Affinity Predictor

Data Mining and Machine Learning project

Luca Arduini

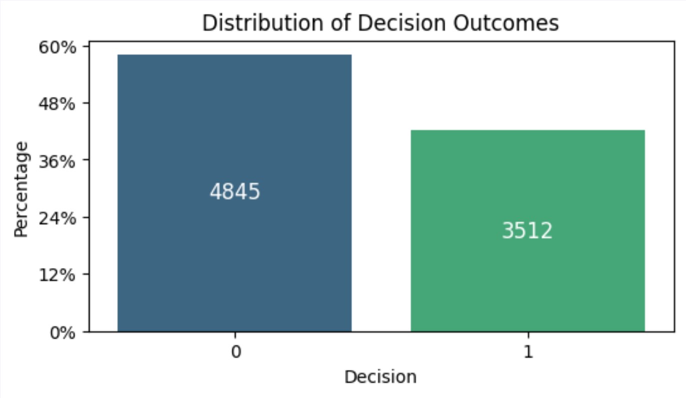
Introduction

- In the era of online dating, this project aims to improve how people find compatible matches.
- Using real data from speed dating experiments, we develop a **Machine Learning model to classify** potential partners for each user.
- The goal is to show users only the profiles they are likely to be interested in, offering smarter and more personalized recommendations.

Dataset

- Detailed dataset from a speed dating study, containing information gathered both before and after participants had 4-minute dates.
- To simulate a real-world dating app, we methodically removed all features that would be unknown before a meeting, such as partner ratings.
- The resulting dataset consists of 8378 interactions, now perfectly tailored to train a model that predicts a user's decision based solely on their sign-up profile.

number of instances	8378
number of features	49 (45 numerical, 4 categorical)
number of classes	2
number of missing values	9995 (2.44%)
number of instances with at least one missing value	6637 (79.42%)



Data Preprocessing: Key Steps

Exploratory Data Analysis (EDA)

Analyzed data structure and confirmed a balanced target variable. Visualizations revealed key behavioral patterns.

Outlier Management

Identified outliers but retained them as they represent valid user data, not errors.

Feature Engineering

Simplified the field attribute by grouping 219 unique values into 9 broader categories.

Data Correction & Cleaning

Fixed data entry errors, resolved inconsistencies, and removed two non-essential columns.

Handling Missing Values

Removed rows with excessive NaNs and used the Miss Forest algorithm to impute the rest.

Correlation Analysis

Confirmed no high correlation between features, ensuring all attributes contribute unique information.

Automated Preprocessing Pipeline

To ensure robust and leak-proof processing within our cross-validation, we built a comprehensive **pipeline** using scikit-learn. The core is a **ColumnTransformer** that applies specific transformations to each feature type before feeding the data to the model.

- **For Numeric Features:**
    - IterativeImputer: Intelligently fills missing values.
    - RobustScaler: Scales data while being insensitive to outliers.
- **For Categorical Features:**
    - SimpleImputer: Fills missing values with the most frequent category.
    - OneHotEncoder: Converts text labels into a numerical format.
- **For Binary Features:**
    - A custom transformer maps text (female/male) to integers (0/1).
    - SimpleImputer: Fills any remaining missing values.

This entire preprocessor was then chained with each model, creating a final, unified Pipeline for GridSearchCV. This guarantees that data is processed correctly and consistently in every step of the evaluation.

Model Selection & Hyperparameter Tuning

We systematically evaluated four powerful ensemble models to find the best predictor for a user's decision.

Methodology

We used **GridSearchCV** with 5-fold stratified cross-validation to perform an exhaustive search for the optimal hyperparameters for each model.

Evaluation Criteria

Performance was measured using Accuracy, F1-Score, and ROC-AUC.
Accuracy was chosen as the primary metric (refit='accuracy') to determine the single best model.

