Dating Application (LOVOO) ML Project

My project is based on the work of Jeffrey Mvutu Mabilama, who decided to gather data in dating app called *Lovoo* to try to answer questions in relation to social metrics. What makes a successful profile? Who will have more matches? What are attractive qualities of a profile?

Using Machine Learning models we can learn what constitutes a popular profile and try to predict how many views a profile will receive based on the available data of a user. The views is represented by column ‘profileVisits’.

Disclaimers about the data:

Mabilama collected 2015 data, while the *Lovoo* was expanding in European countries. The metrics of the *Lovoo* include social media like features such as ‘count kisses’, ‘count fans’, that mirror likes and followers. Also, the ‘profileVisits’ column is usually difficult to obtain.

The data gathered by browsing through profiles and searches **recommended** by *Lovoo*'s algorithm for 2 profiles created for this purpose a male, open to friends & chats & dates. That is why there are only female users in the dataset. Also, the recommendation algorithm’s output produced the same set of user profiles which could mean that it was relying on location (to recommend more people nearby than people in different places or countries). This lowered the number of different user profiles that would be presented and included in the dataset.

Data Preparation:

There were two csv files: users\_api\_results and user instances. Upon exploring, there are a number of columns that are duplicates and have different names, mostly boolean values:

* location= locationCity
* isFlirtstar=flirtstar
* isHighlighted = highlighted
* isMobile= mobile
* isOnline= online
* birthd= hasBirthday
* LastOnlineDate= LastOnline
* LastOnlineTime = lastOnlineTs
* isNew = Freshman
* isVip=isVIP
* IsVerified = verified

There were other mutual columns ('gender','age','name','counts\_details','counts\_profileVisits', 'distance', 'flirtInterests\_chat', 'flirtInterests\_date', 'flirtInterests\_friends', 'isSystemProfile', 'lang\_count', 'lang\_de', 'lang\_en', 'lang\_es', 'lang\_fr', 'lang\_it', 'lang\_pt','pictureId', 'whazzup', 'locationCity'', 'crypt', 'counts\_kisses', 'counts\_pictures', 'city') and also some unique columns for each csv.

The Following actions were taken:

1. For the most optimal merge of the two csvs all the columns were examined and manipulated:
   1. fillna(0)/ fillna(‘NA’).
   2. Replacing in all columns were there were: True and False to values 0 and 1.
   3. Uniforming the date
   4. Rounding the float columns
   5. Removal of punctuation
2. Merging of the two csvs into one dataset – outer join
3. Dropping duplicates by userId. userId will be the index
4. Converting objects to string
5. Deleting irrelevant columns

EDA- Explanatory Data Analysis:

* Print types of values, shape and info
* Using describe() for statistics
* Creating EDA report
* Exploring all column values and types
* Checking histograms to find whether any of the columns are normalized. Non were.

Cleansing – Managing outliers and nulls

* Scatter plots and boxplots to identify outliers
* Identifying columns that have outliers that change the distribution and don’t change the correlation with the target value ‘counts\_profileVisits’
* Two columns met the criteria for dropping: ‘counts\_fans’ and ‘distance’. ‘counts\_fans’ was not manipulated, because there could be a person who can have substantially more fans than others. Outliers of ‘distance’ had to be dropped.
* KNNImputer method was used for imputation of the missing ‘distance’ values.
* Missing values: deleting columns that have more than 60% missing. The column ‘locked’
* Cleaning columns with adding none or 0 to complete all columns for the next step of model selection.
* The results: 4008 records

Feature Engineering and Feature Selection:

* ‘userId’ is set as the index
* Splitting the date to into month and day, since the year is the same in all value
* Creating word cloud of the column ‘whazzup’
* Label Encoding
* Feature selection with Lasso and Random Forest

**he updated performance metrics for your models show the following results:**

**1. Decision Tree**

* **MSE: 8.65×10−38.65 \times 10^{-3}8.65×10−3**
* **RMSE: 0.093**
* **MAE: 0.0028**
* **RMSLE: 0.000113**

**Interpretation: The Decision Tree model shows extremely low errors, which might indicate overfitting to the training data.**

**2. XGB (XGBoost)**

* **MSE: 1.67×1021.67 \times 10^21.67×102**
* **RMSE: 12.91**
* **MAE: 4.58**
* **RMSLE: 0.1603**

**Interpretation: XGBoost performs very well, with low error metrics across the board. It is one of the better-performing models, especially considering the relatively low RMSLE.**

