Current approach:

Single xgboost model + feature engineering + bagging + threshold2class optimization

Steps, which I used to achieve current results:

See function name in {} from utils_tanz.py module.

1. Some values correction
2. Feature selection { greedy_selection}
3. Text processing – replace by value-counts { cats2valueCounts}
4. *Tf-idf on letters (cleared from special symbols) followed by TruncatedSVD to reduce dimensions (from ~10000 to 100) followed by k-means / ICA (Independent component analysis) { addTfIdfFeats , TfIdfFeats }
5. HyperOpt xgboost parameter optimization
6. Usage lag feature date_recorded_lag as difference in days between date_recorded and max(date_recorded)
7. Bagging (up to 30 times) { run_mc_xgb, module level1run.py}
8. Usage regression on the 2level + threshold2class optimization (instead of multiclass classification) {lin2classes_accuracy_threshold, module level2blend.py}

* The idea of 3/4-letter-ngrams was to find some connections between high features with a lot of unique values (like subvillages, wpt_name) with specific Tanzanian names like sangamwampuya. I create feature like [sang, anga, ngam, …] and put it to tf-idf.

For validation, I used 5-fold Cross Validation on level 1 and 2, which corresponds changing quality on LB quite well:

CV level1 Multiclass	CV level2 threshold2class optimization	LB
0.81449	0.81595	0.8211
0.81779	0.81737	0.8251
0.81791	0.81835	0.8259
0.81829	0.81846	0.827

Things didn't help:

Unfortunately, many classical technics didn't help in this competition:

1. Nan / incorrect values replacing:
    a. Correct values for amount_tsh, gps_height, construction_year, population by common sense (mean\median by group instead of zero/nan) { makeDataset }
    b. Usage different levels to correct values{FillGeoByStagesLatLong, FillGeoByStagesRegions, FillGeoByStages}

    c. Combine different replacing techniques to create several datasets for future processing on the 2^{nd} level { module run4datasets.py }
2. Text processing
    a. Leave only common categories in train\test {SetUncommonCatsToNan} - common practice in DS competitions
    b. Get statistic (Letter/Words count/ratios) by text fields { processStringField}
    c. Different types of category encodings:
        i. SVD cat2cat encoding {code_factor},
        ii. one_hot encoding { cats2oneHot},
        iii. cats2valueCounts + noise { cats2valueCountsNoise}
    d. Target encoding - replacing category by Bayesian representation of target variable:
        i. Inside of CV-loop {add_mean_target}
        ii. In-line replacing + noise {cats2targetMeanStd}
    e. Tf-idf on words followed by TruncatedSVD to reduce dimensions (from ~10000 to 100) followed by k-means / ICA
3. Feature interactions (got by XGBoost Feature Interactions Reshaper) to create new features { make_conj} with removing rare cats { delSmallCats}
4. Ensembles (got overfitting on level2 even with simple methods)
5. Other methods (RF, KNN, SVM, NN, LightGBM)
6. Run linear instead of classification models on level1 { run_linear }
7. Other xgboost objective function (Rank, Poisson, reg:linear)
8. Catboost – Yandex Boosting Algorithm for efficient category processing

<u>Future steps, which can help</u>

a. TSNE after tf-idf on text features (too slow, takes half a day to check each text feature)
b. Word-embedding for text representation
c. Deep learning techniques to create meta-features for xgboost
d. Another techniques on level 2