OLA Ensemble Learning

Link to google Colab: https://colab.research.google.com/drive/1r6sbp-ygG2-Y_NOg2ugZz8BHMtdACjZX?usp=sharing

Primary Objective:

The goal is to develop a predictive model that identifies drivers who are likely to leave Ola, allowing the company to take proactive retention measures.

Challenges & Additional Considerations:

1. High Churn Rate & Competition
    - Drivers can easily switch to competitors like Uber based on better rates and incentives.
    - Understanding key drivers of attrition can help in designing competitive retention strategies.
2. Cost of Attrition vs. Retention
    - Acquiring a new driver is more expensive than retaining an existing one.
    - Predicting and preventing attrition will reduce recruitment costs and improve overall operational efficiency.
3. Driver Segmentation for Better Retention Strategie
    - Drivers have different motivations: some work part-time, while others rely on it as a full-time job.
    - Predicting attrition patterns across different driver segments can lead to targeted interventions.
4. Time-Based Predictive Modeling
    - Attrition is a time-sensitive problem; we need to identify drivers at risk well before they leave.
    - A survival analysis or time-series-based approach may be necessary alongside classification models.
5. Intervention Planning & Business Impact
    - The model should not just predict attrition but also provide insights into why drivers leave.
    - Recommendations should be actionable (e.g., offering incentives, targeted engagement programs, or identifying key risk factors).

Outcome & Business Value:

- Reduce driver churn by identifying high-risk drivers early.
- Optimize operational costs by lowering recruitment expenses.
- Improve driver satisfaction through personalized retention strategies.
- Enhance revenue stability by maintaining a steady driver supply.

Dataset Overview:

Shape:

The dataset has 19,104 rows (records) and 14 columns (features).

- Each row represents an individual driver.
- The 14 columns likely contain demographic, tenure, and performance-related attributes.
- The dataset is moderately large, suitable for predictive modeling and exploratory data analysis (EDA).

```
# View the dataset Shape
print("Shape of dataset:", df.shape)
```

```
Shape of dataset: (19104, 14)
```

Datatype:

Numerical Features (int64/float64):

- Age, Gender, Education_Level, Income, Joining Designation, Grade, Total Business Value, Quarterly Rating
- Gender is stored as float64, which might indicate missing values or incorrect encoding.

Categorical Features (object):

- City
- Needs encoding for modeling.

Datetime Features (stored as object, needs conversion):

- MMM-YY, DateofJoining, LastWorkingDate
- These need to be converted to datetime64[ns] for proper analysis.
- Useful for calculating tenure and understanding attrition patterns over time.

Potential Issues:

- Unnamed: 0 appears to be an unnecessary index column.
- Gender stored as float64—needs checking for missing values or incorrect encoding.

- MMM-YY, DateofJoining, LastWorkingDate as object—should be converted to datetime format.

```
1 # Check the data types before converting
2 df.dtypes
```

	0
Unnamed: 0	int64
MMM-YY	object
Driver_ID	int64
Age	float64
Gender	float64
City	object
Education_Level	int64
Income	int64
Dateofjoining	object
LastWorkingDate	object
Joining Designation	int64
Grade	int64
Total Business Value	int64
Quarterly Rating	int64

Missing value:

The dataset contains missing values in the following columns:

1. Age → 61 missing values
2. Gender → 52 missing values
3. LastWorkingDate → 17,488 missing values

Analysis & Possible Handling Strategies

1. Age (61 missing values - relatively low impact)
   Possible Fix:
   - Fill missing values with the mean or median of the age column.
2. Gender (52 missing values - relatively low impact)
   Possible Fix:
   - Fill missing values based on probabilities (if gender distribution is imbalanced).
3. LastWorkingDate (17,488 missing values - major issue)
   This likely represents employees who are still working (i.e., they haven't left).
   Possible Fix:

- Fill missing values with a placeholder date (e.g., today's date) to indicate active employees.

```
# Get the missing values
df.isnull().sum()
```

	0
Unnamed: 0	0
MMM-YY	0
Driver_ID	0
Age	61
Gender	52
City	0
Education_Level	0
Income	0
Dateofjoining	0
LastWorkingDate	17488
Joining Designation	0
Grade	0
Total Business Value	0
Quarterly Rating	0

Statistical Summary:

- Age: Ranges from 21 to 58 years, with a median of 34 years.
- Gender: Binary (0 or 1), with a mean of 0.42, suggesting more males (if 1 represents male).
- Income: Highly varied, ranging from 1,0747 to 1,884,180.
- Education Level: Mostly between 0 and 2, likely categorical.
- Joining Designation & Grade: Spread across multiple levels (1 to 5).
- Total Business Value: Large variation, including negative values (potential data issue).
- Date Columns (MMM-YY, Date of Joining): Need conversion to datetime for proper analysis.

```python
1 # Get the statistical summary of the data
2 df.describe()
```

