Note: If you want to re-run the AutoML code, you may need to downgrade NumPy and restart the session. After restarting, re-import NumPy. This step is only necessary if you intend to re-run the AutoML code.

In [1]:
```
from flaml import AutoML
```

In [2]:
```
import ipywidgets as widgets
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import shap

# from flaml import AutoML
from sklearn.cluster import KMeans
from sklearn.datasets import load_wine
from sklearn.decomposition import PCA
from sklearn.ensemble import (
    GradientBoostingClassifier,
    GradientBoostingRegressor,
    RandomForestClassifier,
    RandomForestRegressor,
)
from sklearn.linear_model import LinearRegression
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix,
    mean_absolute_error,
    mean_squared_error,
    r2_score,
)
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.tree import DecisionTreeClassifier
```

Assignment: Understanding House Price Prediction Using Regression Models

Objective: The goal of this assignment is to explore the relationship between house prices and various features using exploratory data analysis (EDA) and regression models. By the end of the assignment, students will be able to visualize data, apply regression techniques, and compare model performances. Before we get started, we import all the packages used in this problem set.

Part 1: Dataset Overview and Preprocessing

1.**Dataset Overview**

The dataset contains house sale prices and various features such as:

date: The date the house was sold.

price: Target variable representing house sale prices.

bedrooms: Number of bedrooms.

bathrooms: Number of bathrooms.

sqft_living: Square footage of the living space.

sqft_lot: Square footage of the lot.

floors: Number of floors.

waterfront: Indicator if the house is located on a waterfront.

view: View rating of the house.

condition: Condition rating of the house.

sqft_above: Square footage of the living space above ground.

sqft_basement: Square footage of the basement.

yr_built: Year the house was built.

yr_renovated: Year of renovation (0 if never renovated).

street: Street address of the property.

city: City where the house is located.

statezip: State and zip code.

country: Country of the property.

The dataset includes both numerical and categorical features.

Check the first few rows of the dataset:

In [5]:
```
df = pd.read_csv(
    "https://raw.githubusercontent.com/chansen776/MBA-ML-Course-Materials/refs/heads/main/Data/house_prices.csv"
)
display(df.head())
columns = df.columns
columns
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	st
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	1340	0	1955	2005	18 Densn Av
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	3370	280	1921	0	70 Blair
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	1930	0	1966	0	26 26 143rd
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	1000	1000	1963	0	857 1 P
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	1140	800	1976	1992	9 170th

Out[5]:
```
Index(['date', 'price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot',
       'floors', 'waterfront', 'view', 'condition', 'sqft_above',
       'sqft_basement', 'yr_built', 'yr_renovated', 'street', 'city',
       'statezip', 'country'],
      dtype='object')
```

2.**Feature Engineering & Encoding Categorical Data**

Convert categorical features (city, statezip, street) into numerical representations.

Apply one-hot encoding where necessary:

In [6]:
```
df = pd.get_dummies(df, columns=["city", "statezip"], drop_first=True)
```

3.**Feature Scaling**

Important: When standardizing data, ensure that the scaling is performed only on the training data to prevent data leakage.

The mean and standard deviation should be calculated from the training data and then applied to both the training and test sets.

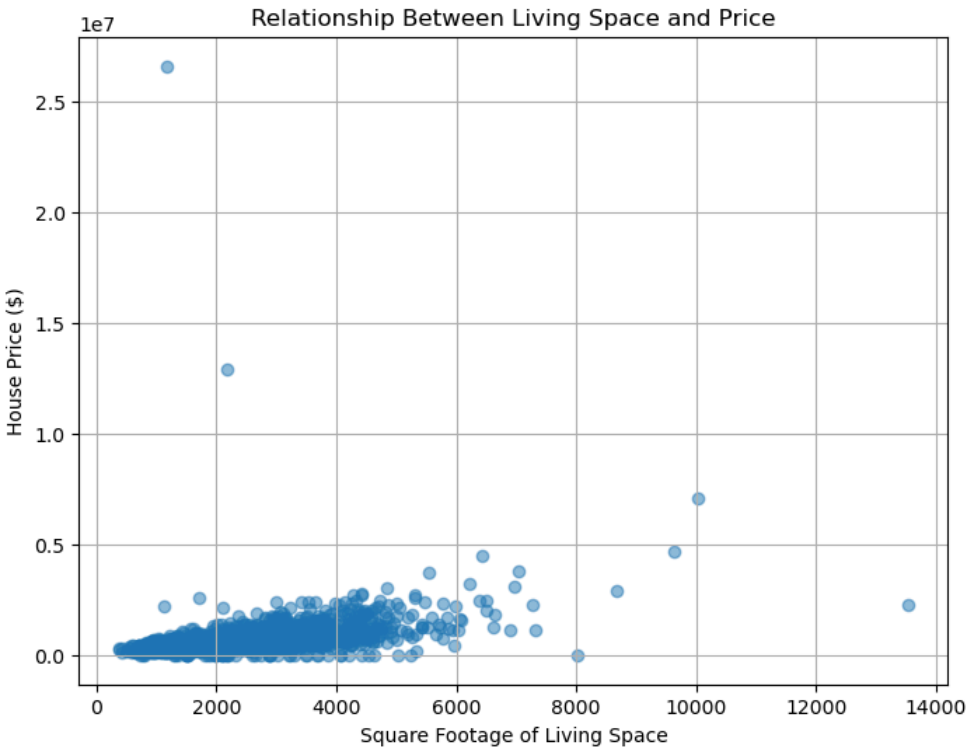
In [7]:
```
# Define predictor and target variable
X = df[["sqft_living"]]
y = df["price"]

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Initialize the StandardScaler
scaler = StandardScaler()

# Fit scaler only on training data and transform both training and test sets
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Part 2: Exploratory Data Analysis (EDA)

Visualization

1.Scatter Plot of sqft_living vs. price

In [8]:
```
plt.figure(figsize=(8, 6))
plt.scatter(df["sqft_living"], df["price"], alpha=0.5)
plt.xlabel("Square Footage of Living Space")
plt.ylabel("House Price ($)")
plt.title("Relationship Between Living Space and Price")
plt.grid(True)
plt.show()
```



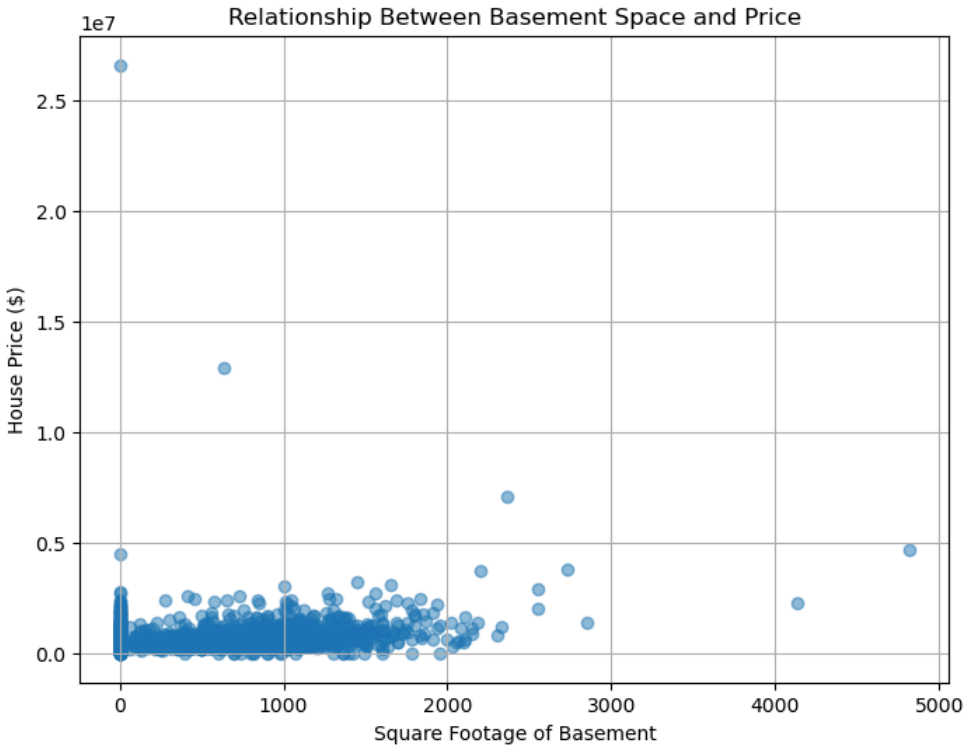
2.**Task:** Create a scatter plot of sqft_basement vs. price.

You have code for the scatter plot of sqft_living vs. price above.

The variable name you need is sqft_basement.

You can use ChatGPT or any other LLM to ask how to modify the provided code.

In [10]:
```
# Insert code and plot here
plt.figure(figsize=(8, 6))
plt.scatter(df["sqft_basement"], df["price"], alpha=0.5)
plt.xlabel("Square Footage of Basement")
plt.ylabel("House Price ($)")
plt.title("Relationship Between Basement Space and Price")
plt.grid(True)
plt.show()

# Calculate correlation coefficients
corr_sqft_basement = df["sqft_basement"].corr(df["price"])
corr_sqft_living = df["sqft_living"].corr(df["price"])

print(f"Correlation between price and sqft_basement: {corr_sqft_basement}")
print(f"Correlation between price and sqft_living: {corr_sqft_living}")
```




```
Correlation between price and sqft_basement: 0.21042657177482943
Correlation between price and sqft_living: 0.43041002543262824
```

3.**Question:** Which feature do you think has a stronger correlation with price: sqft_living or sqft_basement? Calculate the correlation coefficient for both and explain your reasoning.

Your Answer:

Looks like sqft_living has a higher corr coeff with price than sqft_basement, looks like there's an opportunity to think more closely about how we deal with `0` values for sqft_basement.

Part 3: Building Regression Models

1.Linear Regression (Example Given)

2.Train a **Linear Regression model** using sqft_living as the predictor.

3.Evaluate the model using R-squared (R^2), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE)

In [11]:
```
X = df[["sqft_living"]]
y = df["price"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("R²:", r2_score(y_test, y_pred))
print("MAE:", mean_absolute_error(y_test, y_pred))
print("RMSE:", mean_squared_error(y_test, y_pred))
```

```
R²: 0.029065410341410414
MAE: 225375.25345857345
RMSE: 990204087727.1417
```

Random Forest and Gradient Boosting Regression

Below is the code for Random Forest and Gradient Boosting Regression. Do not worry about understanding every part of the code initially.

In [12]:
```
# Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train_scaled, y_train)
y_pred_rf = rf_model.predict(X_test_scaled)

print("Random Forest Results:")
print("R²:", r2_score(y_test, y_pred_rf))
print("MAE:", mean_absolute_error(y_test, y_pred_rf))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_rf)))
```

```
# Gradient Boosting Regressor
gb_model = GradientBoostingRegressor(
    n_estimators=100, learning_rate=0.1, random_state=42
)
gb_model.fit(X_train_scaled, y_train)
y_pred_gb = gb_model.predict(X_test_scaled)

print("Gradient Boosting Results:")
print("R²:", r2_score(y_test, y_pred_gb))
print("MAE:", mean_absolute_error(y_test, y_pred_gb))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_gb)))
```

```
Random Forest Results:
R²: 0.005948601777127971
MAE: 233146.1318696576
RMSE: 1006866.2673542678
Gradient Boosting Results:
R²: 0.006307078747391892
MAE: 224314.3204614628
RMSE: 1006684.7018355782
```

Your Task:

1. Explain what the provided code is doing. You can use ChatGPT or any LLM to help you understand it.

We first create a single train-test split (80/20), then build a simple linear regression using only the sqft_living covariate. We then generate R^2, MAE, and RMSE. In the next cell, we build a random forest and gradient boosted forest using the same train-test split, still using the single covariate. Then calculate the OOS R^2, MAE, RMSE in a similar fashion.

Compare the Results:

2. Which model performed better based on the evaluation metrics (R^2, MAE, RMSE)? The linear model out-performed the other two models across all the evaluation metrics.

3. Consider the presence of outliers in the scatter plot from the beginning of the assignment. We can note small livable sqft $1.5M - 2.0M homes as potential outliers in our dataset. These are likely leading to rules which overfit for the two tree models.

4. (&5). Which evaluation metric (RMSE, R^2, or MAE) do you think is better for this example? Given that this is a simple, first-stage model, looking for large amounts of variation explianed, R^2, makes the most sense to me relative to more concrete evaluation functions like MAE or RMSE which intent to capture the closeness of the prediction in the end model to the true value. With respect to outliers, RMSE will be much more sensitive to these outliers relative to the MAE measure. I believe it's reasonable to not expect this model to handle outliers well, with that context, MAE may be a more relevant measure than RMSE.

Bonus Question:

1.Experiment by adding more features (e.g., bedrooms, bathrooms, sqft_basement). Compare which model performs best with multiple predictors.

Throwing in everything except the kitchen sink (including outliers and un-normalized data like yr-restored), we find that the linear regression captures the highest R^2 and lowest MAE. Given its poor performance in capturing outlier-like behavior, there are significant OOS error magnitudes for a few observations leading to a massive RMSE relative to the tree-based predictors which capture that behavior more reliably, but they fail to perform well in the "average case" leading to relatively worse MAE.

In [16]:
```
X = df[df.columns[~df.columns.isin(["price", "date", "street", "country"])]]
y = df["price"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("Linear Regression Results:")
print("R²:", r2_score(y_test, y_pred))
print("MAE:", mean_absolute_error(y_test, y_pred))
print("RMSE:", mean_squared_error(y_test, y_pred))
# Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train_scaled, y_train)
y_pred_rf = rf_model.predict(X_test_scaled)

print("\nRandom Forest Results:")
print("R²:", r2_score(y_test, y_pred_rf))
print("MAE:", mean_absolute_error(y_test, y_pred_rf))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_rf)))

# Gradient Boosting Regressor
gb_model = GradientBoostingRegressor(
```

```
    n_estimators=100, learning_rate=0.1, random_state=42
)
gb_model.fit(X_train_scaled, y_train)
y_pred_gb = gb_model.predict(X_test_scaled)

print("\nGradient Boosting Results:")
print("R²:", r2_score(y_test, y_pred_gb))
print("MAE:", mean_absolute_error(y_test, y_pred_gb))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_gb)))
```

```
Linear Regression Results:
R²: 0.05496892216006044
MAE: 155853.42077609673
RMSE: 963786486003.4923

Random Forest Results:
R²: 0.005948601777127971
MAE: 233146.1318696576
RMSE: 1006866.2673542678

Gradient Boosting Results:
R²: 0.006307078747391892
MAE: 224314.3204614628
RMSE: 1006684.7018355782
```

Multi-class Classification

In many predictive modeling tasks, the goal is to classify observations into distinct categories based on their characteristics. When there are only two possible categories (e.g., "Yes" or "No," "Up" or "Down"), the problem is called binary classification. However, when there are **more than two possible categories**, we use **multi-class classification**.

An automobile company is expanding into new markets with its existing products. Market research shows that customer behavior in these markets mirrors the existing one.

Previously, the sales team classified customers into four segments (A, B, C, D) using rule-based methods based on demographics, spending behavior, and past purchases. To scale this approach, the company now requires a multi-class machine learning model to automate customer segmentation for new market entrants.

The A, B, C, D segmentation represents customer tiers in the automobile market:

- A (Loyal Customers) – High-spending, repeat buyers, likely to prefer premium or luxury cars.
- B (Regular Customers) – Moderate spenders, consistent buyers, often opting for mid-range vehicles.
- C (Occasional Customers) – Low-spending, infrequent buyers, likely considering budget-friendly or used cars.
- D (Potential Customers) – Least engaged, may be first-time buyers or exploring options with low commitment.

Your task is to develop a predictive solution that accurately assigns new customers to the appropriate segment, enabling data-driven decision-making and scalable outreach.

Dataset_overview:

Dataset Overview The dataset contains information about automobile customers. Each record represents an individual customer with various demographic and behavioral attributes. The key variables in the dataset include:

- **ID**: Unique identifier for each customer. Gender: Customer's gender.
- **Ever_Married**: Indicates if the customer has ever been married.
- **Age**: Customer's age.
- **Graduated**: Indicates whether the customer is a graduate.
- **Profession**: Customer's occupation.
- **Work_Experience**: Number of years the customer has worked.
- **Spending_Score**: Indicates spending behavior, classified as Low, Average, or High based on purchasing patterns.
- **Family_Size**: Number of family members in the household.
- **Hidden_Customer_Category**: An internally assigned, anonymized classification of customers based on undisclosed criteria, such as spending behavior, or engagement patterns.