Candidate Models

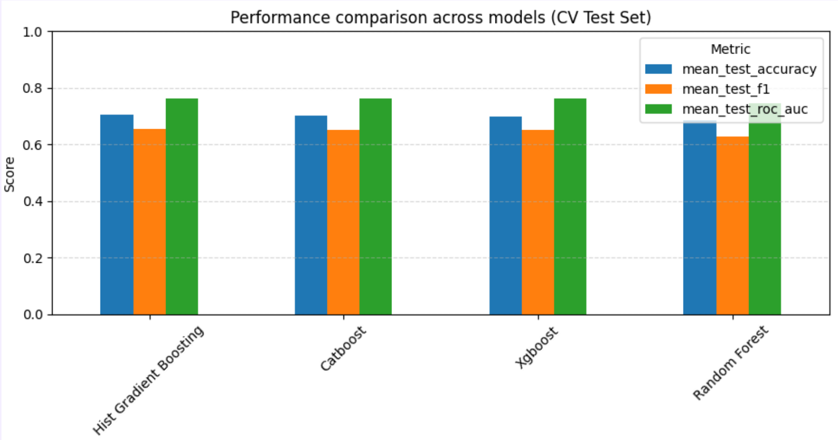
- HistGradientBoostingClassifier
- XGBClassifier
- CatBoostClassifier
- RandomForestClassifier

Model Comparison

After running the grid search, we compared the cross-validation performance to select our final model.

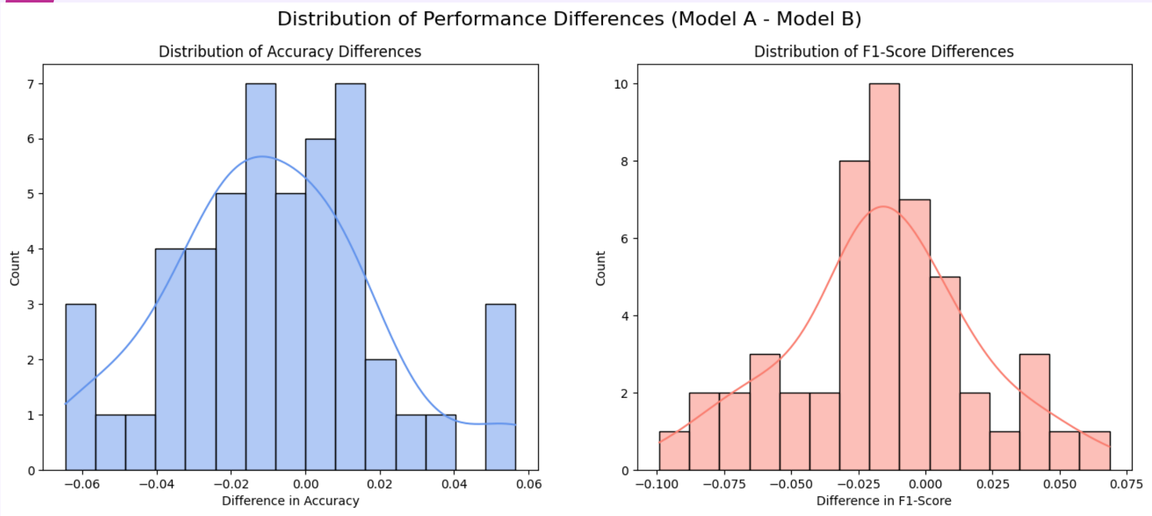
The **HistGradientBoostingClassifier** emerged as the top performer across all key metrics on the validation sets.

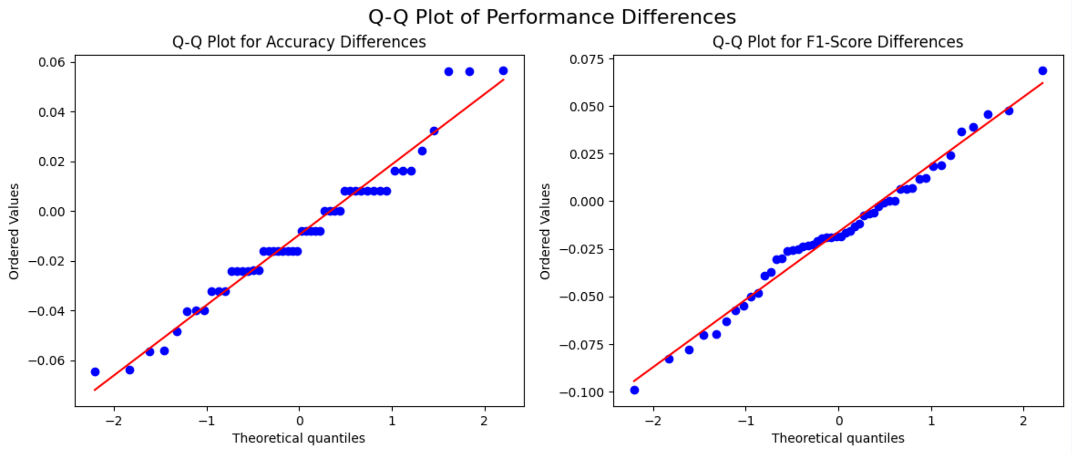
Model	Mean test accuracy	Mean test F1-score	Mean test roc-auc
Hist Gradient Boosting	0.7048	0.6542	0.7640
Catboost	0.7014	0.6524	0.7630
Xgboost	0.6994	0.6516	0.7634
Random Forest	0.6866	0.6287	0.7443



