Mobilewalla - Data Science Intern Assessment

Shriya Mandarapu, Dec 24 2021

Objective: Objective of this exercise is to accurately infer the gender of given set of devices using a machine learning approach. We need to predict the unknown gender for devices based on their device Information, app usage, app description and other features.

Data:

- Test & Train
  Train Dataset and Test Dataset containing deviceId – "ifa" and other aggregated information along with target variable "gender"
  Train Dataset - Shape: (230263, 43)
  Test Dataset - Shape: (49737, 43)
  Train data - Target Variable - Gender - Imbalanced - Female: 13%, Male: 87%
- App MetaData
  App Metadata dataset containing app, app description and other information regarding apps. There could be multiple keys here like appId, id, bundleId, etc.. Choose the id the would help utilize the dataset most in conjunction with other datasets.
- App Propensity Scores
  User App propensity dataset containing deviceId - "ifa" and propensity measure against apps ("bundle") for all deviceIds (both train and test dataset).

Methodology & Steps -

- Data Preprocessing, EDA, Data Visualization
- Data Cleaning
  - Null Values - Imputation
  - Handling Outliers
- Scaling & One hot Encoding
- Feature Extraction, Dimensionality Reduction(PCA) & Scree Plot
- Class Imbalance Techniques
  Random Under Sampling, Random Over Sampling, SMOTE, Tomek Links, Near Miss SMOTEENN, SMOTETomek
- Classification models - Base models
  Logistic Regression, LDA, KNN Classifier, AdaBoost, XGBoost, Gradient Boosting Classifier, Random Forest Classifier, Extra Trees Classifier, Light GBM, CatBoost
- Voting Classifier (Soft & Hard)
- GridSearchCV - HyperParameter Tuning
- Promising - LightGBM
- Keras - Tensorflow

Feature Engineering - App Metadata - App Propensity Score

From App Metadata & Propensity score data - I've created a table with ifa's mapped with corresponding propensity scores for each app category.Ifa is the index to map onto from train or test datasets.
There were several other metrics but I only picked category since it felt most relevant for the use case.

Null Values - Imputation

Possible choices -Mean, Median, Mode, MICE, KNN Imputation
Numerical Columns - Individual columns are visualized to observe their distributions.Almost all the

	ifa	SOCIAL	TOOLS	PERSONALIZATION	PHOTOGRAPHY	PRODUCTIVITY	COMMUNICATION	ART_AND_DESIGN
0	00002c2abb627089988287fa9cb1569b43076bdfdad67d...	0.00708	0.17354	0.0000	0.00000	0.00000	0.00176	0.0
1	00009f893e849715bcc809dd0891e78bc4ef9dd3061eac...	0.00646	0.02586	0.7635	0.03187	0.00000	0.00619	0.0
2	0000ac97665d160ff6c012f0af44a13a83d0065cdd8046...	0.00000	0.51015	0.0000	0.39687	0.03031	0.00000	0.0
3	0000bef11e506141ce0f4159e0412038c892477ff7752b...	0.00000	0.00000	0.0000	0.01475	0.00000	0.00735	0.0
4	000123a2d98e624b7f11b4fc81f9879bc8b7aab9c7dc75...	0.00000	0.25724	0.0000	0.37289	0.27507	0.00000	0.0

Figure 1: App Metadata & Propensity Score Mapped onto ifa

numerical columns are not symmetrical in their distributions. Thereby mean will be a poor choice. Median/mode is the most ideal way to go about it. Although, I did calculate MICE imputed values, it seemed more of an overkill. So in this case, I've chosen to go with **Median** as my imputation choice.
Categorical Columns - Categorical columns such as platform and device_category with one class being the majority are imputed with the **most_frequent** class.

Outliers

Possible choices -IQR, Z score, Quantiles, Other statistical tests such as grubb's test, etc.
Since we have rather unsymmetrical distributions, I've only taken a particular set of columns who show decent spread to eliminate only the most severe outliers.
I've filtered values **"> 75%Quantile + 3*IQR" and "< 25%Quantile - 3*IQR"**.

