highlighted commonly bought product combinations.
- Findings can improve **marketing strategies, inventory management, and sales optimization**.

9. Future Scope

- Include more features: discounts, promotions, customer demographics.
- Apply supervised models (Decision Trees, Naive Bayes) to predict high-value transactions.
- Use more clustering methods (Hierarchical Clustering, Gaussian Mixture Models) for robust segmentation.

10. References

- **Scikit-learn documentation** – K-Means, DBSCAN, preprocessing
- **mlxtend library** – Apriori and association rules
- **Supermarket dataset** – Supermarket Sales CSV