Statistical Validation

- **The Question:** Our initial cross-validation showed HistGradientBoosting as the winner, but CatBoost was extremely close. Was this difference real or just due to chance?
- **Our approach:** We performed a rigorous, head-to-head statistical comparison on the unseen test set to find the definitive best model.
- **The methodology (Paired Statistical Testing):**
  1. **Resampling the Test Set:**
     - We used **RepeatedStratifiedKFold** (50 repeats, 10 folds) to generate 500 stable evaluation subsets from our test data.
  2. **Paired Evaluation (No Re-training):**
     - On each of the 500 folds, we evaluated our **pre-trained** models and calculated the difference in their performance scores.
  3. **Hypothesis Testing:**
     - A **paired t-test** was used on these 500 differences to determine if the average performance gap between the models was statistically significant.


Distribution of Performance Differences (Model A - Model B)
Distribution of Accuracy Differences
Count
Difference in Accuracy
Distribution of F1-Score Differences
Count
Difference in F1-Score


Q-Q Plot of Performance Differences
Q-Q Plot for Accuracy Differences
Ordered Values
Theoretical quantiles
Q-Q Plot for F1-Score Differences
Ordered Values
Theoretical quantiles

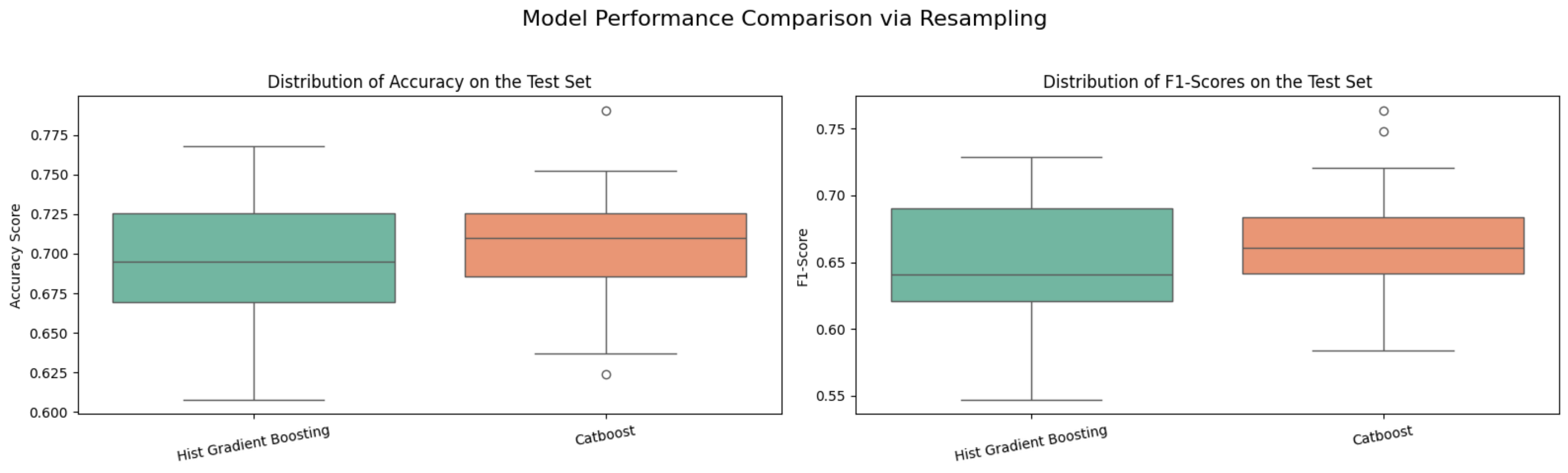
Results & Decision

Before running the t-test, we confirmed our data met the required normality assumption using the Shapiro-Wilk test (p-values > 0.05). This validated our choice of statistical test.