PCA

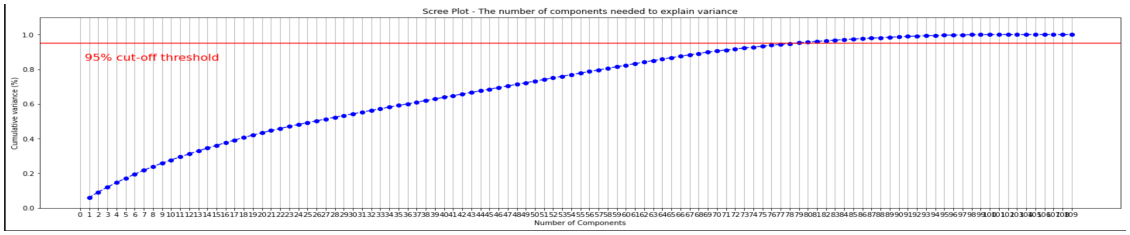

Figure 2: Scree Plot - PCA(n=78) captures 95% Variance

Class Imbalance
When we have clear class imbalance, resampling techniques are used to make it even ground.

- **Over Sampling** : Random Over Sampling, SMOTE
- **Under Sampling** : Random Under Sampling, TomekLinks, NearMiss
- **Combined (Over + Under)** : SMOTETomek, SMOTEENN

Basic Models - Performance Summary
Below the model performances with varying input data conditions. All the models run are based on base(or)default conditions.Tables below give precision, recall and F1_score values for both class zero(Female) and class one(Male) predictions. It can be observed that since Female are the minority class, the values are much lower for Females compared to males.
Also, it can be noted that although models do show high accuracy, there is very poor predictive power for the minority class when sampling methods are not used. Under/Over sampling methods improved the predictions of the minority class, ofcourse at the expense of overall accuracy.
It is desirable to have a model of high "auc_roc_score" than one with high "accuracy".

Voting Classifiers - Soft & Hard
I have used these models as my voting models -

	Accuracy	Precision_One	Precision_Zero	Recall_One	Recall_Zero	ROC_AUC_Score	F1_Score_One	F1_Score_Zero
DummyClassifier	0.8768	0.8768	0.1232	1.0000	0.0000	0.5000	0.9344	0.0000
LogisticRegression	0.8803	0.8974	0.1065	0.9742	0.2117	0.5930	0.9345	0.3035
LDA	0.8751	0.9114	0.0998	0.9475	0.3601	0.6538	0.9301	0.4154
KNeighborsClassifier	0.8697	0.9016	0.1042	0.9542	0.2683	0.6112	0.9277	0.3365
ADABoostClassifier	0.8808	0.9153	0.0985	0.9495	0.3913	0.6704	0.9332	0.4471
GradientBoostClassifier	0.8853	0.9064	0.1017	0.9679	0.2971	0.6325	0.9367	0.3894
RandomForestClassifier	0.8820	0.9035	0.1031	0.9677	0.2721	0.6199	0.9350	0.3623
ExtraTreeClassifier	0.8826	0.9002	0.1049	0.9733	0.2372	0.6052	0.9357	0.3324
XGBoostClassifier	0.8859	0.9115	0.0996	0.9617	0.3464	0.6540	0.9366	0.4278
LightGBMClassifier	0.8864	0.9071	0.1013	0.9684	0.3025	0.6355	0.9373	0.3962
CatBoostClassifier	0.8863	0.9097	0.1003	0.9648	0.3280	0.6464	0.9371	0.4155

Figure 3: Original Scaled Data - 70-30 Split

	Accuracy	Precision_One	Precision_Zero	Recall_One	Recall_Zero	ROC_AUC_Score	F1_Score_One	F1_Score_Zero
DummyClassifier	0.8768	0.8768	0.1232	1.0000	0.0000	0.5000	0.9344	0.0000
LogisticRegression	0.8789	0.8952	0.1079	0.9755	0.1913	0.5834	0.9339	0.2801
LDA	0.8752	0.9107	0.1001	0.9486	0.3524	0.6505	0.9302	0.4102
KNeighborsClassifier	0.8685	0.9010	0.1045	0.9534	0.2637	0.6086	0.9271	0.3306
ADABoostClassifier	0.8768	0.8972	0.1067	0.9698	0.2145	0.5922	0.9324	0.3002
GradientBoostClassifier	0.8795	0.8974	0.1065	0.9730	0.2133	0.5932	0.9340	0.3035
RandomForestClassifier	0.8790	0.8940	0.1087	0.9774	0.1785	0.5779	0.9340	0.2664
ExtraTreeClassifier	0.8788	0.8928	0.1095	0.9789	0.1666	0.5727	0.9341	0.2530
XGBoostClassifier	0.8789	0.9011	0.1044	0.9671	0.2513	0.6092	0.9334	0.3383
LightGBMClassifier	0.8819	0.8980	0.1061	0.9754	0.2160	0.5957	0.9354	0.3105
CatBoostClassifier	0.8806	0.9003	0.1048	0.9704	0.2411	0.6057	0.9344	0.3322

