MoveInSync - ML Project

IIIT Bangalore

Jahnavi Challa - IMT2020103

Contents

About the Task

A company plans to expand into new markets with five existing products. Based on market research, the new market's behavior is expected to resemble their current market.

In their current market, customers are divided into four segments (A, B, C, D), and targeted outreach for each segment has proven effective. The company now wants to apply the same segmentation strategy to 2,627 potential customers in the new market.

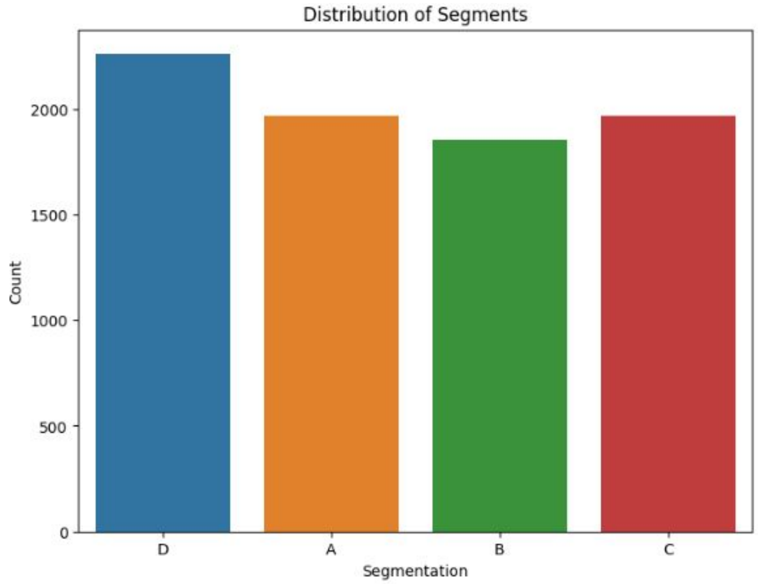
Your task is to assist in predicting the appropriate segment for each new customer.

Data Exploration

	ID	Gender	Ever_Married	Age	Graduated	Profession	Work_Experience	Spending_Score	Family_Size	Var_1	Segmentation
0	462809	Male	No	22	No	Healthcare	1.0	Low	4.0	Cat_4	D
1	462643	Female	Yes	38	Yes	Engineer	NaN	Average	3.0	Cat_4	A
2	466315	Female	Yes	67	Yes	Engineer	1.0	Low	1.0	Cat_6	B
3	461735	Male	Yes	67	Yes	Lawyer	0.0	High	2.0	Cat_6	B
4	462669	Female	Yes	40	Yes	Entertainment	NaN	High	6.0	Cat_6	A
5	461319	Male	Yes	56	No	Artist	0.0	Average	2.0	Cat_6	C
6	460156	Male	No	32	Yes	Healthcare	1.0	Low	3.0	Cat_6	C
7	464347	Female	No	33	Yes	Healthcare	1.0	Low	3.0	Cat_6	D
8	465015	Female	Yes	61	Yes	Engineer	0.0	Low	3.0	Cat_7	D
9	465176	Female	Yes	55	Yes	Artist	1.0	Average	4.0	Cat_6	C

Variable	Definition
ID	Unique ID
Gender	Gender of the customer
Ever_Married	Marital status of the customer
Age	Age of the customer
Graduated	Is the customer a graduate?
Profession	Profession of the customer
Work_Experience	Work Experience in years
Spending_Score	Spending score of the customer
Family_Size	Number of family members for the customer(including the customer)
Var_1	Anonymised Category for the customer
Segmentation (target)	Customer Segment of the customer

About the Dataset



The distribution is uniform enough to ensure models generalize well across all segments.

Data Pre-processing

1. Handling Null Values.
2. Converting categorical labels using one hot encoding (or) label encoding.
3. Scaling the numerical features.