Paired T-Test Results:

Metric	p-value	Statistical Conclusion
Accuracy	0.0185	**Statistically Significant** ($p < 0.05$)
F1-Score	0.0019	**Statistically Significant** ($p < 0.05$)

- The p-values for both metrics are well below 0.05. This allows us to **reject the null hypothesis** that the models perform equally.
- The observed performance advantage of CatBoost on the test set is not a random fluke, but a **statistically significant finding**.
- **Conclusion**: Despite the initial cross-validation results, this more robust analysis proves **CatBoost is the superior model**. It is therefore selected as our final, champion model.



Model Performance Comparison via Resampling
Distribution of Accuracy on the Test Set
Accuracy Score
0.775
0.750
0.725
0.700
0.675
0.650
0.625
0.600
Hist Gradient Boosting
Catboost
Distribution of F1-Scores on the Test Set
F1-Score
0.75
0.70
0.65
0.60
0.55
Hist Gradient Boosting
Catboost

Final Evaluation:
CatBoost Performance on Unseen Data

Our champion model, CatBoost, was evaluated on the hold-out test set to measure its true generalization performance. The results confirm its effectiveness as a robust classifier for our dating app.

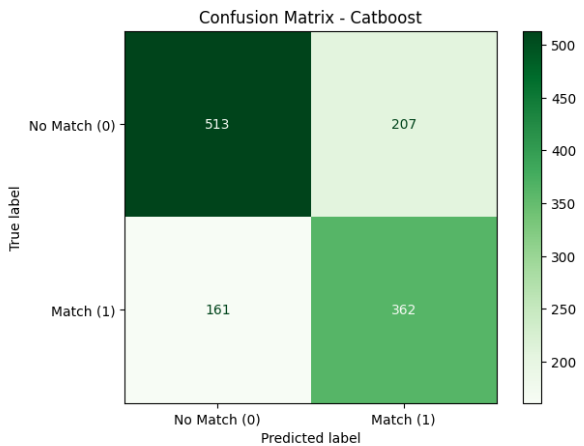
Metric	Score
Accuracy	70.4%
F1-Score	66,3%
ROC-AUC Score	78,6%

```
Classification Report on Test Set:
              precision    recall  f1-score   support

No Match (0)       0.76      0.71      0.74       720
   Match (1)       0.64      0.69      0.66       523

    accuracy                           0.70      1243
   macro avg       0.70      0.70      0.70      1243
weighted avg       0.71      0.70      0.71      1243

--- Final Metrics on Test Set ---
Accuracy: 0.7039
F1-Score: 0.6630
ROC-AUC Score: 0.7859
-----------------------------------
```



- **Strong Overall Performance**: The model correctly predicts a user's decision 7 out of 10 times.
- **Effective Filtering**: It is highly reliable when filtering out incompatible profiles (76% precision for "No Match").

Further Experiments to Enhance Performance

To push our model's performance further, we tested three additional techniques. The goal was to improve accuracy and reduce the observed overfitting.

- **1. PCA (Principal Component Analysis)**
  - **Goal:** Reduce dimensionality and complexity.
  - **Result: Performance significantly degraded.** Accuracy dropped, and overfitting actually *increased* as the transformed features confused our tree-based models.
  - **Decision: REJECTED**
- **2. Feature Selection (SelectKBest)**
  - **Goal:** Improve performance by using only the most impactful original features.
  - **Result: No significant improvement.** The models consistently performed best when using the full feature set. The marginal gains were negligible.
  - **Decision: REJECTED**
- **3. Feature Discretization**
  - **Goal:** Reduce overfitting by binning the top 3 most important continuous features.
  - **Result: Performance slightly decreased.** This targeted approach also failed to provide any benefit.
  - **Decision: REJECTED**

Thanks for your attention!

Luca Arduini