	Unnamed: 0	MMM-YY	Driver_ID	Age	Gender	Education_Level	Income	Dateofjoining	Joining Designation	Grade	Total Business Value	Quarterly Rating
count	19104.000000	19104	19104.000000	19043.000000	19052.000000	19104.000000	19104.000000	19104	19104.000000	19104.000000	1.910400e+04	19104.000000
mean	9551.500000	2019-12-11 02:09:29.849246464	1415.591133	34.668435	0.418749	1.021671	65652.025126	2018-04-28 20:52:54.874371840	1.690536	2.252670	5.716621e+05	2.008899
min	0.000000	2019-01-01 00:00:00	1.000000	21.000000	0.000000	0.000000	10747.000000	2013-04-01 00:00:00	1.000000	1.000000	-6.000000e+06	1.000000
25%	4775.750000	2019-06-01 00:00:00	710.000000	30.000000	0.000000	0.000000	42383.000000	2016-11-29 12:00:00	1.000000	1.000000	0.000000e+00	1.000000
50%	9551.500000	2019-12-01 00:00:00	1417.000000	34.000000	0.000000	1.000000	60087.000000	2018-09-12 00:00:00	1.000000	2.000000	2.500000e+05	2.000000
75%	14327.250000	2020-07-01 00:00:00	2137.000000	39.000000	1.000000	2.000000	83969.000000	2019-11-05 00:00:00	2.000000	3.000000	6.997000e+05	3.000000
max	19103.000000	2020-12-01 00:00:00	2788.000000	58.000000	1.000000	2.000000	188418.000000	2020-12-28 00:00:00	5.000000	5.000000	3.374772e+07	4.000000
std	5514.994107	NaN	810.705321	6.257912	0.493367	0.800167	30914.515344	NaN	0.836984	1.026512	1.128312e+06	1.009832

Final Takeaways from EDA

1. Income Distribution & Outliers

- The Income variable is right-skewed, with a long tail towards higher values.
- There are significant outliers in the upper range, indicating a small group of individuals with very high income.
- The median income is much lower than the maximum values, reinforcing the income disparity.

2. Relationship among classes Insights

- The correlation heatmap reveals weak relationships between variables.
- Income and Education Level have a slight positive correlation but are not strongly dependent.
- Age shows a weak correlation with Income, suggesting other factors might play a larger role in income determination.
- Quarterly Rating has a minor positive relation with Income, but not substantial enough to draw strong conclusions.

3. City-Wise Distribution

- The count plot for City shows an uneven distribution of data across different cities.
- Some cities have a significantly higher representation than others, indicating possible sampling biases or concentrated data collection.

4. Gender-Based Insights

- The correlation between Gender and Income is nearly zero, implying no significant gender-based income disparity in the dataset.
- However, additional analysis may be needed to confirm if other factors influence gender-based earnings.

5. Education Level vs Income

- Boxplot analysis shows that higher education levels do not drastically impact income in this dataset.
- The median income across education levels is quite similar, though higher education levels show a wider spread with more outliers.

6. Outliers & Skewness

- Several attributes have outliers, particularly in Income, requiring proper handling such as capping or transformation.
- The distribution of multiple variables is skewed, meaning transformations like log scaling may help normalize the data.

7. Overall Data Trends

- The dataset displays imbalances in categorical variables, such as City and Education Level distribution.
- No strong linear relationships exist between most numerical variables, meaning other complex interactions may be driving income and other attributes.

Data Preprocessing:

Standardization:

- Demonstrates standardization using StandardScaler(), which scales the features Age, Gender, and Income to have a mean of 0 and a standard deviation of 1. After transformation, the dataset is stored in df_scaled, where each column represents a standardized version of the original features, making them suitable for machine learning models by ensuring equal weightage.

```
# Scaling

scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df[['Age','Gender','Income']]))

```

```
# Data frame after scaling
df_scaled.head()
```

	0	1	2
0	-1.065629	-0.848779	-0.267358
1	-1.065629	-0.848779	-0.267358
2	-1.065629	-0.848779	-0.267358
3	-0.586223	-0.848779	0.044122
4	-0.586223	-0.848779	0.044122

KNN Imputation:

- The image demonstrates K-Nearest Neighbors (KNN) imputation, a technique to fill missing values using the average of the n_neighbors=3 closest data points. The imputed dataset, df_imputed, is checked for missing values, confirming that all have been successfully filled. This method ensures that imputed values maintain the local structure of the data.

```
# KNN imputation

impute = KNNImputer(n_neighbors=3)
df_imputed = pd.DataFrame(impute.fit_transform(df_scaled))
```

```
# Check if it has any missing values
df_imputed.isnull().sum()
```