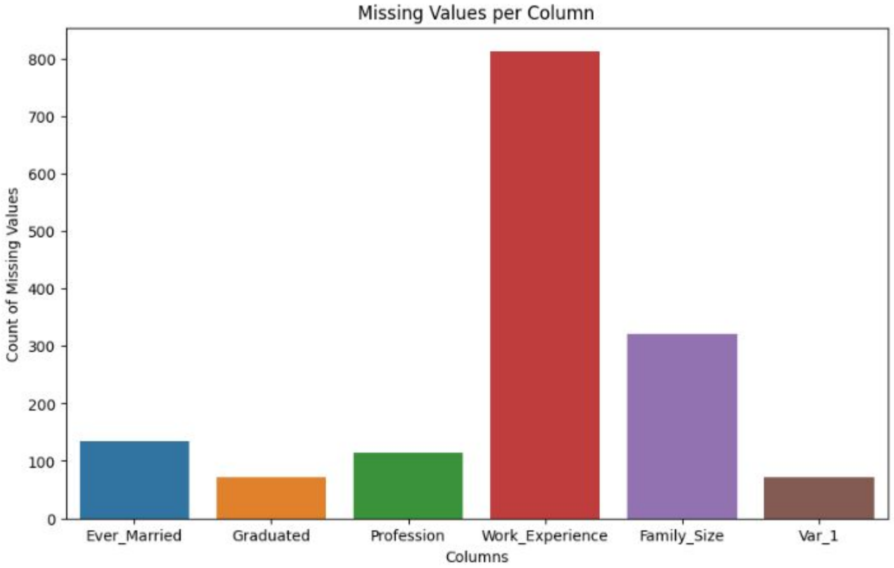
Handling Missing Values

- **General Approach**:
  - For **numerical features**, missing values are typically filled using:
    - **Mean**: When data is symmetrically distributed.
    - **Median**: When data is skewed or contains outliers.
  - For **categorical features**, missing values are commonly replaced with:
    - **Mode**: The most frequently occurring category.

Handling Missing Values

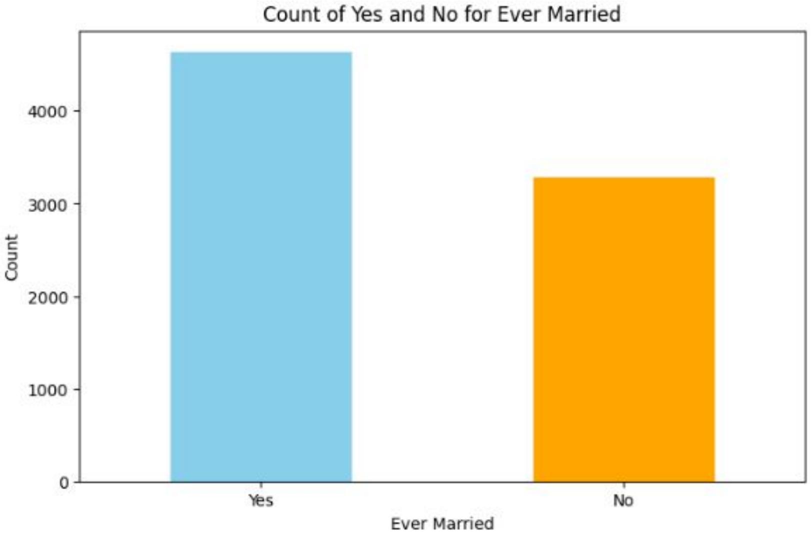
- **Project-Specific Strategy**:
  - In this project, we identified **patterns in the data** and filled missing values based on those observed relationships, ensuring the imputation aligned with the data's context and integrity.
  - This approach allowed us to preserve meaningful relationships in the dataset, potentially improving model accuracy and interpretability.
- **Benefits of Pattern-Based Imputation**:
  - Retains **contextual accuracy** by leveraging domain-specific insights.
  - Reduces the risk of introducing bias compared to generic imputation methods.
  - Ensures consistency with existing data distributions and patterns.

Data Pre-processing



Let's tackle them one by one by analysing any underlying patterns present.