Discussion Question: Binary vs. Multi-Class Classification prior to predictive tasks.

If you were primarily interested in predicting who your potential customers are—perhaps so you can target them with incentives to encourage engagement—would you prefer a binary classification model (Potential Customer: 0/1) or a multi-class classification model that categorizes users into the four segments (A, B, C, D)? You could use ChatGPT to help you think these tradeoffs.

Answer:

We'd prefer a binary classification model if the end use-case is simply potential customer prediction. This helps by reducing the problem complexity (confusion between classes). The primary value we're giving up in exchange for a simpler model (easier to predict, less bais/variance), is the flexibility we'd get from future usecases of using the other classes A/B/C/D vs D / Not-D.

Task 1: Data Preprocessing and Exploration

1.1 Load and Explore the Dataset

- Load the **train and test datasets**.
- Display summary statistics using `.describe()` and `.info()`.

In [23]:
```
# change to your file directory
automobile_customers = pd.read_csv(
    "https://raw.githubusercontent.com/chansen776/MBA-ML-Course-Materials/refs/heads/main/Data/automobile_customer_segmentation.csv"
)
automobile_customers.info()
automobile_customers.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8570 entries, 0 to 8569
Data columns (total 11 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   ID                       8500 non-null   float64
 1   Gender                   8570 non-null   object
 2   Ever_Married             8453 non-null   object
 3   Age                      8478 non-null   float64
 4   Graduated                8481 non-null   object
 5   Profession               8570 non-null   object
 6   Work_Experience          8478 non-null   float64
 7   Spending_Score           8570 non-null   object
 8   Family_Size              8471 non-null   float64
 9   Hidden_Customer_Category  8479 non-null   object
 10  Segmentation             8570 non-null   object
dtypes: float64(4), object(7)
memory usage: 736.6+ KB
```

Out[23]:

	ID	Age	Work_Experience	Family_Size
count	8500.000000	8478.000000	8478.000000	8471.000000
mean	463544.991412	45.363883	4.197924	3.091134
std	2576.241780	16.155434	3.987977	1.625129
min	458985.000000	18.000000	0.000000	1.000000
25%	461420.000000	33.000000	1.000000	2.000000
50%	463584.500000	42.000000	2.000000	3.000000
75%	465730.000000	55.000000	8.000000	4.000000
max	467973.000000	89.000000	14.000000	9.000000

In [24]:
```
display(automobile_customers.isnull().sum())
automobile_customers.dropna(inplace=True)
```

```
ID                          70
Gender                       0
Ever_Married               117
Age                         92
Graduated                   89
Profession                   0
Work_Experience             92
Spending_Score               0
Family_Size                 99
Hidden_Customer_Category    91
Segmentation                 0
dtype: int64
```

Question: What are the two code lines above doing? (You could use your favorite LLM to help).

Answer:

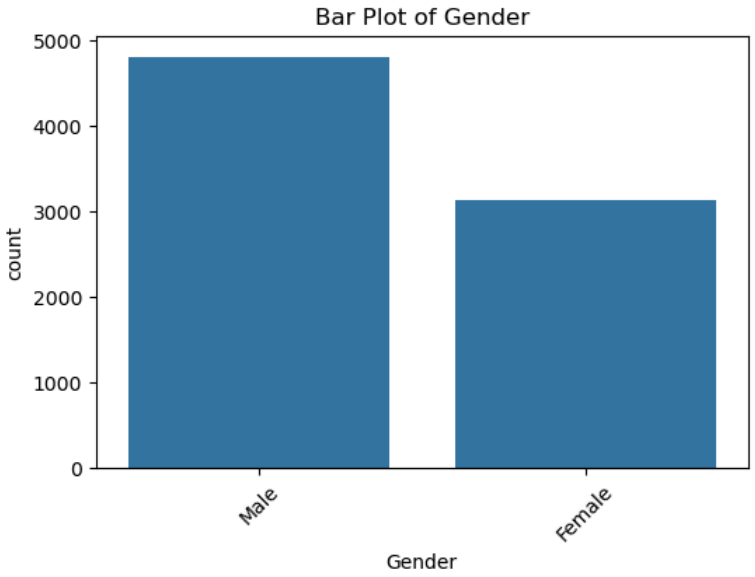
The first line intends to give us a count of the null-cells in the dataset, the second-line drops rows that contain one or more null cells.

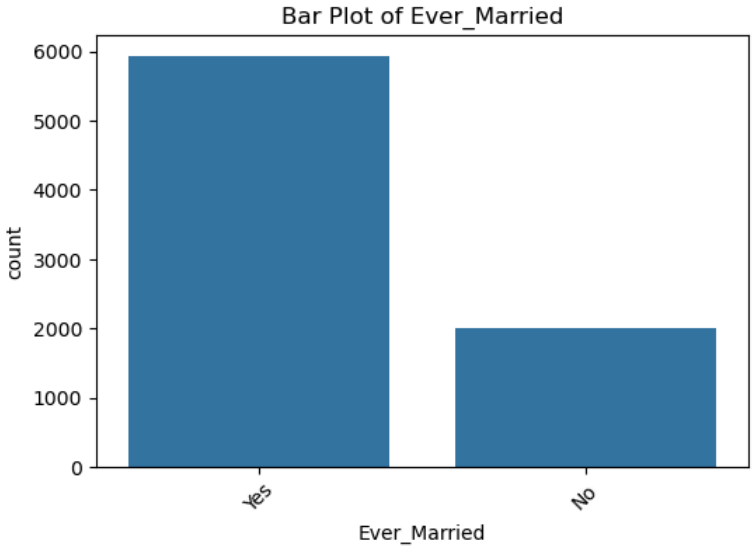
1.2 Visualizing Categorical Variables Using Bar Plots

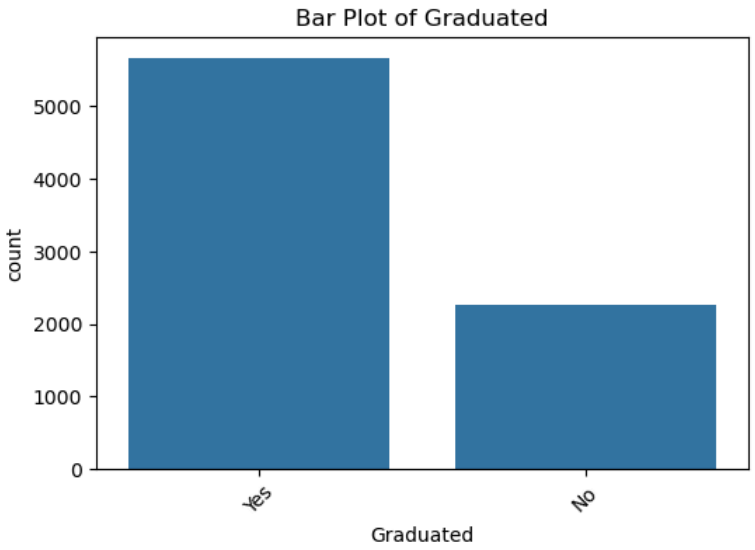
- Use **bar plots** to visualize the distribution of categorical variables.
- Focus on key categorical features such as **Gender, Ever_Married, Graduated, Profession, Spending_Score, Hidden_Customer_Category, and Segmentation**.
- Bar plots help understand the frequency of each category and identify potential imbalances.

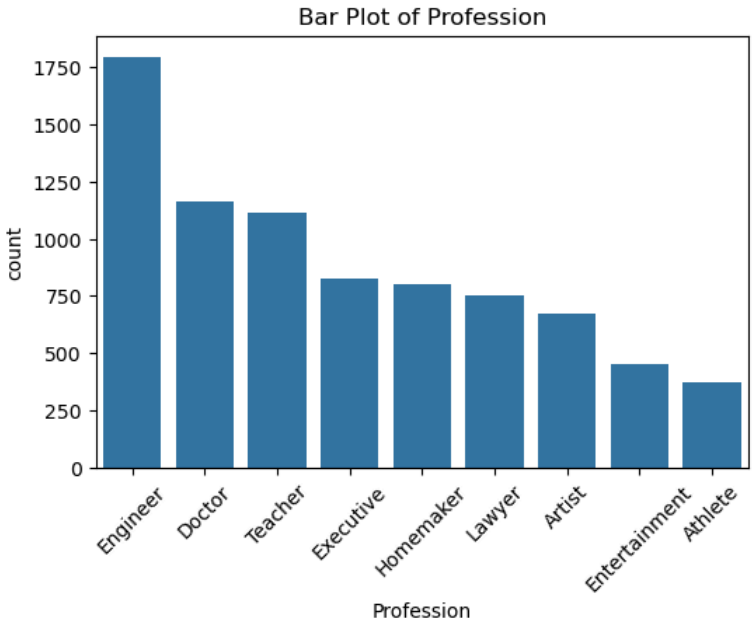
In [25]:
```
# Generate bar plots for each categorical column
categorical_cols = [
    "Gender",
    "Ever_Married",
    "Graduated",
```

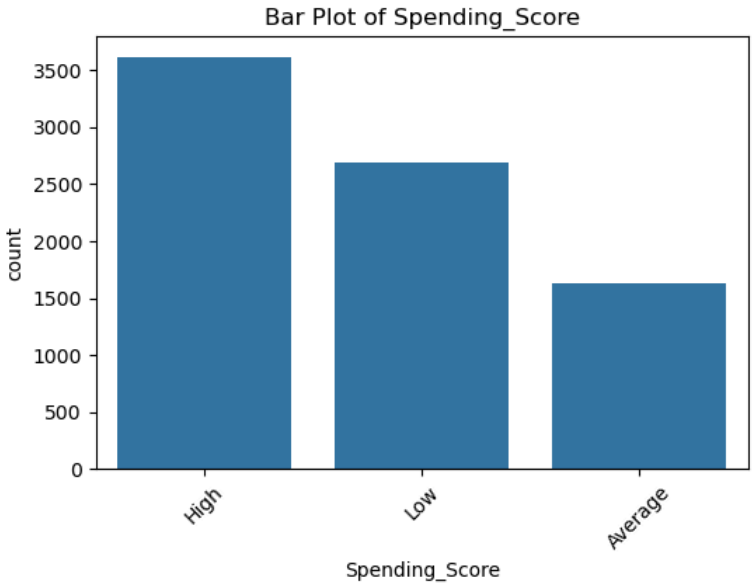
```
    "Profession",
    "Spending_Score",
    "Hidden_Customer_Category",
    "Segmentation",
]

for col in categorical_cols:
    plt.figure(figsize=(6, 4))
    sns.countplot(
        x=automobile_customers[col],
        order=automobile_customers[col].value_counts().index,
    )
    plt.title(f"Bar Plot of {col}")
    plt.xticks(rotation=45)
    plt.show()
```

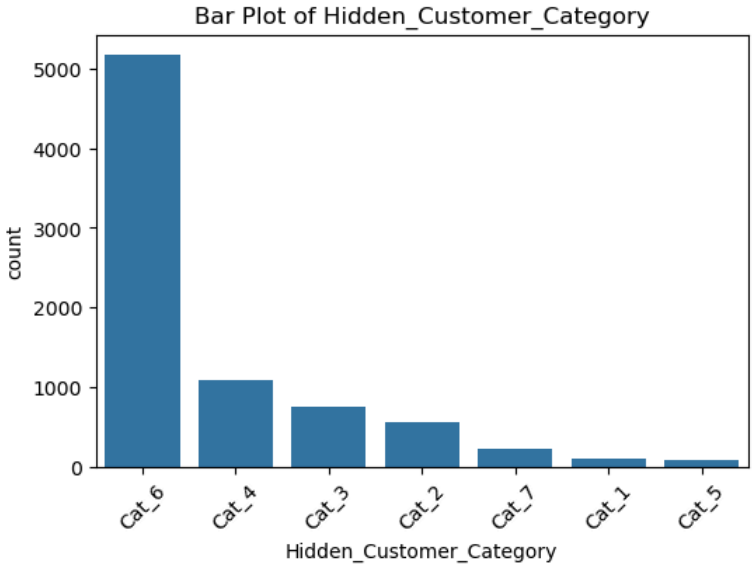


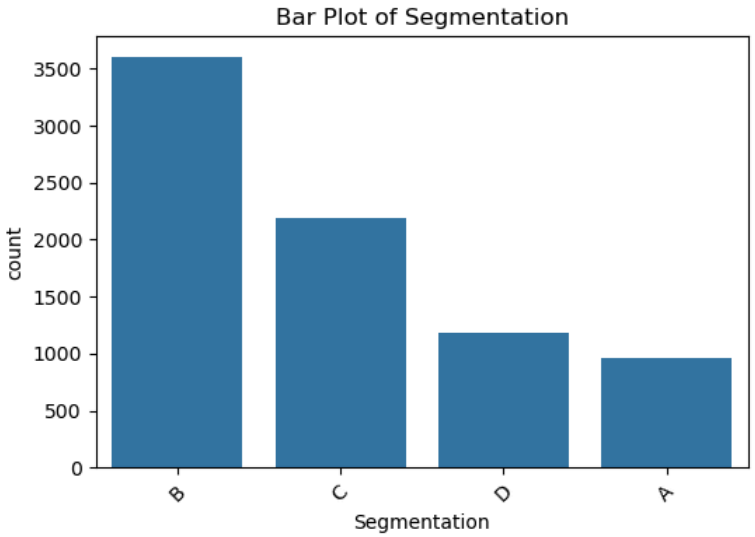





Bar Plot of Profession
count
1750
1500
1250
1000
750
500
250
0
Engineer
Doctor
Teacher
Executive
Homemaker
Lawyer
Artist
Entertainment
Athlete
Profession


Bar Plot of Spending_Score
count
3500
3000
2500
2000
1500
1000
500
0
High
Low
Average
Spending_Score

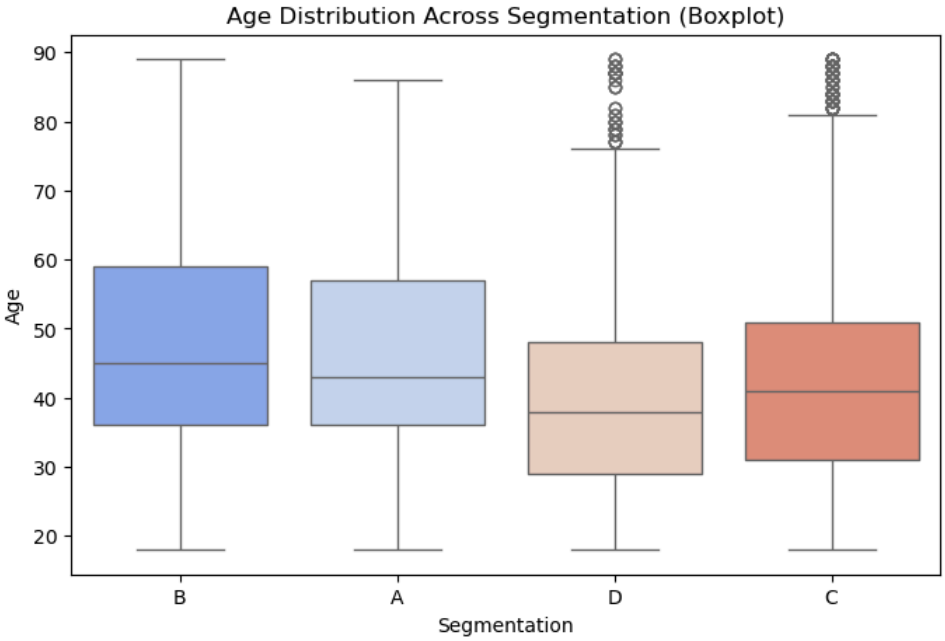

Bar Plot of Hidden_Customer_Category
count
5000
4000
3000
2000
1000
0
Cat_6
Cat_4
Cat_3
Cat_2
Cat_7
Cat_1
Cat_5
Hidden_Customer_Category