	0
0	0
1	0
2	0

Feature Engineering:

The feature engineering performed here includes:

1. Age Grouping:
    - The Age variable is categorized into four distinct groups:
        - Young (0-20 years)
        - Teenage (20-30 years)
        - Middle_Age (30-50 years)
        - Senior (50-60 years)
    - This is achieved using the pd.cut() function, which bins the Age column into specified intervals and assigns corresponding labels.
2. Income Division:
    - The Income variable is categorized into three income groups:
        - Low (10,000 - 70,000)
        - Medium (70,000 - 150,000)
        - High (150,000 - 200,000)
    - This segmentation is done using pd.cut(), which helps in analyzing income-based patterns more effectively.

By performing these transformations, the dataset is enriched with categorical features (Age_Group and Income_Division), making it easier for models to understand patterns and derive insights.

Feature Engineering

```
# Feature Engineering Age
df['Age_Group'] = pd.cut(df['Age'],bins=[0,20,30,50,60],labels=['Young','Teenage','Middle_Age','Senior'])

# Feature engineering Income
df['Income_Division'] = pd.cut(df['Income'],bins=[10000,70000,150000,200000],labels=['Low','Medium','High'])

df.head()
```

	Unnamed: 0	MMM-YY	Driver_ID	Age	Gender	City	Education_Level	Income	Dateofjoining	Joining Designation	Grade	Total Business Value	Quarterly Rating	Age_Group	Income_Division
0	0	2019-01-01	1	28.0	0.0	C23	2	57387.0	2018-12-24	1	1	2381060	2	Teenage	Low
1	1	2019-02-01	1	28.0	0.0	C23	2	57387.0	2018-12-24	1	1	-665480	2	Teenage	Low
2	2	2019-03-01	1	28.0	0.0	C23	2	57387.0	2018-12-24	1	1	0	2	Teenage	Low
3	3	2020-11-01	2	31.0	0.0	C7	2	67016.0	2020-11-06	2	2	0	1	Middle_Age	Low
4	4	2020-12-01	2	31.0	0.0	C7	2	67016.0	2020-11-06	2	2	0	1	Middle_Age	Low

Encoding:

The feature engineering performed in the image includes:

1. Age Grouping:

   - The Age variable is categorized into four distinct groups:
     - Young (0-20 years)
     - Teenage (20-30 years)
     - Middle_Age (30-50 years)
     - Senior (50-60 years)
   - This is achieved using the pd.cut() function, which bins the Age column into specified intervals and assigns corresponding labels.
2. Income Division:

   - The Income variable is categorized into three income groups:
     - Low (10,000 - 70,000)
     - Medium (70,000 - 150,000)
     - High (150,000 - 200,000)
   - This segmentation is done using pd.cut(), which helps in analyzing income-based patterns more effectively.

By performing these transformations, the dataset is enriched with categorical features (Age_Group and Income_Division), making it easier for models to understand patterns and derive insights.

Encoding

+ Code + Text

```
# Encoding Age Groups

df = pd.get_dummies(df, columns=['Age_Group'],drop_first=True)
df[['Age_Group_Teenage','Age_Group_Middle_Age','Age_Group_Senior']] = df[['Age_Group_Teenage','Age_Group_Middle_Age','Age_Group_Senior']].astype(int)

# Encoding Income divisions
df = pd.get_dummies(df, columns=['Income_Division'])
df[['Income_Division_Low','Income_Division_Medium','Income_Division_High']] = df[['Income_Division_Low','Income_Division_Medium','Income_Division_High']].astype(int)

df.head()
```

	Unnamed: 0	MMM-YY	Driver_ID	Age	Gender	City	Education_Level	Income	Dateofjoining	Joining Designation	Grade	Total Business Value	Quarterly Rating	Age_Group_Teenage	Age_Group_Middle_Age	Age_Group_Senior	Income_Divisi
0	0	2019-01-01	1	28.0	0.0	C23	2	57387.0	2018-12-24	1	1	2381060	2	1	0	0	
1	1	2019-02-01	1	28.0	0.0	C23	2	57387.0	2018-12-24	1	1	-665480	2	1	0	0	
2	2	2019-03-01	1	28.0	0.0	C23	2	57387.0	2018-12-24	1	1	0	2	1	0	0	
3	3	2020-11-01	2	31.0	0.0	C7	2	67016.0	2020-11-06	2	2	0	1	0	1	0	
4	4	2020-12-01	2	31.0	0.0	C7	2	67016.0	2020-11-06	2	2	0	1	0	1	0	

Model Building:

Objective:

- Target Variable: Grade (represents the performance category of employees, drivers, or business associates).
- Predictor Variables: Includes Age, Gender, City, Education Level, Income, Total Business Value, Quarterly Rating, and the encoded categorical features (Age Group & Income Division).
- Objective: Predicting Grade effectively helps in business decision-making, such as:
    - Identifying high-performing individuals for promotions or incentives.
    - Detecting underperformers for training or intervention.
    - Allocating resources efficiently based on performance predictions.