Data Pre-processing



Ever_Married Column

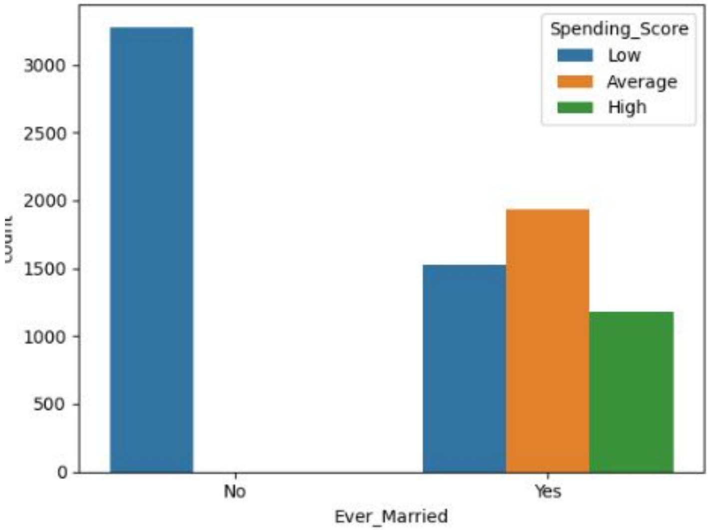
By looking at the plot General notion is to fill the NULL values with YES as it is appearing more times in the dataset.

BUT!!!

Observations Regarding the Ever_Married Feature

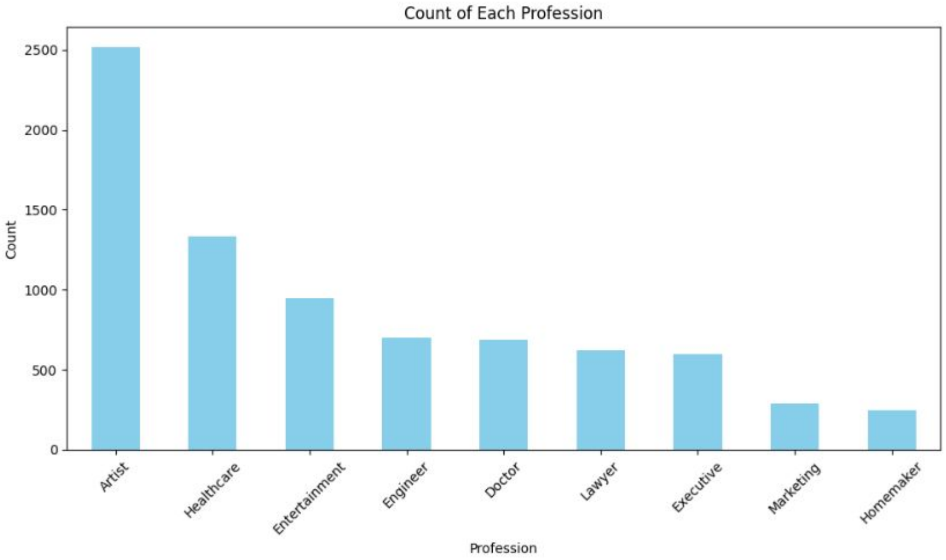
```
Spending_Score  Ever_Married
Average         Yes             1934
High            Yes             1175
Low             No              3280
                Yes             1526
Name: count, dtype: int64
```

- When the Spending_Score is **Average** or **High**, the Ever_Married status is predominantly **Yes** for all customers in the dataset.
- However, when the Spending_Score is **Low**, a significant proportion of the data indicates that the Ever_Married status is **No**.