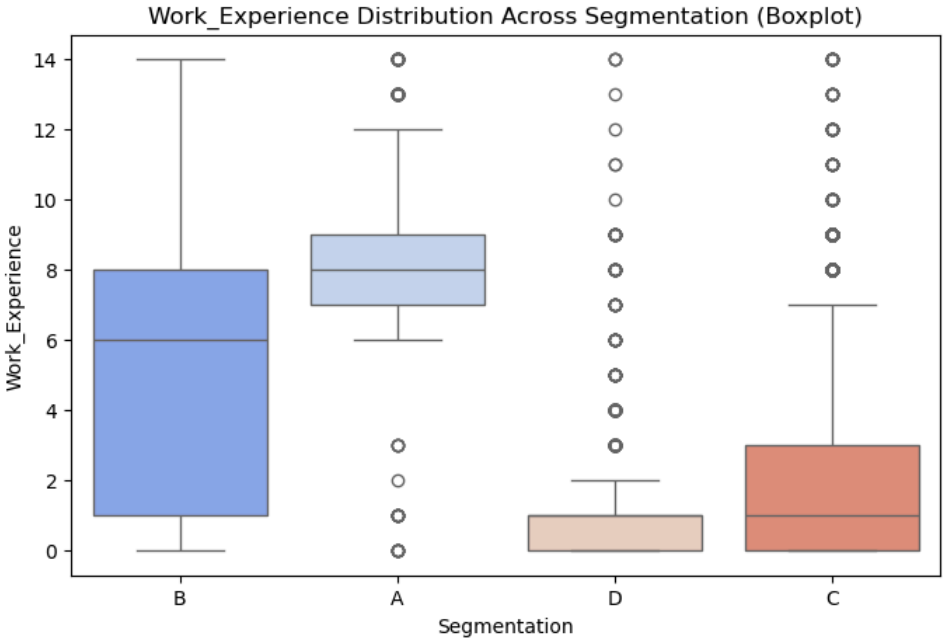


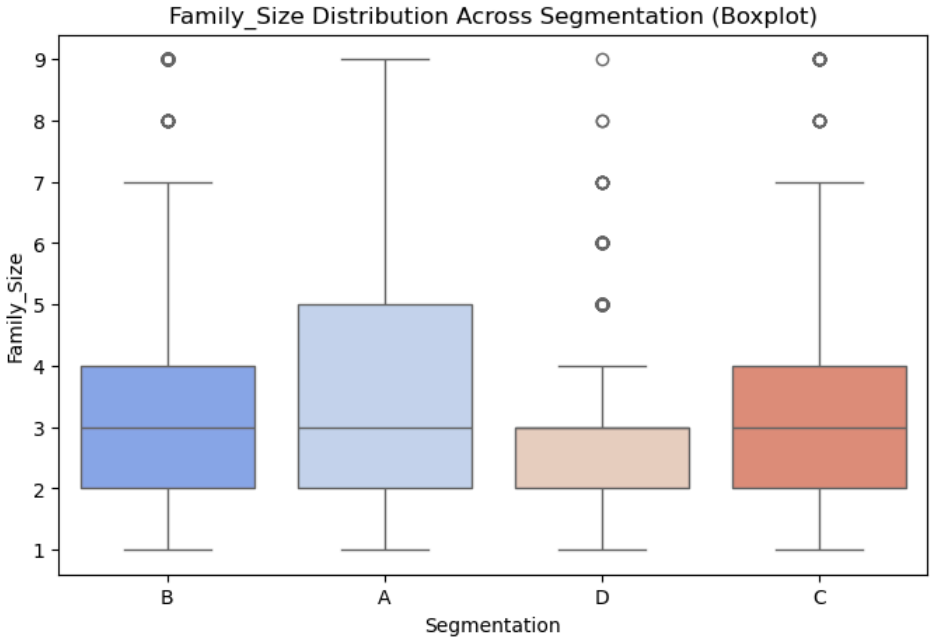
1.3 Discover Correlation with Other Variables

Investigate whether certain **demographic or behavioral attributes** are strong predictors of segmentation.

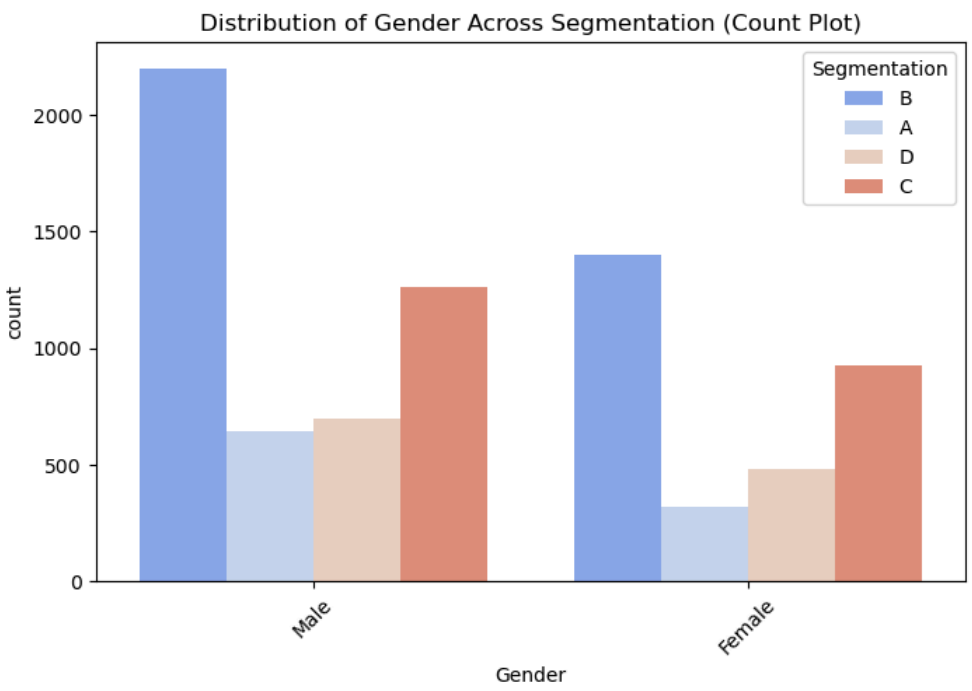
In [ ]:
```
# Boxplots for Numerical Features vs. Segmentation
# Boxplots
numerical_cols = ["Age", "Work_Experience", "Family_Size"]

for col in numerical_cols:
    plt.figure(figsize=(8, 5))
    sns.boxplot(x="Segmentation", hue="Segmentation", y=col, data=automobile_customers, palette="coolwarm")
    plt.title(f"{col} Distribution Across Segmentation (Boxplot)")
    plt.xlabel("Segmentation")
    plt.ylabel(col)
    plt.show()
```

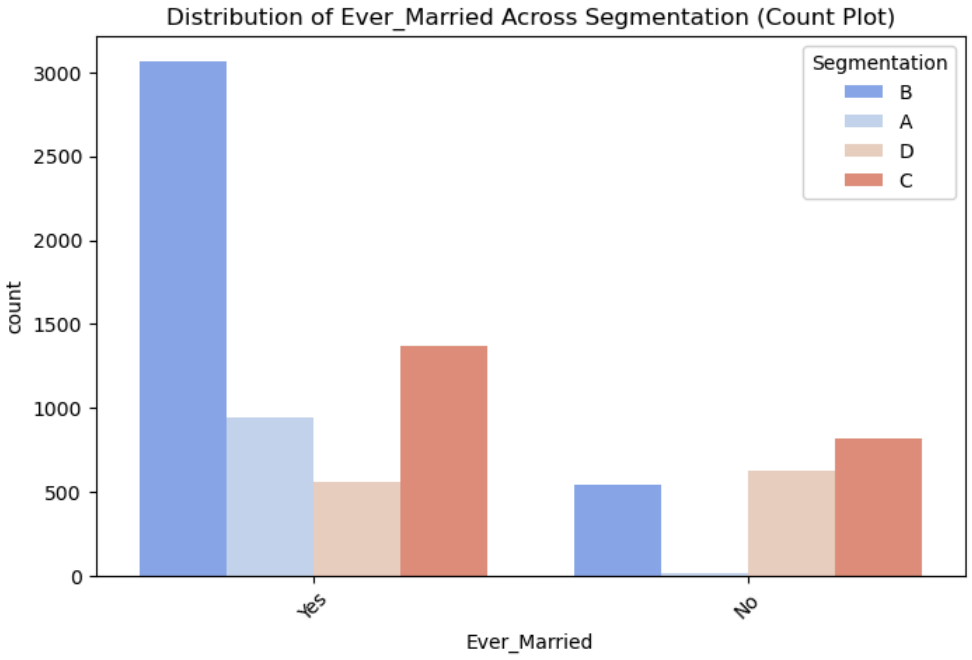


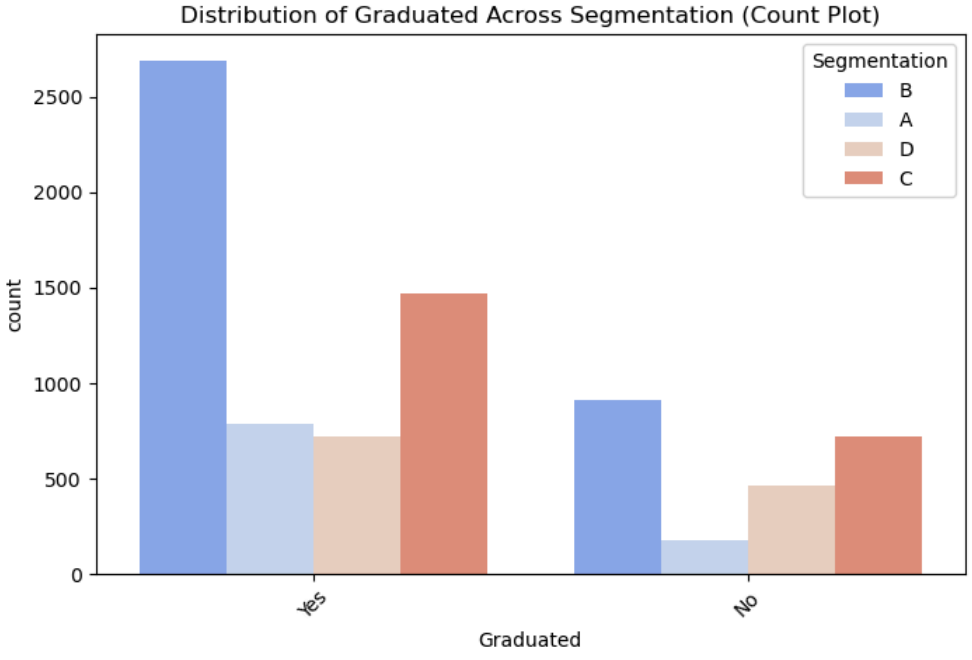


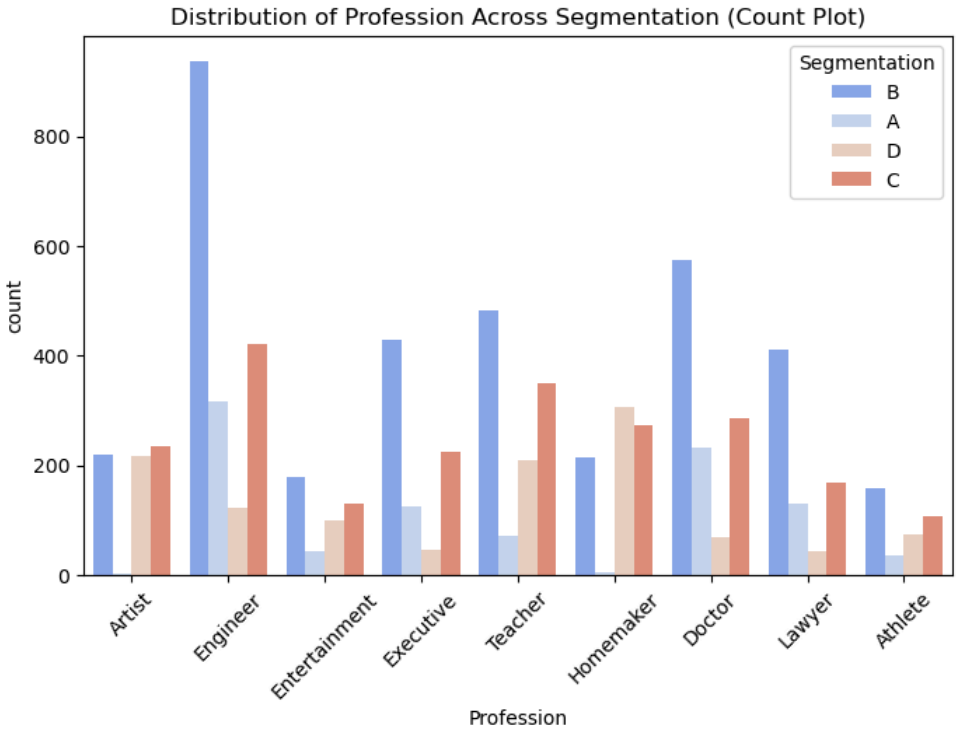


In [30]:
```python
# Countplots for Categorical Features vs. Segmentation
# Countplots
categorical_cols = [
    "Gender",
    "Ever_Married",
    "Graduated",
    "Profession",
    "Spending_Score",
    "Hidden_Customer_Category",
]

for col in categorical_cols:
    plt.figure(figsize=(8, 5))
    sns.countplot(
        x=col, hue="Segmentation", data=automobile_customers, palette="coolwarm"
    )
    plt.title(f"Distribution of {col} Across Segmentation (Count Plot)")
    plt.xticks(rotation=45)
    plt.legend(title="Segmentation")
    plt.show()
```


Distribution of Gender Across Segmentation (Count Plot)
Segmentation
B
A
D
C
count
2000
1500
1000
500
0
Male
Female
Gender


Distribution of Ever_Married Across Segmentation (Count Plot)
Segmentation
B
A
D
C
count
3000
2500
2000
1500
1000
500
0
Yes
No
Ever_Married


Distribution of Graduated Across Segmentation (Count Plot)
Segmentation
B
A
D
C
count
2500
2000
1500
1000
500
0
Yes
No
Graduated


Distribution of Profession Across Segmentation (Count Plot)
Segmentation
B
A
D
C
count
0
200
400
600
800
Artist
Engineer
Entertainment
Executive
Teacher
Homemaker
Doctor
Lawyer
Athlete
Profession


Distribution of Spending_Score Across Segmentation (Count Plot)
Segmentation
B
A
D
C
count
0
500
1000
1500
2000
Average
Low
High
Spending_Score

Question: Analyze and Interpret Data Insights

1. Summarize your key insights from the dataset, focusing on patterns or trends that could influence modeling decisions for customer segmentation.

D segment (potential customers) appear to be representitive of younger, low work experience, low-salary individuals.

2. Here, we examine the full dataset before modeling. Is there a potential issue with analyzing the entire dataset before constructing models based on these insights? What risks might arise from this approach?

We're potentially leaking data into our model in-advertently through biases in the modeling decsions we'll make as model creators.

1.4 Convert Categorical Variables and Encode Data

Encode categorical variables using **LabelEncoder** for all categorical columns, except the target column `Segmentation`. **LabelEncoder** will be applied to **all categorical variables**, including both binary and multi-class categories.

In [31]:
```
# List of categorical columns (excluding the target 'Segmentation')
categorical_cols = [
    "Gender",
    "Ever_Married",
    "Graduated",
    "Profession",
    "Spending_Score",
    "Hidden_Customer_Category",
]

# Initialize LabelEncoder
encoder = LabelEncoder()

# Loop through each categorical column and apply LabelEncoder
for col in categorical_cols:
    automobile_customers[col] = encoder.fit_transform(
        automobile_customers[col].astype(str)
    )
```

Task 2: Build & Compare Classification Models

For Task 2, you'll follow these steps:

- 2.1 Split the dataset into train and test sets.
- 2.2 Train 3 multi-class classification models covered in Note2:
  - Logistic Regression
  - Decision Tree Classifier
  - Random Forest Classifier
- 2.3 Evaluate the models using performance metrics such as accuracy, precision, recall, F1-score, and confusion matrix.

