SPAM CLASSIFICATION STRATEGY

Overview Of The Complete Process Of Creating A Spam Classifier

WHAT WAS DONE

I. INITIALIZATION
  a. Load the necessary packages
  b. Load the data set
  c. Label the columns
  d. Examine the type of each column in the data set
  e. Transform each column into proper type (factor, date, numerical)

II. DATA EXPLORATION
  a. Dimension of the data
  b. Structure of the data
  c. Summary of the data
  d. Check if data is balance or not. That is, if our target variable has an equal distribution of labels (50% spam and 50% ham).
  e. Investigate each column. Check consistency of values/levels of each categorical variable.
  f. Are there any columns that can be deleted?
    - is_privacy_policy_accepted - it only have one level which is "false"
    - is_listed_in_directory - it only have one level which is "false"
    - site_currency - there is no information available, 100% NA.
    - google_analytics_activated_at – data has only 5% completeness.
  g. Tabulate target variable (spam/ham) vs each categorical variable to have initial idea of spam occurrence on each level of predictor. Visualize if necessary
  h. Inspect presence of NA in the data set.
    - How many missing values are there on each variable?
    - Can the rows be deleted?
    - Which columns have NAs that can be dealt with?

III. DATA CLEANING AND PREPARATION FOR MODELLING
  a. Remove columns that can be deleted.
    - Columns that are 100% NA
    - Columns which only have 5% completeness.
  b. Remove rows that has NA which cannot be imputed with a new category. “last_login_at” has very minimal number of rows which can be deleted.
  c. For columns that has a lot of NA that we cannot afford to delete, we can create a new category to represent the NA’s. NA’s in the country and user_city column can be labelled as “unknown”

d. For columns that have significantly high number of levels (like country, email_provider), create a new category that will group the low frequency levels into one bucket say "others". Retain the original column for reference.

e. Feature engineering. Create new features from existing features.
   - Split created_at into
     - created_at_date
     - created_at_day
     - created_at_hour
   - Split last_login_at into
     - last_login _at_date
     - last_login _at_day
     - last_login _at_hour

f. Re-evaluate the cleaned data set and see if there are still any anomalies present.

g. If none. Extract the final columns (exclude the original columns of which has multiple levels)

h. Split the data set into train and test set. At this point, 2 use cases were tested. From the data exploratory stage, it was observed that data exhibit imbalance in the target variable (91.3% ham, 8.7% spam).
   - Use Case 1: Unbalanced train set.
     Train set should exhibit unequal distribution between spam and ham as the original data set (91.3% ham, 8.7% spam).
   - Use Case 2: Balanced train set.
     Train set should exhibit equal distribution between spam and ham (50% ham, 50% spam).

IV. MODELLING

For each use case, two models were fitted: random forest and adaboost.

a. Random Forest
   - Scenario 1: Assuming no regrouping is done on the predictors with significantly high number of levels. Random forest cannot deal with those kind of predictors so in this scenario, completely ignore those predictors and build the model as is from those remaining predictors.
   - Scenario 2: Conduct variable importance and validate the result by doing a forward selection of predictors starting from the most to least important variable to fine tune the optimal set of predictors based on AUC computed on the validation set. Then train the final model using the final set of predictors.

     NOTE: Majority of the most important predictors are either re-coded or generated feature out of the existing feature.

b. Adaboost
   - Scenario 1: Train the model using all predictors.
   - Scenario 2: Train the model using only the final set of predictors (same final set of predictors with random forest final model.

V. PERFORMANCE ASSESSMENT

After running different use case and different scenario, model fitted from scenario 2 were treated as the final model for each use case. For fair comparison, the final Random forest and final Adaboost model had the same set of predictors. Below are the summary of metrics achieved.

Use Case	Model	Accuracy	Precision	Recall	AUC
Balanced	Random Forest	0.8507	0.3558	0.8836	0.9369
	Adaboost	0.8479	0.3442	0.8266	0.7391
Unbalanced	Random Forest	0.9492	0.8225	0.5321	0.9080
	Adaboost	0.9420	0.7257	0.5369	0.5037

VI. FINAL MODEL

With the above result, I treat the unbalanced random forest model as my final spam classifier. It has high AUC and accuracy. In terms of precision, it achieved the highest among the 4 models either though it performs poorly in terms of Recall.

LIMITATIONS/ISSUES ENCOUNTERED

- Limited domain knowledge. I might have cleaned, manipulated, or even transformed the data better if I have domain knowledge.
- Number of trees grown was limited to 500 specially on the unbalanced case due to capacity issues.
  - `Error: cannot allocate vector of size 2.3 Gb`

MODELS TO TRY NEXT

- Generalized Additive Models (GAM) – especially if there will be enough numerical data
- Logistic Regression – but also doubtful if it can handle the number of predictors as well as the type of predictors which are mostly categorical.
- Support Vector Machine (SVM) – without machine/memory limitations, this might have been a good classifier because SVM has a great reputation specially on binary classification.
- Neural Network – same comment as with SVM