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

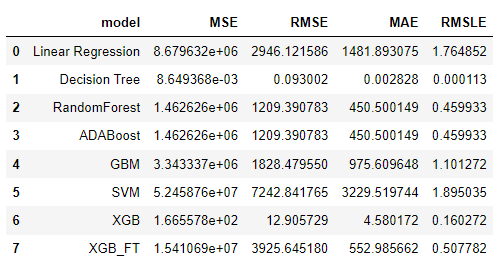
* Scatter plots and boxplots to identify outliers
* Identifying columns that have outliers that change the distribution and don’t change the correlation with the target value ‘counts\_profileVisits’
* Two columns met the criteria for dropping: ‘counts\_fans’ and ‘distance’. ‘counts\_fans’ was not manipulated, 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’
* Feature selection with Lasso, SVM, GradientBoost, Random Forest
* Label Encoding

**Model Selection and Fine-Tuning**

Regression models were examined by the regression metrics:



* The Decision Tree model shows extremely low errors, which might indicate overfitting to the training data.
* XGBoost performs with low error metrics. It is one of the better-performing models, especially considering the relatively low RMSLE.
* Random Forest and ADABoost shows a solid performance with reasonable errors, but it is much less accurate than XGBoost.
* GBM (Gradient Boosting Machine) shows higher error values.
* Linear Regression shows the highest errors among all models except SVM, suggesting it’s not well-suited for this particular dataset.
* SVM (Support Vector Machine)is the poorest performer, with the highest error metrics.
* Best Performers: Decision Tree and XGBoost (XGB) stand out as the most effective models. While the Decision Tree may be overfitting, XGBoost provides a strong balance of accuracy and generalization.

**The Model XGBoost was chosen:**

* Splitting the data for train and test
* GridSearch for hyperparameter fine-tuning

**Best Parameters: enable\_categorical=True, max\_depth=20, colsample\_bytree=1.0, min\_child\_weight=4, gamma=0, n\_estimators=5**

* Model Performance
* Mean Absolute Error: 1416.7915

**Conclusions:**