2.1 Split the Dataset into Train and Test Sets

- Use `train_test_split` to split the data into **80% training** and **20% testing** sets.

- Ensure the split is **randomized** to help the model generalize (set_seed= `42`).

In [32]:
```
# Separate features (X) and target (y)
X = automobile_customers.drop(columns=["Segmentation", "ID"])
y = automobile_customers["Segmentation"]

# Perform the train-test split (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print(f"Training set shape: {X_train.shape}")
print(f"Testing set shape: {X_test.shape}")
```

```
Training set shape: (6352, 9)
Testing set shape: (1589, 9)
```

Question: What did you notify about the `train_test_split` method? Why we use `stratify` parameter?

Answer:

I noticed that we're stratifying across segments, this helps ensure that our training and testing splits have proportional counts of each segment occuring.

2.2 Train 3 Multi-Class Classification Models

1. Train the following models using the prepared dataset:
   - **Decision Tree Classifier**
   - **Random Forest Classifier**
   - **Gradient Boosting Classifier**
2. Since tree-based models do not require feature scaling, use the raw numerical features for training.
3. **Perform Grid Search Cross-Validation (GridSearchCV) with 5-Fold Cross-Validation (cv=5)** to optimize hyperparameters
4. Fit each model to the training dataset and generate predictions on the test dataset.
5. Evaluate the models using accuracy, precision, recall, and F1-score, and analyze the results.

In [33]:
```
# Define hyperparameter grid
param_grid_dt = {
    "max_depth": [10, 20, 30],
    "min_samples_split": [2, 5, 10],
    "min_samples_leaf": [1, 2, 4],
}

# Perform Grid Search
dt_grid = GridSearchCV(
    DecisionTreeClassifier(random_state=42),
    param_grid_dt,
    cv=5,
    scoring="accuracy",
    n_jobs=-1,
    verbose=2,
)
dt_grid.fit(X_train, y_train)

# Get best model and evaluate
best_dt = dt_grid.best_estimator_
y_pred_dt = best_dt.predict(X_test)

print("\nDecision Tree Best Parameters:", dt_grid.best_params_)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))
print(
    "Decision Tree Classification Report:\n", classification_report(y_test, y_pred_dt)
)
```

```
Fitting 5 folds for each of 27 candidates, totalling 135 fits
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2; total time=   0.0s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2; total time=   0.0s[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5;
total time=   0.0s

[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2; total time=   0.0s[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5;
total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10; total time=   0.0s

[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5; total time=   0.0s[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10;
total time=   0.0s
```

```
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2; total time=   0.0s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10; total time=   0.0s[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2;
total time=   0.0s

[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10; total time=   0.0s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10; total time=   0.0s

Decision Tree Best Parameters: {'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 2}
Decision Tree Accuracy: 0.7734424166142227
Decision Tree Classification Report:
              precision    recall  f1-score   support

           A       0.71      0.88      0.79       193
           B       0.85      0.89      0.87       721
           C       0.70      0.68      0.69       438
           D       0.70      0.50      0.58       237

    accuracy                           0.77      1589
   macro avg       0.74      0.74      0.73      1589
weighted avg       0.77      0.77      0.77      1589
```

Question: Briefly explain what `GridSearchCV` is doing. Why we do cross-validation in the training dataset?

Answer:

For the cartesian product of parameters to search, we're going to try all possible combinations, in each case we're going to run a cross-validation process where we take 1 fold and predict on it using the other 4 folds. This processes is repeated for all folds, generating an average score we're looking optimize via our parameter search. We do this with the training data so that we can later test, OOS, the most effective form of cross-validation without having leakage in our model.

Question: Based on the classification report for decision tree, what do terms like precision, recall, F1-score mean? Briefly interpret the decision tree classifier results on the test dataset. (You could use ChatGPT to help interpretation.)

Answer:

Precision is the the fraction of assignments to a category that are correct, recall is the the fraction total samples in a category that are correctly assigned to that category. The f1-score is a way to take these scores and combine them into a summary that's not biased with imbalanced classes

Training code for random forests with grid search cross-validation (would take around 5 minutes to run)

In [34]:

```
# Random Forests
# Define hyperparameter grid
param_grid_rf = {
    "n_estimators": [500],
    "max_depth": [10, 20, 30],
    "min_samples_split": [2, 5, 10],
}

# Perform Grid Search
rf_grid = GridSearchCV(
    RandomForestClassifier(random_state=42),
    param_grid_rf,
    cv=5,
    scoring="accuracy",
    n_jobs=-1,
    verbose=2,
)
rf_grid.fit(X_train, y_train)

# Get best model and evaluate
best_rf = rf_grid.best_estimator_
y_pred_rf = best_rf.predict(X_test)

print("\nRandom Forest Best Parameters:", rf_grid.best_params_)
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print(
    "Random Forest Classification Report:\n", classification_report(y_test, y_pred_rf)
)
```

```
Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV] END max_depth=10, min_samples_split=5, n_estimators=500; total time=   1.9s
[CV] END max_depth=10, min_samples_split=2, n_estimators=500; total time=   1.9s
[CV] END max_depth=10, min_samples_split=5, n_estimators=500; total time=   1.9s
[CV] END max_depth=10, min_samples_split=5, n_estimators=500; total time=   1.9s
[CV] END max_depth=10, min_samples_split=2, n_estimators=500; total time=   1.9s
[CV] END max_depth=10, min_samples_split=2, n_estimators=500; total time=   2.0s
[CV] END max_depth=10, min_samples_split=2, n_estimators=500; total time=   2.0s
[CV] END max_depth=10, min_samples_split=2, n_estimators=500; total time=   2.0s
[CV] END max_depth=10, min_samples_split=10, n_estimators=500; total time=   1.8s
[CV] END max_depth=10, min_samples_split=10, n_estimators=500; total time=   1.8s
[CV] END max_depth=10, min_samples_split=10, n_estimators=500; total time=   1.8s
[CV] END max_depth=10, min_samples_split=5, n_estimators=500; total time=   1.9s
[CV] END max_depth=10, min_samples_split=10, n_estimators=500; total time=   1.8s
[CV] END max_depth=10, min_samples_split=10, n_estimators=500; total time=   1.8s
[CV] END max_depth=10, min_samples_split=5, n_estimators=500; total time=   2.0s
[CV] END max_depth=20, min_samples_split=2, n_estimators=500; total time=   2.7s
[CV] END max_depth=20, min_samples_split=5, n_estimators=500; total time=   2.4s
[CV] END max_depth=20, min_samples_split=5, n_estimators=500; total time=   2.5s
[CV] END max_depth=20, min_samples_split=5, n_estimators=500; total time=   2.5s
[CV] END max_depth=20, min_samples_split=2, n_estimators=500; total time=   2.6s
[CV] END max_depth=20, min_samples_split=2, n_estimators=500; total time=   2.6s
[CV] END max_depth=20, min_samples_split=2, n_estimators=500; total time=   2.6s
[CV] END max_depth=20, min_samples_split=2, n_estimators=500; total time=   2.6s
[CV] END max_depth=20, min_samples_split=5, n_estimators=500; total time=   2.5s
[CV] END max_depth=20, min_samples_split=10, n_estimators=500; total time=   2.3s
[CV] END max_depth=20, min_samples_split=5, n_estimators=500; total time=   2.5s
[CV] END max_depth=20, min_samples_split=10, n_estimators=500; total time=   2.4s
[CV] END max_depth=20, min_samples_split=10, n_estimators=500; total time=   2.4s
[CV] END max_depth=20, min_samples_split=10, n_estimators=500; total time=   2.5s
[CV] END max_depth=20, min_samples_split=10, n_estimators=500; total time=   2.4s
[CV] END max_depth=30, min_samples_split=2, n_estimators=500; total time=   2.8s
[CV] END max_depth=30, min_samples_split=2, n_estimators=500; total time=   2.8s
[CV] END max_depth=30, min_samples_split=5, n_estimators=500; total time=   2.5s
[CV] END max_depth=30, min_samples_split=5, n_estimators=500; total time=   2.6s
[CV] END max_depth=30, min_samples_split=5, n_estimators=500; total time=   2.6s
[CV] END max_depth=30, min_samples_split=2, n_estimators=500; total time=   2.9s
[CV] END max_depth=30, min_samples_split=2, n_estimators=500; total time=   2.8s
[CV] END max_depth=30, min_samples_split=2, n_estimators=500; total time=   2.9s
[CV] END max_depth=30, min_samples_split=5, n_estimators=500; total time=   2.6s
[CV] END max_depth=30, min_samples_split=5, n_estimators=500; total time=   2.5s
[CV] END max_depth=30, min_samples_split=10, n_estimators=500; total time=   2.1s
[CV] END max_depth=30, min_samples_split=10, n_estimators=500; total time=   2.1s
[CV] END max_depth=30, min_samples_split=10, n_estimators=500; total time=   2.1s
[CV] END max_depth=30, min_samples_split=10, n_estimators=500; total time=   2.2s
[CV] END max_depth=30, min_samples_split=10, n_estimators=500; total time=   2.2s

Random Forest Best Parameters: {'max_depth': 30, 'min_samples_split': 10, 'n_estimators': 500}
Random Forest Accuracy: 0.775959723096287
Random Forest Classification Report:
              precision    recall  f1-score   support

           A       0.75      0.76      0.76       193
           B       0.81      0.93      0.87       721
           C       0.73      0.66      0.69       438
           D       0.73      0.54      0.62       237

    accuracy                           0.78      1589
   macro avg       0.76      0.72      0.73      1589
weighted avg       0.77      0.78      0.77      1589
```

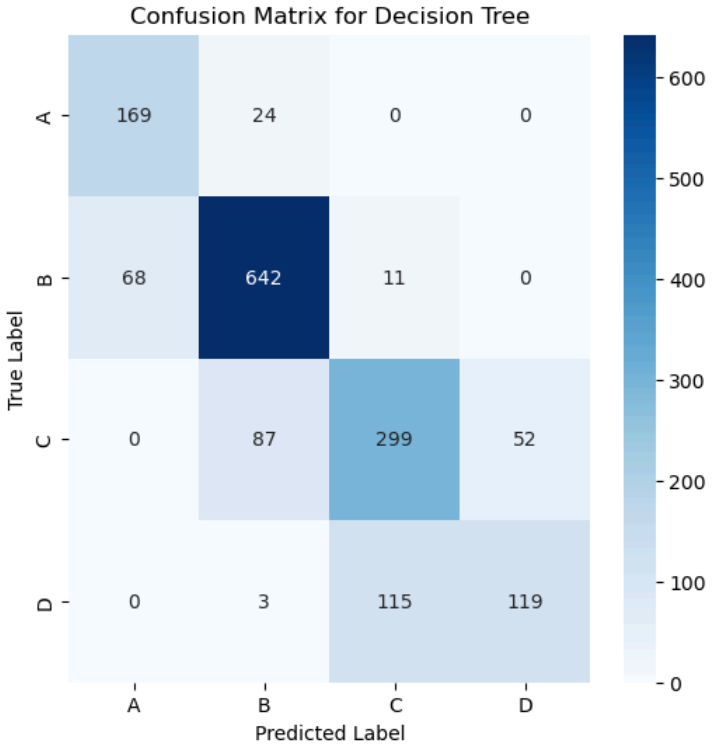
Considering time cost, training code for gradient boosting is **without grid search cross-validation**.

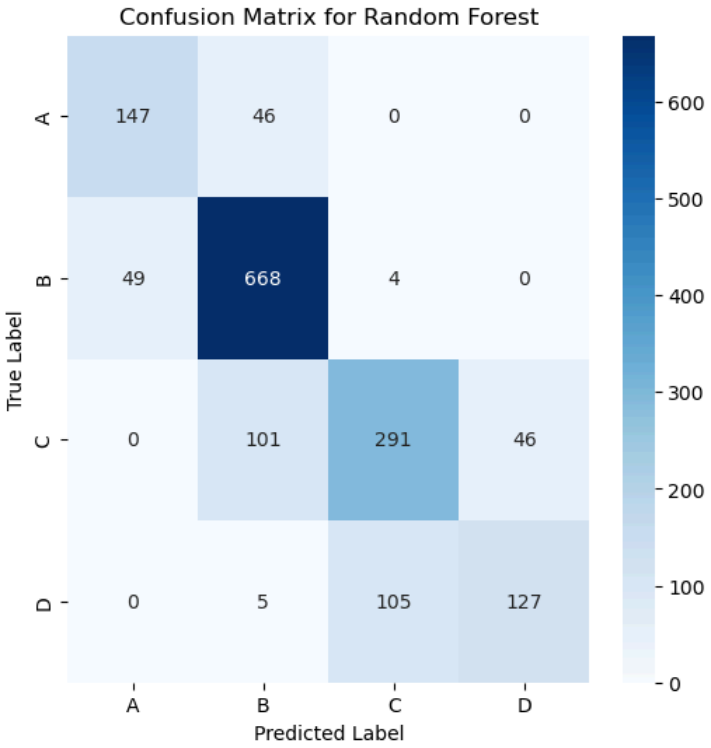
In [35]:
```python
# Pre-Selected GBM Hyperparameters, considering time cost
best_gb = GradientBoostingClassifier(
    n_estimators=200, max_depth=7, learning_rate=0.1, random_state=42
)

# Train the model
best_gb.fit(X_train, y_train)

# Make predictions
y_pred_gb = best_gb.predict(X_test)

# Evaluate performance
print("\nGradient Boosting Accuracy:", accuracy_score(y_test, y_pred_gb))
print(
    "Gradient Boosting Classification Report:\n",
    classification_report(y_test, y_pred_gb),
)
```

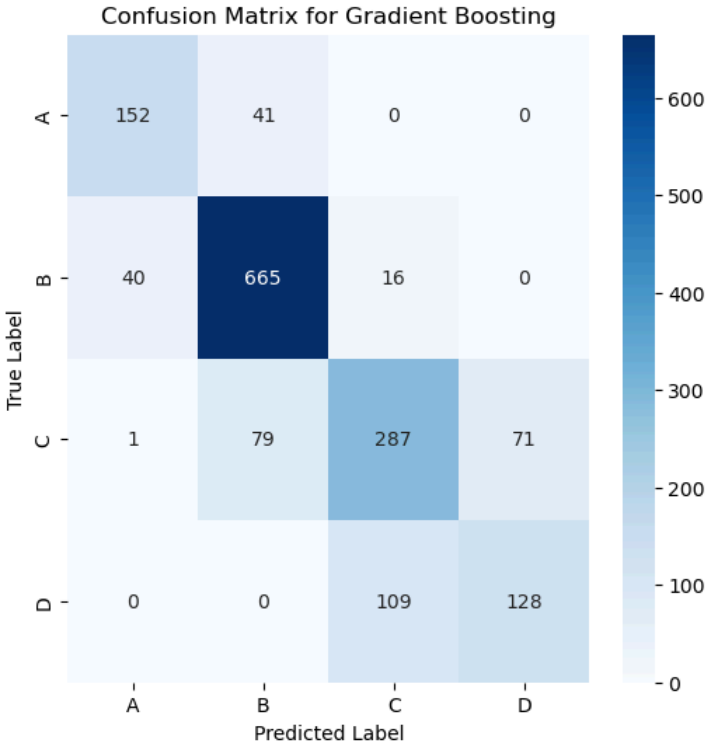
```
Gradient Boosting Accuracy: 0.775330396475771
Gradient Boosting Classification Report:
              precision    recall  f1-score   support

           A       0.79      0.79      0.79       193
           B       0.85      0.92      0.88       721
           C       0.70      0.66      0.68       438
           D       0.64      0.54      0.59       237

    accuracy                           0.78      1589
   macro avg       0.74      0.73      0.73      1589
weighted avg       0.77      0.78      0.77      1589
```

Confusion matrix for three tree models

In [36]:
```python
for model, name in zip(
    [best_dt, best_rf, best_gb], ["Decision Tree", "Random Forest", "Gradient Boosting"]
):
    cm = confusion_matrix(y_test, model.predict(X_test))
    plt.figure(figsize=(6, 6))
    sns.heatmap(
        cm,
        annot=True,
        fmt="d",
        cmap="Blues",
        xticklabels=model.classes_,
        yticklabels=model.classes_,
    )
    plt.title(f"Confusion Matrix for {name}")
    plt.xlabel("Predicted Label")
    plt.ylabel("True Label")
    plt.show()
```