1. Ensemble Bagging Algorithm

Model performance and Insights

- High Accuracy (~99.69%)
    - The model is highly accurate, meaning it can effectively classify employees into different performance grades.
    - Businesses can confidently automate employee assessment models using this approach.
- Potential Overfitting Concern
    - The very high accuracy suggests the model may be memorizing training data rather than generalizing well.

  - To ensure robustness, cross-validation and feature importance analysis should be performed.
- Feature Impact in Business Terms:
  - Income & Business Value: Likely strong indicators of performance (Grade).
  - Quarterly Rating: A crucial metric, possibly correlating directly with Grade.
  - Age & Experience: Can help assess how tenure impacts performance.

Recommendations

✔ Automated Performance Evaluation: The company can replace manual grading systems with this model to enhance efficiency.
✔ Employee Retention & Incentives: Identify top performers for promotions, salary hikes, and bonuses. Detect low-performing individuals for re-training or improvement programs.
✔ Resource Optimization: Allocate projects based on predicted employee/business performance.
✔ Risk Mitigation: Detect potential fraud or performance drops in business metrics early.

Ensemble Bagging Algorithm

```
# Train the model
rfclassifier = RandomForestClassifier(n_estimators=100, random_state=42)
rfclassifier.fit(X_train, y_train)

# Predict using the trained model
y_pred = rfclassifier.predict(X_test)

# Find accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy score from RFClassifier : ", accuracy)
```

```
Accuracy score from RFClassifier :  0.9968594608741167
```

2. Ensemble Boosting Algorithm:

Model performance and Insights:

- High Accuracy (~99.50%)
  - XGBoost delivers high accuracy, making it a powerful choice for predicting employee or business performance categories.
  - It is slightly lower than Random Forest (~99.69%), but boosting models generally perform better in complex, imbalanced scenarios.
- Why Boosting Works Better in Business Cases:
  - Gives importance to impactful factors: It will highlight features that significantly drive business/employee performance.
  - More robust than Random Forest for real-world scenarios: Especially if applied to HR analytics, performance reviews, and risk assessments.

Recommendations:

✔ Fairer Performance Evaluation & Employee Retention: The model helps identify under-recognized high performers who might have been overlooked in manual grading.
It prevents biases in employee promotions, salary hikes, and incentives, making the evaluation process more objective.
✔ Risk Management in Business Operations: XGBoost can be applied to detect employees at risk of leaving based on performance trends.
Business managers can intervene early with retention plans (salary hikes, role changes, training programs).
✔ Better Hiring & Talent Management: HR can use this model for new candidate assessments, ensuring only those who align with high performance trends are selected.
✔ Sales & Business Unit Performance Prediction: If applied beyond HR, XGBoost can predict high/low revenue-generating teams or projects based on past performance.

Actionable Insights and Recommendations:

Business Impact of Model Performance

A. Workforce Talent Management

- Insight: High accuracy suggests effective employee grading based on historical data.
- Recommendation:
  - Use the model's predictions to streamline promotions, salary increments, and retention plans.
  - Identify high performers early and implement a talent retention strategy.
  - For misclassified employees, analyze why predictions were incorrect (e.g., missing skills, underutilization).

B. Employee Attrition & Retention

- Insight: If the model predicts low-performing employees, the company can take preemptive action to upskill or replace them.
- Recommendation:
  - Provide personalized training programs for employees likely to be classified as lower grade.
  - Use AI-driven insights to redesign job roles, ensuring employees are in the right roles.

C. Hiring & Recruitment Optimization

- Insight: Predicting employee grades accurately allows HR teams to adjust recruitment strategies.
- Recommendation:
  - Filter candidates based on predicted performance scores and align hiring strategies accordingly.
  - Reduce time-to-hire by integrating this model into recruitment screening.
  - Target candidates who fit into higher-grade roles based on historical success.

Business Strategy Enhancements

A. Personalized Employee Growth Plans

- Insight: Employee grading predictions can help create personalized career pathways.
- Recommendation:
    - Offer targeted mentorship and training based on predicted performance trends.
    - Implement AI-driven career progression recommendations.

B. Performance-Based Compensation Models

- Insight: The model's insights allow for data-driven salary and bonus decisions.
- Recommendation:
    - Link model predictions to performance-based compensation systems.
    - Adjust salary structures dynamically to retain high performers.