AI Data Analysis Report

Report generated on 2025-04-18 20:48:47

Final Dataset Snapshot

Shape: (3000, 11)

Columns: id, name, age, city, score, salary, currency, account_balance, last_transaction, above_average_balance, last_transaction_year

Sample:

```
                                      id           name  age           city      score        salary currency  account_balance
last_transaction  above_average_balance  last_transaction_year
640a3066-483e-4cef-a00a-eacb70c2f0ed Alan Wilkinson   46 Lake Jamesland 338.262807  66511.460515
 GBP      4488.583235       2024-02-03                     0                  2024
5217454c-31b5-4a22-977f-7e9b62d073c5   Jamie Miller   30    Johnsonbury 491.617433  80313.122900
GBP     52175.220035       2024-10-17                     1                  2024
9d96c3da-c69c-4140-9486-95576b2110ed    Scott Haney   74     Torresbury 576.010300 112883.437864
GBP     43101.324112       2025-04-06                     0                  2025
e5e58691-7db3-4ac2-b7e9-77df8ca52eb1  Steven Lawson   64 West Brianfurt 607.868117  33158.711798
 USD     60923.464859       2023-08-17                     1                  2023
3b8c88a3-ea22-4949-a316-115f581e873d  Jeffrey Jones   45      Garymouth 378.799081  79358.097460
GBP     73552.418775       2025-01-01                     1                  2025
```

Data Distributions (Numeric Columns)

AI Data Analysis Report

Analyzer Output

technical_profile: {'shape': (3000, 9), 'columns': ['id', 'name', 'age', 'city', 'score', 'salary', 'currency', 'account_balance', 'last_transaction'], 'dtypes': {'id': 'object', 'name': 'object', 'age': 'int64', 'city': 'object', 'score': 'float64', 'salary': 'float64', 'currency': 'object', 'account_balance': 'float64', 'last_transaction': 'object'}, 'missing_values': {'id': 0, 'name': 0, 'age': 0, 'city': 0, 'score': 0, 'salary': 0, 'currency': 0, 'account_balance': 0, 'last_transaction': 0}, 'memory_usage': '1029.53 KB'}

ai_analysis: {'structure': 'Section not found', 'quality': 'Section not found', 'context': 'Section not found', 'recommendations': []}

preprocessing_ready: {'missing': [], 'outliers': [], 'transformations': []}

Operator Output

executed_operations: [{'operation': "df['last_transaction'] = pd.to_datetime(df['last_transaction'],

errors='coerce')", 'impact': "Executed: Convert 'last_transaction' to datetime objects and handle potential errors"}, {'operation': "df['above_average_balance'] = (df['account_balance'] > df['account_balance'].mean()).astype(int)", 'impact': 'Executed: Create a new feature indicating if the account balance is above average'}, {'operation': "df['last_transaction_year'] = df['last_transaction'].dt.year", 'impact': 'Executed: Extract the year of the last transaction for potential time-series analysis'}, {'operation': "# This example imputes with median, but you might choose a different strategy.\ndf['age'].fillna(df['age'].median(), inplace=True)", 'impact': 'Executed: Identify and handle missing values (impute age with the median, fill with 0, drop row)'}, {'operation': "df['score'] = df['score'].astype('float64')", 'impact': "Executed: Data Type conversions. Explicitly cast the 'score' to float64 to ensure consistency and prevent potential issues."}]

suggested_operations: [{'purpose': "Convert 'last_transaction' to datetime objects and handle potential errors", 'code': "df['last_transaction'] = pd.to_datetime(df['last_transaction'], errors='coerce')", 'safe_to_execute': True}, {'purpose': 'Create a new feature indicating if the account balance is above average', 'code': "df['above_average_balance'] = (df['account_balance'] > df['account_balance'].mean()).astype(int)", 'safe_to_execute': True}, {'purpose': 'Extract the year of the last transaction for potential time-series analysis', 'code': "df['last_transaction_year'] = df['last_transaction'].dt.year", 'safe_to_execute': True}, {'purpose': 'Identify and handle missing values (impute age with the median, fill with 0, drop row)', 'code': "# This example imputes with median, but you might choose a different strategy.\ndf['age'].fillna(df['age'].median(), inplace=True)", 'safe_to_execute': True}, {'purpose': "Data Type conversions. Explicitly cast the 'score' to float64 to ensure consistency and prevent potential issues.", 'code': "df['score'] = df['score'].astype('float64')", 'safe_to_execute': True}]

data_snapshot: {'shape': (3000, 11), 'missing_values': {'id': 0, 'name': 0, 'age': 0, 'city': 0, 'score': 0, 'salary': 0, 'currency': 0, 'account_balance': 0, 'last_transaction': 0, 'above_average_balance': 0, 'last_transaction_year': 0}, 'dtypes': {'id': 'object', 'name': 'object', 'age': 'int64', 'city': 'object', 'score': 'float64', 'salary': 'float64', 'currency': 'object', 'account_balance': 'float64', 'last_transaction': 'datetime64[ns]', 'above_average_balance': 'int64', 'last_transaction_year': 'int32'}}

Scientist Output

model_type: RandomForestClassifier

task: classification

target: last_transaction_year

features: ['age', 'city', 'score', 'salary', 'account_balance', 'above_average_balance']