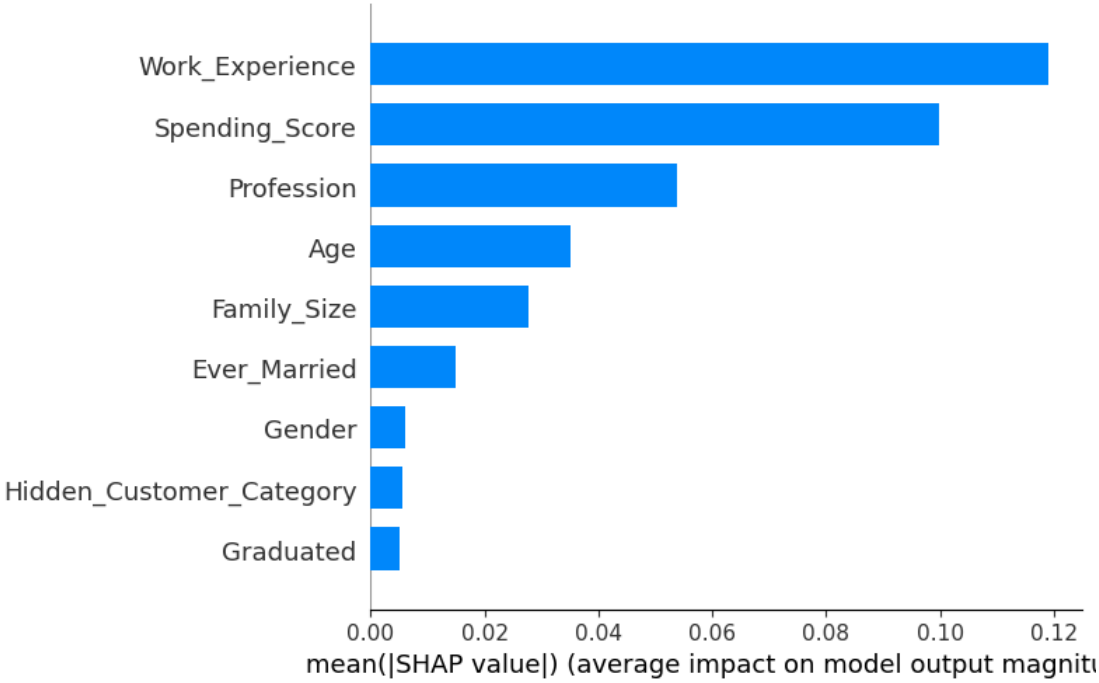


2.3 Model Explainability with SHAP & AutoML using FLAML

SHAP Value Analysis

In [37]:
```
# Sample a small subset of X_test
small_X_test = X_test.sample(n=100, random_state=42)

# Explain the Random Forest Model
explainer_rf = shap.TreeExplainer(best_rf)

shap_values_rf = explainer_rf.shap_values(small_X_test)
shap_values_mean = np.mean(np.abs(shap_values_rf), axis=2)

print("SHAP Summary Plot (Mean Across All Classes)")
shap.summary_plot(
    shap_values_mean, small_X_test, plot_type="bar", feature_names=small_X_test.columns
)
```

```
SHAP Summary Plot (Mean Across All Classes)
```



Question: Briefly explain the most influential predictors for customer segmentation, based on SHAP value.

Answer:

The most useful predictors appear to be work experience, spending score, and profession based on the contribution to the models performance uncovered by leaving that value out and measuring the marginal change in score.

FLAML AutoML for Multi-Class Classification

Train models using **FLAML AutoML**, an efficient and lightweight AutoML framework.