**3. Random Forest**

* **MSE: 1.46×1061.46 \times 10^61.46×106**
* **RMSE: 1209.39**
* **MAE: 450.50**
* **RMSLE: 0.4599**

**Interpretation: Random Forest shows a solid performance with reasonable errors, but it is less accurate than XGBoost.**

**4. ADABoost**

* **MSE: 1.46×1061.46 \times 10^61.46×106**
* **RMSE: 1209.39**
* **MAE: 450.50**
* **RMSLE: 0.4599**

**Interpretation: ADABoost has identical performance to Random Forest, indicating a similar level of accuracy.**

**5. GBM (Gradient Boosting Machine)**

* **MSE: 3.34×1063.34 \times 10^63.34×106**
* **RMSE: 1828.48**
* **MAE: 975.61**
* **RMSLE: 1.1013**

**Interpretation: GBM performs worse than XGBoost and the ensemble methods (Random Forest and ADABoost), showing higher error values.**

**6. Linear Regression**

* **MSE: 8.68×1068.68 \times 10^68.68×106**
* **RMSE: 2946.12**
* **MAE: 1481.89**
* **RMSLE: 1.7649**

**Interpretation: Linear Regression shows the highest errors among all models except SVM, suggesting it’s not well-suited for this particular dataset.**

**7. SVM (Support Vector Machine)**

* **MSE: 5.25×1075.25 \times 10^75.25×107**
* **RMSE: 7242.84**
* **MAE: 3229.52**
* **RMSLE: 1.8950**

**Interpretation: SVM continues to be the poorest performer, with the highest error metrics.**

**Conclusion:**

* **Best Performers: Decision Tree and XGBoost (XGB) stand out as the top models. While the Decision Tree may be overfitting, XGBoost provides a strong balance of accuracy and generalization.**
* **Moderate Performers: Random Forest and ADABoost are reliable options but not as strong as XGBoost.**
* **Poor Performers: GBM, Linear Regression, and SVM show higher errors and are less effective in this context.**

**If generalization is important, XGBoost appears to be the best overall choice.**

**4o**

**You said:**

**Best Parameters: {'max\_depth': None, 'max\_features': 'auto', 'min\_samples\_leaf': 4, 'min\_samples\_split': 5, 'n\_estimators': 10} Best Score: -0.15420739988044505**

**ChatGPT said:**

**ChatGPT**

**The results you've provided indicate the best hyperparameters found during the tuning process for a model, likely using a method like grid search or random search. Here’s a breakdown of what these parameters mean and how they relate to the model:**

**Best Parameters:**

* **max\_depth: None: This means that the decision trees in your model are allowed to grow as deep as necessary, potentially until all leaves are pure or contain fewer than min\_samples\_split samples. This setting could lead to overfitting if not controlled with other parameters like min\_samples\_leaf or n\_estimators.**
* **max\_features: auto: This setting means the model will consider all features when looking for the best split. This is typical for Random Forest models, where auto usually implies using the square root of the number of features in the case of classification or all features for regression.**
* **min\_samples\_leaf: 4: This parameter controls the minimum number of samples that a leaf node must have after the split. Setting this to 4 helps in reducing overfitting by ensuring that the trees don’t grow too complex with very small leaf nodes.**
* **min\_samples\_split: 5: This parameter specifies the minimum number of samples required to split an internal node. A value of 5 helps in preventing splits that create very small child nodes, which could help in generalization.**
* **n\_estimators: 10: This represents the number of trees in the ensemble. A smaller number like 10 indicates a lightweight model, though this might lead to underfitting. However, this can be balanced by the depth and structure of the trees.**

**Best Score:**

* **Best Score: -0.15420739988044505: This score is likely a negative value of some error metric (such as the negative mean squared error) obtained during cross-validation. A score closer to 0 is better, meaning the model has lower errors. However, since it's negative, the magnitude gives an indication of the average error the model is making. The closer this score is to 0, the better the model performed during cross-validation.**

**Interpretation:**

* **The model with these parameters has been optimized to balance complexity and generalization. However, the n\_estimators being quite low at 10 could suggest that the model might benefit from more trees, depending on computational constraints.**
* **The negative score suggests there’s room for improvement, potentially by fine-tuning the parameters further or by exploring other algorithms or preprocessing techniques.**

**Next Steps:**

* **Increase n\_estimators: You might want to experiment with increasing the number of estimators, as more trees generally improve the model's performance up to a certain point.**
* **Cross-validation: Re-run cross-validation with these parameters to confirm that they generalize well to unseen data.**
* **Feature Engineering: Consider additional feature engineering or trying different algorithms if the score remains suboptimal.**