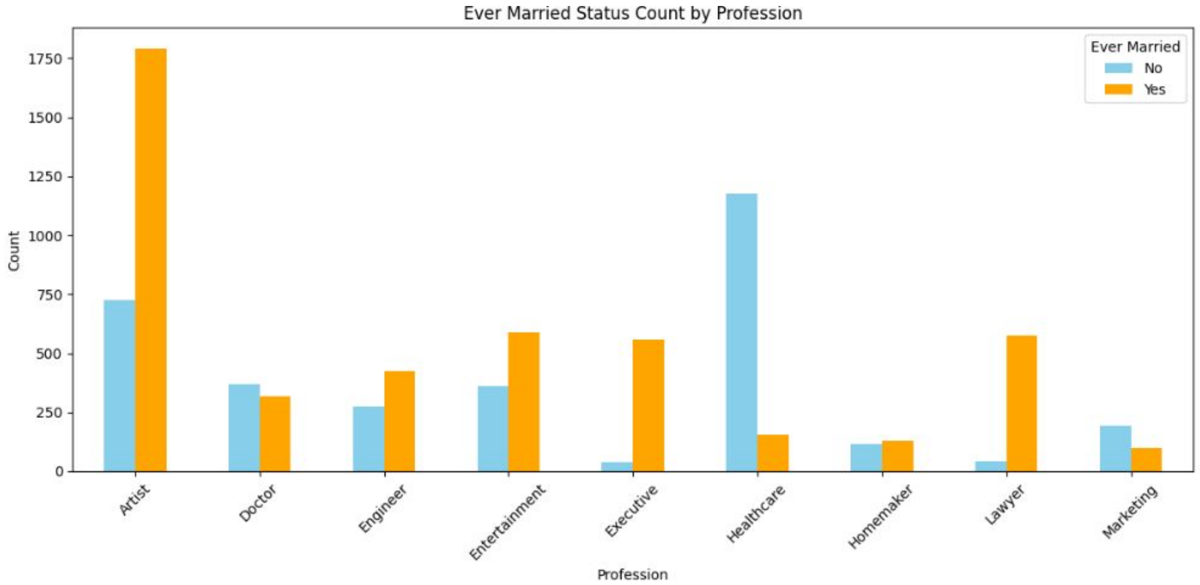
This pattern highlights that directly imputing missing values in the Ever_Married column with the most frequent value (e.g., "Yes") may lead to incorrect assumptions, especially for customers with a **Low Spending_Score**.

Data Pre-processing



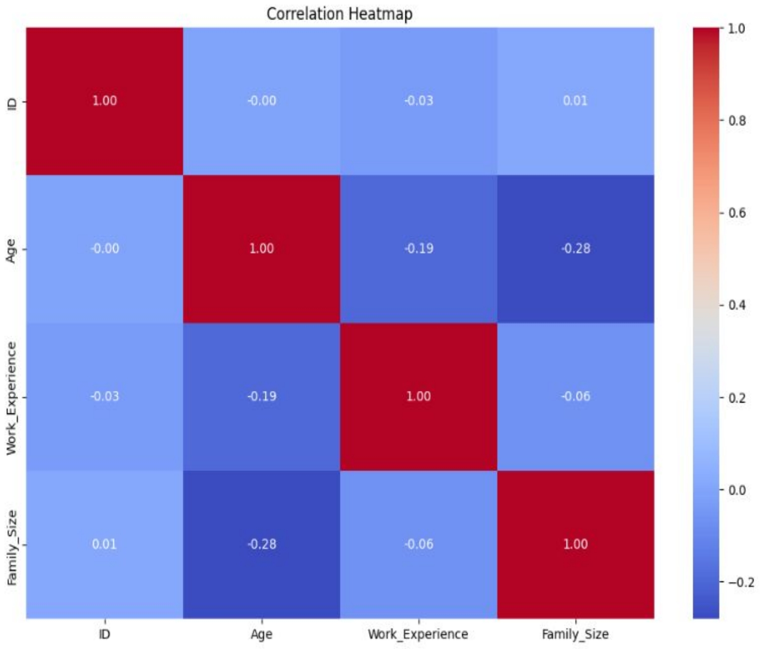
Here by mode we will fill null values with Artist only but we can observe few patterns here as well...

Data Pre-processing



- If a customer is marked as **"Ever Married = Yes"**, the majority of such customers belong to the **"Artist"** profession. Hence, missing values for married customers can be imputed as **"Artist"** based on this trend.
- Conversely, for customers marked as **"Ever Married = No"**, the majority belong to the **"Healthcare"** profession. Thus, missing values for unmarried customers should be imputed as **"Healthcare"**.