In [38]:
```
# Initialize FLAML AutoML
automl = AutoML()

# Train AutoML with time limit
automl.fit(X_train, y_train, task="classification", time_budget=300)

# Get predictions from the best model
y_pred_flaml = automl.predict(X_test)

# Evaluate model performance
print("FLAML AutoML Accuracy:", accuracy_score(y_test, y_pred_flaml))
print(
    "FLAML AutoML Classification Report:\n", classification_report(y_test, y_pred_flaml)
)

# Print the best model selected by FLAML
print("\nFLAML AutoML Best Model:", automl.model.estimator)
```

```
[flaml.automl.logger: 04-16 11:49:58] {1728} INFO - task = classification
[flaml.automl.logger: 04-16 11:49:58] {1739} INFO - Evaluation method: cv
[flaml.automl.logger: 04-16 11:49:58] {1838} INFO - Minimizing error metric: log_loss
[flaml.automl.logger: 04-16 11:49:58] {1955} INFO - List of ML learners in AutoML Run: ['lgbm', 'rf', 'xgboost', 'extra_tree', 'xgb_limitdepth',
'sgd', 'lrl1']
[flaml.automl.logger: 04-16 11:49:58] {1739} INFO - Evaluation method: cv
[flaml.automl.logger: 04-16 11:49:58] {1838} INFO - Minimizing error metric: log_loss
[flaml.automl.logger: 04-16 11:49:58] {1955} INFO - List of ML learners in AutoML Run: ['lgbm', 'rf', 'xgboost', 'extra_tree', 'xgb_limitdepth',
'sgd', 'lrl1']
[flaml.automl.logger: 04-16 11:49:58] {2258} INFO - iteration 0, current learner lgbm
[flaml.automl.logger: 04-16 11:49:58] {2393} INFO - Estimated sufficient time budget=1081s. Estimated necessary time budget=25s.
[flaml.automl.logger: 04-16 11:49:58] {2442} INFO -  at 0.1s,   estimator lgbm's best error=1.0504,     best estimator lgbm's best error=1.0504
[flaml.automl.logger: 04-16 11:49:58] {2258} INFO - iteration 1, current learner lgbm
[flaml.automl.logger: 04-16 11:49:58] {2442} INFO -  at 0.2s,   estimator lgbm's best error=1.0504,     best estimator lgbm's best error=1.0504
[flaml.automl.logger: 04-16 11:49:58] {2258} INFO - iteration 2, current learner lgbm
[flaml.automl.logger: 04-16 11:49:58] {2442} INFO -  at 0.3s,   estimator lgbm's best error=0.8964,     best estimator lgbm's best error=0.8964
[flaml.automl.logger: 04-16 11:49:58] {2258} INFO - iteration 3, current learner sgd
[flaml.automl.logger: 04-16 11:49:58] {2442} INFO -  at 0.8s,   estimator sgd's best error=1.2045,      best estimator lgbm's best error=0.8964
[flaml.automl.logger: 04-16 11:49:58] {2258} INFO - iteration 4, current learner lgbm
[flaml.automl.logger: 04-16 11:49:58] {2442} INFO -  at 0.9s,   estimator lgbm's best error=0.6375,     best estimator lgbm's best error=0.6375
[flaml.automl.logger: 04-16 11:49:58] {2258} INFO - iteration 5, current learner lgbm
[flaml.automl.logger: 04-16 11:49:58] {2442} INFO -  at 1.0s,   estimator lgbm's best error=0.6375,     best estimator lgbm's best error=0.6375
[flaml.automl.logger: 04-16 11:49:58] {2258} INFO - iteration 6, current learner lgbm
[flaml.automl.logger: 04-16 11:49:59] {2442} INFO -  at 1.1s,   estimator lgbm's best error=0.6028,     best estimator lgbm's best error=0.6028
[flaml.automl.logger: 04-16 11:49:59] {2258} INFO - iteration 7, current learner lgbm
[flaml.automl.logger: 04-16 11:49:59] {2442} INFO -  at 1.1s,   estimator lgbm's best error=0.6028,     best estimator lgbm's best error=0.6028
[flaml.automl.logger: 04-16 11:49:59] {2258} INFO - iteration 8, current learner lgbm
[flaml.automl.logger: 04-16 11:49:59] {2442} INFO -  at 1.2s,   estimator lgbm's best error=0.6028,     best estimator lgbm's best error=0.6028
[flaml.automl.logger: 04-16 11:49:59] {2258} INFO - iteration 9, current learner lgbm
[flaml.automl.logger: 04-16 11:49:59] {2442} INFO -  at 1.4s,   estimator lgbm's best error=0.5964,     best estimator lgbm's best error=0.5964
[flaml.automl.logger: 04-16 11:49:59] {2258} INFO - iteration 10, current learner xgboost
[flaml.automl.logger: 04-16 11:50:00] {2442} INFO -  at 2.2s,   estimator xgboost's best error=1.1927,  best estimator lgbm's best error=0.5964
[flaml.automl.logger: 04-16 11:50:00] {2258} INFO - iteration 11, current learner extra_tree
[flaml.automl.logger: 04-16 11:50:00] {2442} INFO -  at 2.4s,   estimator extra_tree's best error=1.0056,       best estimator lgbm's best error=
0.5964
[flaml.automl.logger: 04-16 11:50:00] {2258} INFO - iteration 12, current learner extra_tree
[flaml.automl.logger: 04-16 11:50:00] {2442} INFO -  at 2.6s,   estimator extra_tree's best error=0.9608,       best estimator lgbm's best error=
0.5964
[flaml.automl.logger: 04-16 11:50:00] {2258} INFO - iteration 13, current learner rf
[flaml.automl.logger: 04-16 11:50:00] {2442} INFO -  at 2.8s,   estimator rf's best error=0.9488,       best estimator lgbm's best error=0.5964
[flaml.automl.logger: 04-16 11:50:00] {2258} INFO - iteration 14, current learner rf
[flaml.automl.logger: 04-16 11:50:01] {2442} INFO -  at 3.1s,   estimator rf's best error=0.8751,       best estimator lgbm's best error=0.5964
[flaml.automl.logger: 04-16 11:50:01] {2258} INFO - iteration 15, current learner lgbm
[flaml.automl.logger: 04-16 11:50:01] {2442} INFO -  at 3.2s,   estimator lgbm's best error=0.5964,     best estimator lgbm's best error=0.5964
[flaml.automl.logger: 04-16 11:50:01] {2258} INFO - iteration 16, current learner xgboost
[flaml.automl.logger: 04-16 11:50:01] {2442} INFO -  at 3.3s,   estimator xgboost's best error=1.1927,  best estimator lgbm's best error=0.5964
[flaml.automl.logger: 04-16 11:50:01] {2258} INFO - iteration 17, current learner lgbm
[flaml.automl.logger: 04-16 11:50:02] {2442} INFO -  at 4.8s,   estimator lgbm's best error=0.5439,     best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:02] {2258} INFO - iteration 18, current learner sgd
[flaml.automl.logger: 04-16 11:50:03] {2442} INFO -  at 5.6s,   estimator sgd's best error=1.0886,      best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:03] {2258} INFO - iteration 19, current learner lgbm
[flaml.automl.logger: 04-16 11:50:04] {2442} INFO -  at 6.9s,   estimator lgbm's best error=0.5439,     best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:04] {2258} INFO - iteration 20, current learner xgboost
[flaml.automl.logger: 04-16 11:50:05] {2442} INFO -  at 7.0s,   estimator xgboost's best error=1.0183,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:05] {2258} INFO - iteration 21, current learner rf
[flaml.automl.logger: 04-16 11:50:05] {2442} INFO -  at 7.2s,   estimator rf's best error=0.8751,       best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:05] {2258} INFO - iteration 22, current learner xgboost
[flaml.automl.logger: 04-16 11:50:05] {2442} INFO -  at 7.3s,   estimator xgboost's best error=0.7628,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:05] {2258} INFO - iteration 23, current learner xgboost
[flaml.automl.logger: 04-16 11:50:05] {2442} INFO -  at 7.5s,   estimator xgboost's best error=0.7628,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:05] {2258} INFO - iteration 24, current learner rf
[flaml.automl.logger: 04-16 11:50:05] {2442} INFO -  at 7.7s,   estimator rf's best error=0.8751,       best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:05] {2258} INFO - iteration 25, current learner xgboost
[flaml.automl.logger: 04-16 11:50:05] {2442} INFO -  at 7.8s,   estimator xgboost's best error=0.7628,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:05] {2258} INFO - iteration 26, current learner sgd
[flaml.automl.logger: 04-16 11:50:06] {2442} INFO -  at 8.1s,   estimator sgd's best error=1.0886,      best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:06] {2258} INFO - iteration 27, current learner xgboost
[flaml.automl.logger: 04-16 11:50:06] {2442} INFO -  at 8.3s,   estimator xgboost's best error=0.6526,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:06] {2258} INFO - iteration 28, current learner xgboost
[flaml.automl.logger: 04-16 11:50:06] {2442} INFO -  at 8.4s,   estimator xgboost's best error=0.6168,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:06] {2258} INFO - iteration 29, current learner extra_tree
[flaml.automl.logger: 04-16 11:50:06] {2442} INFO -  at 8.6s,   estimator extra_tree's best error=0.9608,       best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:06] {2258} INFO - iteration 30, current learner xgboost
[flaml.automl.logger: 04-16 11:50:06] {2442} INFO -  at 8.8s,   estimator xgboost's best error=0.6168,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:06] {2258} INFO - iteration 31, current learner lgbm
[flaml.automl.logger: 04-16 11:50:07] {2442} INFO -  at 9.4s,   estimator lgbm's best error=0.5439,     best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:07] {2258} INFO - iteration 32, current learner lgbm
[flaml.automl.logger: 04-16 11:50:07] {2442} INFO -  at 9.9s,   estimator lgbm's best error=0.5439,     best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:07] {2258} INFO - iteration 33, current learner lgbm
[flaml.automl.logger: 04-16 11:50:09] {2442} INFO -  at 11.6s,  estimator lgbm's best error=0.5439,     best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:09] {2258} INFO - iteration 34, current learner extra_tree
[flaml.automl.logger: 04-16 11:50:09] {2442} INFO -  at 11.9s,  estimator extra_tree's best error=0.9608,       best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:09] {2258} INFO - iteration 35, current learner lgbm
[flaml.automl.logger: 04-16 11:50:11] {2442} INFO -  at 13.7s,  estimator lgbm's best error=0.5439,     best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:11] {2258} INFO - iteration 36, current learner xgboost
[flaml.automl.logger: 04-16 11:50:11] {2442} INFO -  at 14.0s,  estimator xgboost's best error=0.5684,  best estimator lgbm's best error=0.5439
```

```
[flaml.automl.logger: 04-16 11:50:11] {2258} INFO - iteration 37, current learner extra_tree
[flaml.automl.logger: 04-16 11:50:12] {2442} INFO -  at 14.2s,  estimator extra_tree's best error=0.8868,       best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:12] {2258} INFO - iteration 38, current learner xgboost
[flaml.automl.logger: 04-16 11:50:12] {2442} INFO -  at 14.3s,  estimator xgboost's best error=0.5684,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:12] {2258} INFO - iteration 39, current learner xgboost
[flaml.automl.logger: 04-16 11:50:12] {2442} INFO -  at 14.5s,  estimator xgboost's best error=0.5684,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:12] {2258} INFO - iteration 40, current learner xgboost
[flaml.automl.logger: 04-16 11:50:12] {2442} INFO -  at 14.9s,  estimator xgboost's best error=0.5684,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:12] {2258} INFO - iteration 41, current learner xgboost
[flaml.automl.logger: 04-16 11:50:13] {2442} INFO -  at 15.1s,  estimator xgboost's best error=0.5684,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:13] {2258} INFO - iteration 42, current learner extra_tree
[flaml.automl.logger: 04-16 11:50:13] {2442} INFO -  at 15.5s,  estimator extra_tree's best error=0.8868,       best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:13] {2258} INFO - iteration 43, current learner lgbm
[flaml.automl.logger: 04-16 11:50:14] {2442} INFO -  at 16.1s,  estimator lgbm's best error=0.5439,     best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:14] {2258} INFO - iteration 44, current learner xgboost
[flaml.automl.logger: 04-16 11:50:14] {2442} INFO -  at 16.6s,  estimator xgboost's best error=0.5684,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:14] {2258} INFO - iteration 45, current learner sgd
[flaml.automl.logger: 04-16 11:50:14] {2442} INFO -  at 17.0s,  estimator sgd's best error=1.0886,      best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:14] {2258} INFO - iteration 46, current learner xgboost
[flaml.automl.logger: 04-16 11:50:15] {2442} INFO -  at 17.2s,  estimator xgboost's best error=0.5684,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:15] {2258} INFO - iteration 47, current learner rf
[flaml.automl.logger: 04-16 11:50:15] {2442} INFO -  at 17.4s,  estimator rf's best error=0.8188,       best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:15] {2258} INFO - iteration 48, current learner xgboost
[flaml.automl.logger: 04-16 11:50:15] {2442} INFO -  at 17.9s,  estimator xgboost's best error=0.5585,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:15] {2258} INFO - iteration 49, current learner lgbm
[flaml.automl.logger: 04-16 11:50:17] {2442} INFO -  at 19.4s,  estimator lgbm's best error=0.5439,     best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:17] {2258} INFO - iteration 50, current learner rf
[flaml.automl.logger: 04-16 11:50:17] {2442} INFO -  at 19.6s,  estimator rf's best error=0.8188,       best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:17] {2258} INFO - iteration 51, current learner sgd
[flaml.automl.logger: 04-16 11:50:18] {2442} INFO -  at 20.2s,  estimator sgd's best error=1.0553,      best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:18] {2258} INFO - iteration 52, current learner rf
[flaml.automl.logger: 04-16 11:50:18] {2442} INFO -  at 20.4s,  estimator rf's best error=0.7647,       best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:18] {2258} INFO - iteration 53, current learner rf
[flaml.automl.logger: 04-16 11:50:18] {2442} INFO -  at 20.8s,  estimator rf's best error=0.7529,       best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:18] {2258} INFO - iteration 54, current learner lgbm
[flaml.automl.logger: 04-16 11:50:19] {2442} INFO -  at 21.5s,  estimator lgbm's best error=0.5439,     best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:19] {2258} INFO - iteration 55, current learner lgbm
[flaml.automl.logger: 04-16 11:50:19] {2442} INFO -  at 21.9s,  estimator lgbm's best error=0.5439,     best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:19] {2258} INFO - iteration 56, current learner xgboost
[flaml.automl.logger: 04-16 11:50:20] {2442} INFO -  at 22.1s,  estimator xgboost's best error=0.5585,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:20] {2258} INFO - iteration 57, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:20] {2442} INFO -  at 22.3s,  estimator xgb_limitdepth's best error=0.6280,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:20] {2258} INFO - iteration 58, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:20] {2442} INFO -  at 22.5s,  estimator xgb_limitdepth's best error=0.6280,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:20] {2258} INFO - iteration 59, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:20] {2442} INFO -  at 22.7s,  estimator xgb_limitdepth's best error=0.5737,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:20] {2258} INFO - iteration 60, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:20] {2442} INFO -  at 22.8s,  estimator xgb_limitdepth's best error=0.5737,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:20] {2258} INFO - iteration 61, current learner rf
[flaml.automl.logger: 04-16 11:50:21] {2442} INFO -  at 23.0s,  estimator rf's best error=0.7529,       best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:21] {2258} INFO - iteration 62, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:21] {2442} INFO -  at 23.2s,  estimator xgb_limitdepth's best error=0.5737,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:21] {2258} INFO - iteration 63, current learner xgboost
[flaml.automl.logger: 04-16 11:50:22] {2442} INFO -  at 24.7s,  estimator xgboost's best error=0.5564,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:22] {2258} INFO - iteration 64, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:22] {2442} INFO -  at 24.8s,  estimator xgb_limitdepth's best error=0.5737,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:22] {2258} INFO - iteration 65, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:23] {2442} INFO -  at 25.2s,  estimator xgb_limitdepth's best error=0.5737,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:23] {2258} INFO - iteration 66, current learner rf
[flaml.automl.logger: 04-16 11:50:23] {2442} INFO -  at 25.5s,  estimator rf's best error=0.7529,       best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:23] {2258} INFO - iteration 67, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:23] {2442} INFO -  at 25.8s,  estimator xgb_limitdepth's best error=0.5737,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:23] {2258} INFO - iteration 68, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:24] {2442} INFO -  at 26.0s,  estimator xgb_limitdepth's best error=0.5737,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:24] {2258} INFO - iteration 69, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:24] {2442} INFO -  at 26.2s,  estimator xgb_limitdepth's best error=0.5737,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:24] {2258} INFO - iteration 70, current learner lgbm
[flaml.automl.logger: 04-16 11:50:27] {2442} INFO -  at 29.0s,  estimator lgbm's best error=0.5439,     best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:27] {2258} INFO - iteration 71, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:27] {2442} INFO -  at 29.2s,  estimator xgb_limitdepth's best error=0.5737,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:27] {2258} INFO - iteration 72, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:27] {2442} INFO -  at 29.3s,  estimator xgb_limitdepth's best error=0.5737,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:27] {2258} INFO - iteration 73, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:27] {2442} INFO -  at 29.6s,  estimator xgb_limitdepth's best error=0.5581,   best estimator lgbm's best error=
```

```
0.5439
[flaml.automl.logger: 04-16 11:50:27] {2258} INFO - iteration 74, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:27] {2442} INFO -  at 29.8s,  estimator xgb_limitdepth's best error=0.5581,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:27] {2258} INFO - iteration 75, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:28] {2442} INFO -  at 30.4s,  estimator xgb_limitdepth's best error=0.5581,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:28] {2258} INFO - iteration 76, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:28] {2442} INFO -  at 30.5s,  estimator xgb_limitdepth's best error=0.5581,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:28] {2258} INFO - iteration 77, current learner lgbm
[flaml.automl.logger: 04-16 11:50:29] {2442} INFO -  at 31.5s,  estimator lgbm's best error=0.5439,     best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:29] {2258} INFO - iteration 78, current learner lgbm
[flaml.automl.logger: 04-16 11:50:30] {2442} INFO -  at 32.4s,  estimator lgbm's best error=0.5439,     best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:30] {2258} INFO - iteration 79, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:30] {2442} INFO -  at 32.8s,  estimator xgb_limitdepth's best error=0.5581,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:30] {2258} INFO - iteration 80, current learner extra_tree
[flaml.automl.logger: 04-16 11:50:31] {2442} INFO -  at 33.0s,  estimator extra_tree's best error=0.8412,       best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:31] {2258} INFO - iteration 81, current learner rf
[flaml.automl.logger: 04-16 11:50:31] {2442} INFO -  at 33.4s,  estimator rf's best error=0.6440,       best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:31] {2258} INFO - iteration 82, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:31] {2442} INFO -  at 33.6s,  estimator xgb_limitdepth's best error=0.5581,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:31] {2258} INFO - iteration 83, current learner rf
[flaml.automl.logger: 04-16 11:50:31] {2442} INFO -  at 33.8s,  estimator rf's best error=0.6440,       best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:31] {2258} INFO - iteration 84, current learner extra_tree
[flaml.automl.logger: 04-16 11:50:32] {2442} INFO -  at 34.1s,  estimator extra_tree's best error=0.8409,       best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:32] {2258} INFO - iteration 85, current learner xgboost
[flaml.automl.logger: 04-16 11:50:32] {2442} INFO -  at 34.5s,  estimator xgboost's best error=0.5564,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:32] {2258} INFO - iteration 86, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:33] {2442} INFO -  at 35.0s,  estimator xgb_limitdepth's best error=0.5581,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:33] {2258} INFO - iteration 87, current learner rf
[flaml.automl.logger: 04-16 11:50:33] {2442} INFO -  at 35.4s,  estimator rf's best error=0.5867,       best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:33] {2258} INFO - iteration 88, current learner rf
[flaml.automl.logger: 04-16 11:50:33] {2442} INFO -  at 35.7s,  estimator rf's best error=0.5867,       best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:33] {2258} INFO - iteration 89, current learner rf
[flaml.automl.logger: 04-16 11:50:34] {2442} INFO -  at 36.5s,  estimator rf's best error=0.5867,       best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:34] {2258} INFO - iteration 90, current learner rf
[flaml.automl.logger: 04-16 11:50:34] {2442} INFO -  at 36.8s,  estimator rf's best error=0.5711,       best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:34] {2258} INFO - iteration 91, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:34] {2442} INFO -  at 37.0s,  estimator xgb_limitdepth's best error=0.5581,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:34] {2258} INFO - iteration 92, current learner rf
[flaml.automl.logger: 04-16 11:50:35] {2442} INFO -  at 37.4s,  estimator rf's best error=0.5711,       best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:35] {2258} INFO - iteration 93, current learner sgd
[flaml.automl.logger: 04-16 11:50:35] {2442} INFO -  at 37.6s,  estimator sgd's best error=1.0553,      best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:35] {2258} INFO - iteration 94, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:36] {2442} INFO -  at 38.3s,  estimator xgb_limitdepth's best error=0.5581,   best estimator lgbm's best error=
0.5439
[flaml.automl.logger: 04-16 11:50:36] {2258} INFO - iteration 95, current learner xgboost
[flaml.automl.logger: 04-16 11:50:42] {2442} INFO -  at 44.2s,  estimator xgboost's best error=0.5564,  best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:42] {2258} INFO - iteration 96, current learner lgbm
[flaml.automl.logger: 04-16 11:50:42] {2442} INFO -  at 44.8s,  estimator lgbm's best error=0.5439,     best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:42] {2258} INFO - iteration 97, current learner sgd
[flaml.automl.logger: 04-16 11:50:43] {2442} INFO -  at 45.2s,  estimator sgd's best error=1.0486,      best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:43] {2258} INFO - iteration 98, current learner rf
[flaml.automl.logger: 04-16 11:50:43] {2442} INFO -  at 45.5s,  estimator rf's best error=0.5711,       best estimator lgbm's best error=0.5439
[flaml.automl.logger: 04-16 11:50:43] {2258} INFO - iteration 99, current learner rf
[flaml.automl.logger: 04-16 11:50:44] {2442} INFO -  at 46.0s,  estimator rf's best error=0.5220,       best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:44] {2258} INFO - iteration 100, current learner rf
[flaml.automl.logger: 04-16 11:50:44] {2442} INFO -  at 46.6s,  estimator rf's best error=0.5220,       best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:44] {2258} INFO - iteration 101, current learner rf
[flaml.automl.logger: 04-16 11:50:45] {2442} INFO -  at 47.1s,  estimator rf's best error=0.5220,       best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:45] {2258} INFO - iteration 102, current learner rf
[flaml.automl.logger: 04-16 11:50:45] {2442} INFO -  at 47.6s,  estimator rf's best error=0.5220,       best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:45] {2258} INFO - iteration 103, current learner rf
[flaml.automl.logger: 04-16 11:50:46] {2442} INFO -  at 48.0s,  estimator rf's best error=0.5220,       best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:46] {2258} INFO - iteration 104, current learner lgbm
[flaml.automl.logger: 04-16 11:50:47] {2442} INFO -  at 49.4s,  estimator lgbm's best error=0.5439,     best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:47] {2258} INFO - iteration 105, current learner rf
[flaml.automl.logger: 04-16 11:50:48] {2442} INFO -  at 50.4s,  estimator rf's best error=0.5220,       best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:48] {2258} INFO - iteration 106, current learner rf
[flaml.automl.logger: 04-16 11:50:48] {2442} INFO -  at 50.7s,  estimator rf's best error=0.5220,       best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:48] {2258} INFO - iteration 107, current learner rf
[flaml.automl.logger: 04-16 11:50:49] {2442} INFO -  at 51.4s,  estimator rf's best error=0.5220,       best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:49] {2258} INFO - iteration 108, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:49] {2442} INFO -  at 51.6s,  estimator xgb_limitdepth's best error=0.5581,   best estimator rf's best error=0.5
220
[flaml.automl.logger: 04-16 11:50:49] {2258} INFO - iteration 109, current learner rf
[flaml.automl.logger: 04-16 11:50:50] {2442} INFO -  at 52.0s,  estimator rf's best error=0.5220,       best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:50] {2258} INFO - iteration 110, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:50] {2442} INFO -  at 52.4s,  estimator xgb_limitdepth's best error=0.5581,   best estimator rf's best error=0.5
220
[flaml.automl.logger: 04-16 11:50:50] {2258} INFO - iteration 111, current learner rf
```

```
[flaml.automl.logger: 04-16 11:50:51] {2442} INFO -  at 53.1s,  estimator rf's best error=0.5220,       best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:51] {2258} INFO - iteration 112, current learner lgbm
[flaml.automl.logger: 04-16 11:50:53] {2442} INFO -  at 55.1s,  estimator lgbm's best error=0.5439,     best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:53] {2258} INFO - iteration 113, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:53] {2442} INFO -  at 55.3s,  estimator xgb_limitdepth's best error=0.5581,   best estimator rf's best error=0.5
220
[flaml.automl.logger: 04-16 11:50:53] {2258} INFO - iteration 114, current learner rf
[flaml.automl.logger: 04-16 11:50:53] {2442} INFO -  at 55.8s,  estimator rf's best error=0.5220,       best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:53] {2258} INFO - iteration 115, current learner sgd
[flaml.automl.logger: 04-16 11:50:54] {2442} INFO -  at 56.4s,  estimator sgd's best error=1.0486,      best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:54] {2258} INFO - iteration 116, current learner lgbm
[flaml.automl.logger: 04-16 11:50:54] {2442} INFO -  at 56.9s,  estimator lgbm's best error=0.5439,     best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:54] {2258} INFO - iteration 117, current learner lgbm
[flaml.automl.logger: 04-16 11:50:57] {2442} INFO -  at 59.6s,  estimator lgbm's best error=0.5439,     best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:57] {2258} INFO - iteration 118, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:50:58] {2442} INFO -  at 60.2s,  estimator xgb_limitdepth's best error=0.5581,   best estimator rf's best error=0.5
220
[flaml.automl.logger: 04-16 11:50:58] {2258} INFO - iteration 119, current learner rf
[flaml.automl.logger: 04-16 11:50:58] {2442} INFO -  at 60.7s,  estimator rf's best error=0.5220,       best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:58] {2258} INFO - iteration 120, current learner lgbm
[flaml.automl.logger: 04-16 11:50:58] {2442} INFO -  at 61.0s,  estimator lgbm's best error=0.5439,     best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:58] {2258} INFO - iteration 121, current learner rf
[flaml.automl.logger: 04-16 11:50:59] {2442} INFO -  at 61.5s,  estimator rf's best error=0.5220,       best estimator rf's best error=0.5220
[flaml.automl.logger: 04-16 11:50:59] {2258} INFO - iteration 122, current learner rf
[flaml.automl.logger: 04-16 11:51:00] {2442} INFO -  at 62.3s,  estimator rf's best error=0.5216,       best estimator rf's best error=0.5216
[flaml.automl.logger: 04-16 11:51:00] {2258} INFO - iteration 123, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:51:00] {2442} INFO -  at 62.4s,  estimator xgb_limitdepth's best error=0.5581,   best estimator rf's best error=0.5
216
[flaml.automl.logger: 04-16 11:51:00] {2258} INFO - iteration 124, current learner rf
[flaml.automl.logger: 04-16 11:51:00] {2442} INFO -  at 62.9s,  estimator rf's best error=0.5216,       best estimator rf's best error=0.5216
[flaml.automl.logger: 04-16 11:51:00] {2258} INFO - iteration 125, current learner lgbm