AI Data Analysis Report

metrics: {'accuracy': 0.17166666666666666, 'classification_report': {'2020': {'precision': 0.1746031746031746, 'recall': 0.2018348623853211, 'f1-score': 0.18723404255319148, 'support': 109.0}, '2021': {'precision': 0.19672131147540983, 'recall': 0.21621621621621623, 'f1-score': 0.20600858369098712, 'support': 111.0}, '2022': {'precision': 0.1732283464566929, 'recall': 0.1981981981981982, 'f1-score': 0.18487394957983194, 'support': 111.0}, '2023': {'precision': 0.1326530612244898, 'recall': 0.10317460317460317, 'f1-score': 0.11607142857142858, 'support': 126.0}, '2024': {'precision': 0.176, 'recall': 0.19130434782608696, 'f1-score': 0.18333333333333332, 'support': 115.0}, '2025': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 28.0}, 'accuracy': 0.17166666666666666, 'macro avg': {'precision': 0.1422009822932945, 'recall': 0.1517880379667376, 'f1-score': 0.14625355628812875, 'support': 600.0}, 'weighted avg': {'precision': 0.16175073962749192, 'recall': 0.17166666666666666, 'f1-score': 0.16584134194115352, 'support': 600.0}}}
insights: ["Fallback classification model trained on target 'last_transaction_year'"]
training_code: None
warnings: ["Fallback used due to: 'parts'"]

Final AI Summary

? Data Quality & Structure

The dataset consists of 3000 rows and 9 columns initially. After feature engineering, it expanded to 11 columns. The data includes identifiers (id), personal information (name, age, city), financial details (score, salary, currency, account_balance), and transaction information (last_transaction).

- **Column Types:** Includes objects (strings), integers (age, above_average_balance, last_transaction_year), floats (score, salary, account_balance), and datetime (last_transaction).
- **Data Quality Issues:** No missing values were initially reported in the technical profile. The 'last_transaction' column was initially read as 'object' but correctly converted to datetime format.
- **Missing Values:** Handled by imputation (median) for age.
- **Outliers:** No explicit outlier detection or handling reported, which might be a concern for salary and account balance.

Business Context: The data appears to represent customer information, potentially for a financial institution. Understanding the relationships between customer demographics, financial status, and transaction history could be valuable for business decisions such as targeted marketing, risk assessment, or fraud detection.

? Preprocessing Actions

The following transformations were executed:

- **Datetime Conversion:** 'last_transaction' column converted to datetime objects using `pd.to_datetime()`, handling potential parsing errors.
- **Feature Engineering:**
    - 'above_average_balance': Created a binary feature indicating whether an account balance is above the average account balance.
    - 'last_transaction_year': Extracted the year from the 'last_transaction' column.
- **Missing Value Handling:** The 'age' column's missing values were imputed with the median.
- **Data Type Conversion:** The 'score' column was explicitly cast to 'float64'.

No skipped operations are reported.

? Modeling & Results

- **ML Task:** Classification
- **Target:** `last_transaction_year` (predicting the year of the last transaction).
- **Model Type:** RandomForestClassifier (fallback model).
- **Features:** `['age', 'city', 'score', 'salary', 'account_balance', 'above_average_balance']`
- **Performance Metrics:**
    - Accuracy: 0.172
    - The classification report shows low precision, recall, and F1-score for each year. Precision ranges from 0 to 0.196 and recall from 0 to 0.216, demonstrating a poor performance for a classification model.

AI Data Analysis Report

Insights: The model's accuracy is very low, indicating that it is not effectively predicting the `last_transaction_year`. The model used was a "fallback" model, indicating an issue with the original model selection or training process, specifically citing "parts". The features used do not seem to be strong predictors of transaction year.

? Key Recommendations

1. **Improve Model Performance:**
    - **Investigate "parts" error:** Find out why the fallback model was used by inspecting the logs. Debug and resolve the underlying issue preventing the intended model from training properly.
    - **Feature Engineering:** Explore creating more relevant features that might correlate with the transaction year. Consider interaction terms or aggregated features.
    - **Model Selection:** Experiment with other classification algorithms.
    - **Hyperparameter Tuning:** Tune the hyperparameters of the chosen model using techniques like cross-validation and grid search.
2. **Data Quality:**
    - **Outlier Analysis:** Perform outlier analysis on numerical columns like `salary` and `account_balance` to determine if outliers are affecting model performance.
    - **Currency Conversion**: Standardize currency to a single currency for accurate analysis of financial features.
3. **Further Exploration:**
    - **Target Distribution:** Analyze the distribution of `last_transaction_year`. If the distribution is imbalanced, consider using techniques like oversampling or undersampling to address the imbalance.
    - **Feature Importance:** Analyze feature importance from a properly trained model to understand which features contribute most to the prediction and guide feature engineering efforts.
4. **Pipeline Enhancement:**
    - **Automated Feature Selection:** Implement automated feature selection techniques to identify the most relevant features for the model.
    - **Error Handling:** Improve error handling in the pipeline to gracefully handle unexpected data issues

and provide informative error messages.

5. **Refine Business Understanding:**

- Collaborate with domain experts to identify potential features or data sources that could improve model accuracy.

- Define clear business objectives for the model and ensure that the model's performance is aligned with those objectives.