Data Pre-processing



None of the numerical features are **highly correlated** with each other which means:

1. There's **no multicollinearity** issue (you don't need to drop features to avoid redundancy).
2. Features appear to provide **independent** information, which is good for modeling.

Handling Categorical Columns

Categorical data is often encoded to be used in machine learning models. Two commonly used methods are **One-Hot Encoding** and **Label Encoding**. Steps Involved are:

1. Defining Columns

- **Ordinal Columns (ordinal_cols):**
  - These are columns where the categories have an inherent order (e.g., Low < Medium < High).
  - Examples: 'Age_Bin', 'Work_Exp_Category', 'Family_Size_Category'.
- **Nominal Columns (nominal_cols):**
  - These are columns where categories have no specific order (e.g., Gender, Profession).

Handling Categorical Columns

2. Label Encoding for Ordinal Columns

Label Encoding assigns a unique integer to each category **while preserving the order**.

- For example: If Age_Bin = ['Young', 'Middle-Aged', 'Old']:
  It will be encoded as: Young: 0, Middle-Aged: 1, Old: 2.

3. One-Hot Encoding for Nominal Columns

One-Hot Encoding creates binary columns for each category in nominal columns. Each category gets its own column, and the value is 1 if the row corresponds to that category and 0 otherwise.

Example: If Gender = ['Male', 'Female'], it creates:

- Gender_Male and Gender_Female.

Handling Categorical Columns

Why Was This Done?

1. **Machine Learning Models Require Numeric Data**:
   - Models like Logistic Regression, Decision Trees, or Neural Networks cannot directly handle categorical data.
2. **Preserved Information**:
   - Label Encoding was used for ordinal data to preserve its order.
   - One-Hot Encoding was used for nominal data to ensure no ordinal assumptions are made.
3. **Improves Model Performance**:
   - This encoding strategy ensures the model can correctly interpret both ordinal and nominal categorical data.

Scaling Numerical Features

- **Ensure Equal Contribution**:
  - Features like Age, Work_Experience, and Family_Size might have very different ranges.
    For example: Age: [18, 65], Work_Experience: [0, 40], Family_Size: [1, 10]
  - Machine learning models that use distance-based metrics (e.g., Logistic Regression, SVM, k-NN) may give more weight to features with larger ranges, leading to biased results.
- **Key Takeaways**
  - Scaling improves model performance, particularly for algorithms that depend on distance metrics or gradient-based optimization.
  - Tree-based models (e.g., Random Forest, XGBoost) don’t require scaling, but it doesn’t harm to apply it.

Model Training and Evaluation

- **Data Splitting**
  - The dataset was split into **80% training** and **20% testing** sets to evaluate model performance effectively.
- **Models Trained**
  - A variety of machine learning models were trained and evaluated:

- XGBoost Classifier
- Logistic Regression
- Random Forest Classifier
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)
- Gradient Boosting
- CatBoost Classifier
- LightGBM Classifier
- Voting Classifier (Ensemble of the best-performing models)

Results

- Among all the models, **XGBoost Classifier** achieved the **best performance**.
- Below are the key metrics for the XGBoost Classifier:

```
Training Accuracy: 0.5875207067918278

Testing Accuracy: 0.5739130434782609

Classification Report:
              precision    recall  f1-score   support

           A       0.49      0.52      0.50       197
           B       0.49      0.36      0.41       185
           C       0.60      0.62      0.61       197
           D       0.67      0.76      0.71       226

    accuracy                           0.57       805
   macro avg       0.56      0.56      0.56       805
weighted avg       0.57      0.57      0.57       805
```

Results

Model	Training Accuracy	Testing Accuracy
XGBoost Classifier	0.5767	0.5689
Logistic Regression	0.5183	0.5142
Random Forest Classifier	0.6120	0.5565
SVM	0.5712	0.5329
KNN	0.5828	0.4981
Gradient Boosting	0.5731	0.5478
CatBoost Classifier	0.5491	0.5552
LightGBM Classifier	0.6546	0.5291

Using the best model we have predicted the output on test.csv file.

THANK YOU!!!