[flaml.automl.logger: 04-16 11:51:01] {2442} INFO -  at 63.2s,  estimator lgbm's best error=0.5439,     best estimator rf's best error=0.5216
[flaml.automl.logger: 04-16 11:51:01] {2258} INFO - iteration 126, current learner lgbm
[flaml.automl.logger: 04-16 11:51:03] {2442} INFO -  at 65.9s,  estimator lgbm's best error=0.5439,     best estimator rf's best error=0.5216
[flaml.automl.logger: 04-16 11:51:03] {2258} INFO - iteration 127, current learner lgbm
[flaml.automl.logger: 04-16 11:51:04] {2442} INFO -  at 66.7s,  estimator lgbm's best error=0.5439,     best estimator rf's best error=0.5216
[flaml.automl.logger: 04-16 11:51:04] {2258} INFO - iteration 128, current learner lgbm
[flaml.automl.logger: 04-16 11:51:05] {2442} INFO -  at 67.8s,  estimator lgbm's best error=0.5393,     best estimator rf's best error=0.5216
[flaml.automl.logger: 04-16 11:51:05] {2258} INFO - iteration 129, current learner rf
[flaml.automl.logger: 04-16 11:51:06] {2442} INFO -  at 68.4s,  estimator rf's best error=0.5216,       best estimator rf's best error=0.5216
[flaml.automl.logger: 04-16 11:51:06] {2258} INFO - iteration 130, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:51:07] {2442} INFO -  at 69.1s,  estimator xgb_limitdepth's best error=0.5581,   best estimator rf's best error=0.5
216
[flaml.automl.logger: 04-16 11:51:07] {2258} INFO - iteration 131, current learner rf
[flaml.automl.logger: 04-16 11:51:08] {2442} INFO -  at 70.1s,  estimator rf's best error=0.5216,       best estimator rf's best error=0.5216
[flaml.automl.logger: 04-16 11:51:08] {2258} INFO - iteration 132, current learner rf
[flaml.automl.logger: 04-16 11:51:09] {2442} INFO -  at 71.9s,  estimator rf's best error=0.5135,       best estimator rf's best error=0.5135
[flaml.automl.logger: 04-16 11:51:09] {2258} INFO - iteration 133, current learner rf
[flaml.automl.logger: 04-16 11:51:10] {2442} INFO -  at 72.7s,  estimator rf's best error=0.5135,       best estimator rf's best error=0.5135
[flaml.automl.logger: 04-16 11:51:10] {2258} INFO - iteration 134, current learner rf
[flaml.automl.logger: 04-16 11:51:11] {2442} INFO -  at 73.9s,  estimator rf's best error=0.5135,       best estimator rf's best error=0.5135
[flaml.automl.logger: 04-16 11:51:11] {2258} INFO - iteration 135, current learner rf
[flaml.automl.logger: 04-16 11:51:14] {2442} INFO -  at 76.8s,  estimator rf's best error=0.5135,       best estimator rf's best error=0.5135
[flaml.automl.logger: 04-16 11:51:14] {2258} INFO - iteration 136, current learner rf
[flaml.automl.logger: 04-16 11:51:18] {2442} INFO -  at 80.5s,  estimator rf's best error=0.5135,       best estimator rf's best error=0.5135
[flaml.automl.logger: 04-16 11:51:18] {2258} INFO - iteration 137, current learner sgd
[flaml.automl.logger: 04-16 11:51:18] {2442} INFO -  at 80.6s,  estimator sgd's best error=1.0486,      best estimator rf's best error=0.5135
[flaml.automl.logger: 04-16 11:51:18] {2258} INFO - iteration 138, current learner rf
[flaml.automl.logger: 04-16 11:51:19] {2442} INFO -  at 81.6s,  estimator rf's best error=0.5135,       best estimator rf's best error=0.5135
[flaml.automl.logger: 04-16 11:51:19] {2258} INFO - iteration 139, current learner lgbm
[flaml.automl.logger: 04-16 11:51:21] {2442} INFO -  at 83.0s,  estimator lgbm's best error=0.5297,     best estimator rf's best error=0.5135
[flaml.automl.logger: 04-16 11:51:21] {2258} INFO - iteration 140, current learner rf
[flaml.automl.logger: 04-16 11:51:23] {2442} INFO -  at 85.7s,  estimator rf's best error=0.5135,       best estimator rf's best error=0.5135
[flaml.automl.logger: 04-16 11:51:23] {2258} INFO - iteration 141, current learner lgbm
[flaml.automl.logger: 04-16 11:51:24] {2442} INFO -  at 86.8s,  estimator lgbm's best error=0.5297,     best estimator rf's best error=0.5135
[flaml.automl.logger: 04-16 11:51:24] {2258} INFO - iteration 142, current learner rf
[flaml.automl.logger: 04-16 11:51:26] {2442} INFO -  at 88.2s,  estimator rf's best error=0.5135,       best estimator rf's best error=0.5135
[flaml.automl.logger: 04-16 11:51:26] {2258} INFO - iteration 143, current learner lgbm
[flaml.automl.logger: 04-16 11:51:32] {2442} INFO -  at 94.6s,  estimator lgbm's best error=0.5297,     best estimator rf's best error=0.5135
[flaml.automl.logger: 04-16 11:51:32] {2258} INFO - iteration 144, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:51:32] {2442} INFO -  at 94.9s,  estimator xgb_limitdepth's best error=0.5581,   best estimator rf's best error=0.5
135
[flaml.automl.logger: 04-16 11:51:32] {2258} INFO - iteration 145, current learner rf
[flaml.automl.logger: 04-16 11:51:36] {2442} INFO -  at 98.1s,  estimator rf's best error=0.5135,       best estimator rf's best error=0.5135
[flaml.automl.logger: 04-16 11:51:36] {2258} INFO - iteration 146, current learner rf
[flaml.automl.logger: 04-16 11:51:37] {2442} INFO -  at 99.2s,  estimator rf's best error=0.5135,       best estimator rf's best error=0.5135
[flaml.automl.logger: 04-16 11:51:37] {2258} INFO - iteration 147, current learner lrl1
[flaml.automl.logger: 04-16 11:51:38] {2442} INFO -  at 100.8s, estimator lrl1's best error=0.9844,     best estimator rf's best error=0.5135
[flaml.automl.logger: 04-16 11:51:38] {2258} INFO - iteration 148, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:51:39] {2442} INFO -  at 101.1s, estimator xgb_limitdepth's best error=0.5581,   best estimator rf's best error=0.5
135
[flaml.automl.logger: 04-16 11:51:39] {2258} INFO - iteration 149, current learner rf
[flaml.automl.logger: 04-16 11:51:41] {2442} INFO -  at 103.5s, estimator rf's best error=0.5135,       best estimator rf's best error=0.5135
[flaml.automl.logger: 04-16 11:51:41] {2258} INFO - iteration 150, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:51:41] {2442} INFO -  at 103.7s, estimator xgb_limitdepth's best error=0.5581,   best estimator rf's best error=0.5
135
[flaml.automl.logger: 04-16 11:51:41] {2258} INFO - iteration 151, current learner lgbm
[flaml.automl.logger: 04-16 11:51:42] {2442} INFO -  at 104.1s, estimator lgbm's best error=0.5297,     best estimator rf's best error=0.5135
```

```
[flaml.automl.logger: 04-16 11:51:42] {2258} INFO - iteration 152, current learner rf
[flaml.automl.logger: 04-16 11:51:43] {2442} INFO -  at 105.5s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:51:43] {2258} INFO - iteration 153, current learner lrl1
[flaml.automl.logger: 04-16 11:51:45] {2442} INFO -  at 107.1s, estimator lrl1's best error=0.9844,     best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:51:45] {2258} INFO - iteration 154, current learner xgboost
[flaml.automl.logger: 04-16 11:51:45] {2442} INFO -  at 107.9s, estimator xgboost's best error=0.5564,  best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:51:45] {2258} INFO - iteration 155, current learner rf
[flaml.automl.logger: 04-16 11:51:47] {2442} INFO -  at 109.0s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:51:47] {2258} INFO - iteration 156, current learner lgbm
[flaml.automl.logger: 04-16 11:51:49] {2442} INFO -  at 111.6s, estimator lgbm's best error=0.5297,     best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:51:49] {2258} INFO - iteration 157, current learner lgbm
[flaml.automl.logger: 04-16 11:51:50] {2442} INFO -  at 112.4s, estimator lgbm's best error=0.5258,     best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:51:50] {2258} INFO - iteration 158, current learner sgd
[flaml.automl.logger: 04-16 11:51:51] {2442} INFO -  at 113.2s, estimator sgd's best error=1.0486,      best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:51:51] {2258} INFO - iteration 159, current learner rf
[flaml.automl.logger: 04-16 11:51:53] {2442} INFO -  at 115.0s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:51:53] {2258} INFO - iteration 160, current learner rf
[flaml.automl.logger: 04-16 11:51:54] {2442} INFO -  at 116.2s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:51:54] {2258} INFO - iteration 161, current learner rf
[flaml.automl.logger: 04-16 11:51:55] {2442} INFO -  at 118.0s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:51:55] {2258} INFO - iteration 162, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:51:56] {2442} INFO -  at 118.8s, estimator xgb_limitdepth's best error=0.5531,   best estimator rf's best error=0.5
134
[flaml.automl.logger: 04-16 11:51:56] {2258} INFO - iteration 163, current learner rf
[flaml.automl.logger: 04-16 11:51:59] {2442} INFO -  at 121.7s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:51:59] {2258} INFO - iteration 164, current learner rf
[flaml.automl.logger: 04-16 11:52:00] {2442} INFO -  at 122.5s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:00] {2258} INFO - iteration 165, current learner sgd
[flaml.automl.logger: 04-16 11:52:00] {2442} INFO -  at 122.6s, estimator sgd's best error=1.0486,      best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:00] {2258} INFO - iteration 166, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:52:01] {2442} INFO -  at 123.4s, estimator xgb_limitdepth's best error=0.5531,   best estimator rf's best error=0.5
134
[flaml.automl.logger: 04-16 11:52:01] {2258} INFO - iteration 167, current learner rf
[flaml.automl.logger: 04-16 11:52:03] {2442} INFO -  at 125.9s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:03] {2258} INFO - iteration 168, current learner rf
[flaml.automl.logger: 04-16 11:52:04] {2442} INFO -  at 126.7s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:04] {2258} INFO - iteration 169, current learner rf
[flaml.automl.logger: 04-16 11:52:05] {2442} INFO -  at 127.5s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:05] {2258} INFO - iteration 170, current learner extra_tree
[flaml.automl.logger: 04-16 11:52:05] {2442} INFO -  at 127.7s, estimator extra_tree's best error=0.8409,       best estimator rf's best error=0.5
134
[flaml.automl.logger: 04-16 11:52:05] {2258} INFO - iteration 171, current learner lgbm
[flaml.automl.logger: 04-16 11:52:09] {2442} INFO -  at 131.9s, estimator lgbm's best error=0.5258,     best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:09] {2258} INFO - iteration 172, current learner lgbm
[flaml.automl.logger: 04-16 11:52:10] {2442} INFO -  at 132.1s, estimator lgbm's best error=0.5258,     best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:10] {2258} INFO - iteration 173, current learner lgbm
[flaml.automl.logger: 04-16 11:52:10] {2442} INFO -  at 132.3s, estimator lgbm's best error=0.5258,     best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:10] {2258} INFO - iteration 174, current learner rf
[flaml.automl.logger: 04-16 11:52:13] {2442} INFO -  at 135.4s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:13] {2258} INFO - iteration 175, current learner rf
[flaml.automl.logger: 04-16 11:52:14] {2442} INFO -  at 136.5s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:14] {2258} INFO - iteration 176, current learner rf
[flaml.automl.logger: 04-16 11:52:16] {2442} INFO -  at 138.4s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:16] {2258} INFO - iteration 177, current learner rf
[flaml.automl.logger: 04-16 11:52:17] {2442} INFO -  at 139.4s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:17] {2258} INFO - iteration 178, current learner sgd
[flaml.automl.logger: 04-16 11:52:17] {2442} INFO -  at 139.6s, estimator sgd's best error=1.0486,      best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:17] {2258} INFO - iteration 179, current learner rf
[flaml.automl.logger: 04-16 11:52:19] {2442} INFO -  at 141.7s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:19] {2258} INFO - iteration 180, current learner rf
[flaml.automl.logger: 04-16 11:52:20] {2442} INFO -  at 142.6s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:20] {2258} INFO - iteration 181, current learner rf
[flaml.automl.logger: 04-16 11:52:23] {2442} INFO -  at 145.1s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:23] {2258} INFO - iteration 182, current learner rf
[flaml.automl.logger: 04-16 11:52:24] {2442} INFO -  at 146.0s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:24] {2258} INFO - iteration 183, current learner rf
[flaml.automl.logger: 04-16 11:52:26] {2442} INFO -  at 148.2s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:26] {2258} INFO - iteration 184, current learner rf
[flaml.automl.logger: 04-16 11:52:27] {2442} INFO -  at 149.3s, estimator rf's best error=0.5134,       best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:27] {2258} INFO - iteration 185, current learner lgbm
[flaml.automl.logger: 04-16 11:52:30] {2442} INFO -  at 152.7s, estimator lgbm's best error=0.5258,     best estimator rf's best error=0.5134
[flaml.automl.logger: 04-16 11:52:30] {2258} INFO - iteration 186, current learner rf
[flaml.automl.logger: 04-16 11:52:32] {2442} INFO -  at 154.6s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:52:32] {2258} INFO - iteration 187, current learner rf
[flaml.automl.logger: 04-16 11:52:33] {2442} INFO -  at 155.6s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:52:33] {2258} INFO - iteration 188, current learner rf
[flaml.automl.logger: 04-16 11:52:37] {2442} INFO -  at 159.6s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:52:37] {2258} INFO - iteration 189, current learner xgboost
[flaml.automl.logger: 04-16 11:52:40] {2442} INFO -  at 162.0s, estimator xgboost's best error=0.5564,  best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:52:40] {2258} INFO - iteration 190, current learner lgbm
[flaml.automl.logger: 04-16 11:52:42] {2442} INFO -  at 164.4s, estimator lgbm's best error=0.5258,     best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:52:42] {2258} INFO - iteration 191, current learner rf
[flaml.automl.logger: 04-16 11:52:47] {2442} INFO -  at 169.6s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:52:47] {2258} INFO - iteration 192, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:52:48] {2442} INFO -  at 170.4s, estimator xgb_limitdepth's best error=0.5495,   best estimator rf's best error=0.5
115
[flaml.automl.logger: 04-16 11:52:48] {2258} INFO - iteration 193, current learner rf
[flaml.automl.logger: 04-16 11:52:49] {2442} INFO -  at 171.3s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
```

```
[flaml.automl.logger: 04-16 11:52:49] {2258} INFO - iteration 194, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:52:50] {2442} INFO -  at 172.7s, estimator xgb_limitdepth's best error=0.5468,   best estimator rf's best error=0.5
115
[flaml.automl.logger: 04-16 11:52:50] {2258} INFO - iteration 195, current learner rf
[flaml.automl.logger: 04-16 11:52:51] {2442} INFO -  at 173.8s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:52:51] {2258} INFO - iteration 196, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:52:52] {2442} INFO -  at 174.6s, estimator xgb_limitdepth's best error=0.5468,   best estimator rf's best error=0.5
115
[flaml.automl.logger: 04-16 11:52:52] {2258} INFO - iteration 197, current learner rf
[flaml.automl.logger: 04-16 11:52:56] {2442} INFO -  at 178.5s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:52:56] {2258} INFO - iteration 198, current learner lgbm
[flaml.automl.logger: 04-16 11:52:56] {2442} INFO -  at 178.8s, estimator lgbm's best error=0.5258,     best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:52:56] {2258} INFO - iteration 199, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:53:02] {2442} INFO -  at 184.9s, estimator xgb_limitdepth's best error=0.5468,   best estimator rf's best error=0.5
115
[flaml.automl.logger: 04-16 11:53:02] {2258} INFO - iteration 200, current learner rf
[flaml.automl.logger: 04-16 11:53:06] {2442} INFO -  at 188.0s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:06] {2258} INFO - iteration 201, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:53:06] {2442} INFO -  at 188.5s, estimator xgb_limitdepth's best error=0.5468,   best estimator rf's best error=0.5
115
[flaml.automl.logger: 04-16 11:53:06] {2258} INFO - iteration 202, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:53:07] {2442} INFO -  at 189.8s, estimator xgb_limitdepth's best error=0.5468,   best estimator rf's best error=0.5
115
[flaml.automl.logger: 04-16 11:53:07] {2258} INFO - iteration 203, current learner lgbm
[flaml.automl.logger: 04-16 11:53:08] {2442} INFO -  at 190.0s, estimator lgbm's best error=0.5258,     best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:08] {2258} INFO - iteration 204, current learner rf
[flaml.automl.logger: 04-16 11:53:09] {2442} INFO -  at 191.7s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:09] {2258} INFO - iteration 205, current learner rf
[flaml.automl.logger: 04-16 11:53:12] {2442} INFO -  at 194.6s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:12] {2258} INFO - iteration 206, current learner sgd
[flaml.automl.logger: 04-16 11:53:12] {2442} INFO -  at 194.8s, estimator sgd's best error=1.0486,      best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:12] {2258} INFO - iteration 207, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:53:14] {2442} INFO -  at 196.3s, estimator xgb_limitdepth's best error=0.5468,   best estimator rf's best error=0.5
115
[flaml.automl.logger: 04-16 11:53:14] {2258} INFO - iteration 208, current learner rf
[flaml.automl.logger: 04-16 11:53:15] {2442} INFO -  at 197.7s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:15] {2258} INFO - iteration 209, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:53:17] {2442} INFO -  at 199.6s, estimator xgb_limitdepth's best error=0.5468,   best estimator rf's best error=0.5
115
[flaml.automl.logger: 04-16 11:53:17] {2258} INFO - iteration 210, current learner rf
[flaml.automl.logger: 04-16 11:53:20] {2442} INFO -  at 202.8s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:20] {2258} INFO - iteration 211, current learner rf
[flaml.automl.logger: 04-16 11:53:22] {2442} INFO -  at 204.0s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:22] {2258} INFO - iteration 212, current learner rf
[flaml.automl.logger: 04-16 11:53:25] {2442} INFO -  at 207.5s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:25] {2258} INFO - iteration 213, current learner rf
[flaml.automl.logger: 04-16 11:53:26] {2442} INFO -  at 208.6s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:26] {2258} INFO - iteration 214, current learner rf
[flaml.automl.logger: 04-16 11:53:27] {2442} INFO -  at 209.6s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:27] {2258} INFO - iteration 215, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:53:28] {2442} INFO -  at 210.7s, estimator xgb_limitdepth's best error=0.5468,   best estimator rf's best error=0.5
115
[flaml.automl.logger: 04-16 11:53:28] {2258} INFO - iteration 216, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:53:29] {2442} INFO -  at 211.4s, estimator xgb_limitdepth's best error=0.5468,   best estimator rf's best error=0.5
115
[flaml.automl.logger: 04-16 11:53:29] {2258} INFO - iteration 217, current learner rf
[flaml.automl.logger: 04-16 11:53:33] {2442} INFO -  at 215.6s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:33] {2258} INFO - iteration 218, current learner sgd
[flaml.automl.logger: 04-16 11:53:34] {2442} INFO -  at 216.2s, estimator sgd's best error=1.0486,      best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:34] {2258} INFO - iteration 219, current learner rf
[flaml.automl.logger: 04-16 11:53:35] {2442} INFO -  at 217.1s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:35] {2258} INFO - iteration 220, current learner lgbm
[flaml.automl.logger: 04-16 11:53:39] {2442} INFO -  at 221.3s, estimator lgbm's best error=0.5258,     best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:39] {2258} INFO - iteration 221, current learner lgbm
[flaml.automl.logger: 04-16 11:53:39] {2442} INFO -  at 221.5s, estimator lgbm's best error=0.5258,     best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:39] {2258} INFO - iteration 222, current learner rf
[flaml.automl.logger: 04-16 11:53:44] {2442} INFO -  at 226.2s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:44] {2258} INFO - iteration 223, current learner lgbm
[flaml.automl.logger: 04-16 11:53:48] {2442} INFO -  at 230.3s, estimator lgbm's best error=0.5258,     best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:48] {2258} INFO - iteration 224, current learner sgd
[flaml.automl.logger: 04-16 11:53:48] {2442} INFO -  at 230.5s, estimator sgd's best error=1.0486,      best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:48] {2258} INFO - iteration 225, current learner xgboost
[flaml.automl.logger: 04-16 11:53:49] {2442} INFO -  at 231.8s, estimator xgboost's best error=0.5564,  best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:49] {2258} INFO - iteration 226, current learner lgbm
[flaml.automl.logger: 04-16 11:53:49] {2442} INFO -  at 231.9s, estimator lgbm's best error=0.5258,     best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:49] {2258} INFO - iteration 227, current learner rf
[flaml.automl.logger: 04-16 11:53:53] {2442} INFO -  at 235.0s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:53] {2258} INFO - iteration 228, current learner rf
[flaml.automl.logger: 04-16 11:53:54] {2442} INFO -  at 236.3s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:53:54] {2258} INFO - iteration 229, current learner lgbm
[flaml.automl.logger: 04-16 11:54:01] {2442} INFO -  at 243.2s, estimator lgbm's best error=0.5258,     best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:01] {2258} INFO - iteration 230, current learner xgboost
[flaml.automl.logger: 04-16 11:54:03] {2442} INFO -  at 245.0s, estimator xgboost's best error=0.5564,  best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:03] {2258} INFO - iteration 231, current learner rf
[flaml.automl.logger: 04-16 11:54:04] {2442} INFO -  at 247.0s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:04] {2258} INFO - iteration 232, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:54:07] {2442} INFO -  at 249.9s, estimator xgb_limitdepth's best error=0.5468,   best estimator rf's best error=0.5
115
```

```
[flaml.automl.logger: 04-16 11:54:07] {2258} INFO - iteration 233, current learner rf
[flaml.automl.logger: 04-16 11:54:09] {2442} INFO -  at 251.9s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:09] {2258} INFO - iteration 234, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:54:11] {2442} INFO -  at 253.5s, estimator xgb_limitdepth's best error=0.5468,   best estimator rf's best error=0.5
115
[flaml.automl.logger: 04-16 11:54:11] {2258} INFO - iteration 235, current learner extra_tree
[flaml.automl.logger: 04-16 11:54:11] {2442} INFO -  at 253.8s, estimator extra_tree's best error=0.8409,       best estimator rf's best error=0.5
115
[flaml.automl.logger: 04-16 11:54:11] {2258} INFO - iteration 236, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:54:13] {2442} INFO -  at 255.2s, estimator xgb_limitdepth's best error=0.5468,   best estimator rf's best error=0.5
115
[flaml.automl.logger: 04-16 11:54:13] {2258} INFO - iteration 237, current learner rf
[flaml.automl.logger: 04-16 11:54:14] {2442} INFO -  at 256.2s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:14] {2258} INFO - iteration 238, current learner rf
[flaml.automl.logger: 04-16 11:54:18] {2442} INFO -  at 260.2s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:18] {2258} INFO - iteration 239, current learner xgboost
[flaml.automl.logger: 04-16 11:54:18] {2442} INFO -  at 260.5s, estimator xgboost's best error=0.5564,  best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:18] {2258} INFO - iteration 240, current learner rf
[flaml.automl.logger: 04-16 11:54:19] {2442} INFO -  at 261.9s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:19] {2258} INFO - iteration 241, current learner rf
[flaml.automl.logger: 04-16 11:54:22] {2442} INFO -  at 264.7s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:22] {2258} INFO - iteration 242, current learner rf
[flaml.automl.logger: 04-16 11:54:23] {2442} INFO -  at 265.7s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:23] {2258} INFO - iteration 243, current learner lgbm
[flaml.automl.logger: 04-16 11:54:26] {2442} INFO -  at 268.5s, estimator lgbm's best error=0.5258,     best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:26] {2258} INFO - iteration 244, current learner rf
[flaml.automl.logger: 04-16 11:54:30] {2442} INFO -  at 272.8s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:30] {2258} INFO - iteration 245, current learner rf
[flaml.automl.logger: 04-16 11:54:32] {2442} INFO -  at 274.2s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:32] {2258} INFO - iteration 246, current learner xgboost
[flaml.automl.logger: 04-16 11:54:37] {2442} INFO -  at 279.5s, estimator xgboost's best error=0.5564,  best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:37] {2258} INFO - iteration 247, current learner rf
[flaml.automl.logger: 04-16 11:54:40] {2442} INFO -  at 282.4s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:40] {2258} INFO - iteration 248, current learner xgb_limitdepth
[flaml.automl.logger: 04-16 11:54:42] {2442} INFO -  at 284.9s, estimator xgb_limitdepth's best error=0.5468,   best estimator rf's best error=0.5
115
[flaml.automl.logger: 04-16 11:54:42] {2258} INFO - iteration 249, current learner rf
[flaml.automl.logger: 04-16 11:54:43] {2442} INFO -  at 285.9s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:43] {2258} INFO - iteration 250, current learner rf
[flaml.automl.logger: 04-16 11:54:47] {2442} INFO -  at 289.9s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:47] {2258} INFO - iteration 251, current learner xgboost
[flaml.automl.logger: 04-16 11:54:48] {2442} INFO -  at 290.7s, estimator xgboost's best error=0.5564,  best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:48] {2258} INFO - iteration 252, current learner rf
[flaml.automl.logger: 04-16 11:54:51] {2442} INFO -  at 293.1s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:51] {2258} INFO - iteration 253, current learner lgbm
[flaml.automl.logger: 04-16 11:54:51] {2442} INFO -  at 293.3s, estimator lgbm's best error=0.5258,     best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:51] {2258} INFO - iteration 254, current learner rf
[flaml.automl.logger: 04-16 11:54:52] {2442} INFO -  at 294.6s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:52] {2258} INFO - iteration 255, current learner rf
[flaml.automl.logger: 04-16 11:54:56] {2442} INFO -  at 298.7s, estimator rf's best error=0.5115,       best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:56] {2258} INFO - iteration 256, current learner sgd
[flaml.automl.logger: 04-16 11:54:57] {2442} INFO -  at 299.2s, estimator sgd's best error=1.0486,      best estimator rf's best error=0.5115
[flaml.automl.logger: 04-16 11:54:57] {2258} INFO - iteration 257, current learner extra_tree
[flaml.automl.logger: 04-16 11:54:57] {2442} INFO -  at 299.5s, estimator extra_tree's best error=0.7579,       best estimator rf's best error=0.5
115
[flaml.automl.logger: 04-16 11:54:57] {2258} INFO - iteration 258, current learner extra_tree
[flaml.automl.logger: 04-16 11:54:57] {2442} INFO -  at 299.7s, estimator extra_tree's best error=0.7579,       best estimator rf's best error=0.5
115
[flaml.automl.logger: 04-16 11:54:57] {2258} INFO - iteration 259, current learner extra_tree
[flaml.automl.logger: 04-16 11:54:58] {2442} INFO -  at 300.0s, estimator extra_tree's best error=0.7128,       best estimator rf's best error=0.5
115
[flaml.automl.logger: 04-16 11:54:58] {2685} INFO - retrain rf for 0.3s
[flaml.automl.logger: 04-16 11:54:58] {2688} INFO - retrained model: RandomForestClassifier(criterion='entropy', max_features=0.7865259557990161,
                      max_leaf_nodes=183, n_estimators=112, n_jobs=-1,
                      random_state=12032022)
[flaml.automl.logger: 04-16 11:54:58] {1985} INFO - fit succeeded
[flaml.automl.logger: 04-16 11:54:58] {1986} INFO - Time taken to find the best model: 154.59597897529602
FLAML AutoML Accuracy: 0.7998741346758967
FLAML AutoML Classification Report:
              precision    recall  f1-score   support

           A       0.73      0.93      0.82       193
           B       0.86      0.91      0.88       721
           C       0.74      0.71      0.73       438
           D       0.77      0.53      0.63       237

    accuracy                           0.80      1589
   macro avg       0.78      0.77      0.76      1589
weighted avg       0.80      0.80      0.79      1589


FLAML AutoML Best Model: RandomForestClassifier(criterion='entropy', max_features=0.7865259557990161,
                      max_leaf_nodes=183, n_estimators=112, n_jobs=-1,
                      random_state=12032022)
```

Question: Did FLAML AutoML provide better performance compared to manually tuned models? Briefly explain.

Answer:

Given that we're looking to target class D customers, we'd want to know more about the relative costliness of making a type I vs type II error in assigning customers in D or out of D. Without taking AutoML into account, our best precision is .73 for class D and best recall is .54 with the random forest approach. AutoML improves upon the precision marginally to .77, while recall is nearly identical at .53, but still uses a RandomForest under the hood, but with slightly different parameters. The perforance is rougly in line with the manually tuned options we'd created.

Short-Answer Questions

- Question 1 : You are part of the marketing team at a subscription-based service (e.g., streaming platform). The goal is to predict whether a user will subscribe to a premium plan based on their usage patterns and demographics. You're using a machine learning model to predict subscription likelihood (Class 1: Subscribe, Class 0: No Subscribe).

How would you prioritize precision or recall for a subscription service when targeting users for a premium plan?

Again, prioritizing precision vs. recall depends on the specific cost trade-offs of the problem. Precision measures how many predicted positives are actually correct, while recall measures how many actual positives are correctly identified. To decide which to prioritize, we need to define a cost function that quantifies the impact of errors. In our subscription model classificatin problem, missing a potential subscriber (low recall) might mean losing revenue opportunities, while wrongly assuming someone will subscribe (low precision) could waste marketing resources. Understanding which error is more costly to the total value function helps determine whether to optimize for precision or recall (more specifically, how to optimize between the two).

- Question 2: You are working for an e-commerce company and have built a classification model to segment customers into high, medium, and low-value groups based on their likelihood of making future purchases (Class 1: High-Value, Class 2: Medium-Value, Class 3: Low-Value).

Is it likely that data imbalance issues will occur, and what precautions should be taken when interpreting the results?

When faced with imbalanced data, both precision and recall suffer. Precision can be boosted by predicting positives seldomly, over-indexing here might lead to a conservative model. The inverse, recall performs well when nearly all samples are predicted to be in-group. In the context of imbalanced data, we'd see high model accuracy when only predicting the majority class, a clear error.

- Question 3: Dynamic Pricing Model for Airline Industry You work at an airline, and the company uses a model to predict whether a customer will purchase a flight ticket at a given price (Class 1: Purchase, Class 0: No Purchase). The price changes dynamically based on factors like demand, booking time, and competitor pricing.

How would you adjust your model's decision threshold to optimize for maximum profit in this dynamic pricing scenario? What considerations should you take into account when setting this threshold?

We'd like to set our threshold such that we're maximizing expected model profit for a given factor set. Taking the total profit equation, (P-C)*Q(P), we care about finding the p/(p-c)=elasticity=dq*p / (dp*q) point for a given factor set. Given that the question is ambiguous about how a classification model is used to set prices, I'll assume it's used as a component which generated Q(P). We have q = (p-c)* dq/dp, we'd want a threshold such that, after accounting for classification errors, our expectation of purchases is exactly q. This would mean having some adjustment to our outputted prediction count based on the inverse of recall.

- Question 4: You work for an online payment company and are tasked with building a model to identify fraudulent transactions (Class 1: Fraud, Class 0: Legitimate). The dataset contains transaction amounts, user behavior, and historical fraud data.

How would you use model explainability tools like SHAP or partial dependence plots to communicate the rationale behind the model's decisions to stakeholders who may not be familiar with machine learning?

To communicate model decisions to non-technical stakeholders, I would use SHAP values to show how individual features (e.g., unusually high transaction amount or atypical login time) influenced the fraud prediction for a specific transaction, visualized as a bar or force plot. For broader patterns, I'd use partial dependence plots to illustrate how the predicted fraud probability changes as key variables (like transaction amount) vary, holding others constant. These tools help make the model's reasoning transparent by linking decisions to intuitive, real-world behaviors.

Question 5: Supervised learning is widely used in business applications to make data-driven decisions.

Task:

1. Identify a real-world business case where supervised learning can be applied. Define the outcome variable (target variable) for this problem.
2. Explain how you would evaluate the model's performance: Which evaluation metrics (e.g., Accuracy, Precision, Recall, F1-score, RMSE, AUC-ROC) would you choose?
3. Justify why those metrics are the most appropriate for your chosen business case.

💡 Hint: Consider cases like customer churn prediction, fraud detection, loan default prediction, or sales forecasting. Think about the cost of errors and which metric best captures the business objective.

A real-world business case for supervised learning is loan default prediction, where a bank aims to predict whether a loan applicant will default (target variable: 1 for default, 0 for non-default). To evaluate the model's performance, I would use precision, recall, F1-score, and AUC-ROC. Recall is important because failing to identify actual defaulters (false negatives) poses financial risk to the bank. Precision ensures that the model doesn't incorrectly label reliable borrowers as risky, which could lead to lost business. The F1-score provides a balanced view of precision and recall, especially valuable in imbalanced datasets. AUC-ROC helps evaluate overall

model performance across different thresholds. Together, these metrics align well with the bank's objective of minimizing credit risk while preserving customer relationships.