Figure 4: PCA Data - 70-30 Split

	Accuracy	Precision_One	Precision_Zero	Recall_One	Recall_Zero	ROC_AUC_Score	F1_Score_One	F1_Score_Zero
DummyClassifier	0.8768	0.8768	0.1232	1.0000	0.0000	0.5000	0.9344	0.0000
LogisticRegression	0.7838	0.9422	0.1038	0.7862	0.7666	0.7764	0.8645	0.4663
LDA	0.7876	0.9420	0.1034	0.7915	0.7599	0.7757	0.8673	0.4685
KNeighborsClassifier	0.7681	0.9244	0.1005	0.7883	0.6241	0.7062	0.8563	0.3986
ADABoostClassifier	0.7717	0.9398	0.1037	0.7732	0.7605	0.7669	0.8559	0.4507
GradientBoostClassifier	0.8019	0.9426	0.1027	0.8096	0.7474	0.7785	0.8776	0.4817
RandomForestClassifier	0.8734	0.9060	0.1020	0.9528	0.3082	0.6305	0.9296	0.3749
ExtraTreeClassifier	0.8778	0.8904	0.1113	0.9812	0.1423	0.5617	0.9337	0.2229
XGBoostClassifier	0.8197	0.9397	0.1006	0.8366	0.6996	0.7681	0.8906	0.4888
LightGBMClassifier	0.8091	0.9442	0.1028	0.8170	0.7534	0.7852	0.8824	0.4930
CatBoostClassifier	0.8252	0.9420	0.1010	0.8408	0.7139	0.7774	0.8940	0.5015

Figure 5: RandomOverSampling-PCAData-70-30Split

	Accuracy	Precision_One	Precision_Zero	Recall_One	Recall_Zero	ROC_AUC_Score	F1_Score_One	F1_Score_Zero
DummyClassifier	0.8768	0.8768	0.1232	1.0000	0.0000	0.5000	0.9344	0.0000
LogisticRegression	0.7814	0.9420	0.1039	0.7833	0.7682	0.7758	0.8627	0.4640
LDA	0.7861	0.9425	0.1037	0.7889	0.7662	0.7775	0.8661	0.4687
KNeighborsClassifier	0.7400	0.9345	0.1038	0.7384	0.7518	0.7451	0.8328	0.4160
ADABoostClassifier	0.7681	0.9390	0.1036	0.7696	0.7578	0.7637	0.8534	0.4460
GradientBoostClassifier	0.7990	0.9435	0.1032	0.8045	0.7595	0.7820	0.8753	0.4820
RandomForestClassifier	0.7874	0.9415	0.1032	0.7919	0.7555	0.7737	0.8672	0.4668
ExtraTreeClassifier	0.7828	0.9405	0.1032	0.7871	0.7524	0.7698	0.8641	0.4605
XGBoostClassifier	0.7716	0.9412	0.1042	0.7715	0.7726	0.7721	0.8556	0.4546
LightGBMClassifier	0.7919	0.9438	0.1039	0.7947	0.7713	0.7830	0.8701	0.4772
CatBoostClassifier	0.7907	0.9446	0.1043	0.7923	0.7790	0.7857	0.8691	0.4783

Figure 6: RandomUnderSampling-PCAData-70-30Split

	Accuracy	Precision_One	Precision_Zero	Recall_One	Recall_Zero	ROC_AUC_Score	F1_Score_One	F1_Score_Zero
DummyClassifier	0.8768	0.8768	0.1232	1.0000	0.0000	0.5000	0.9344	0.0000
LogisticRegression	0.7899	0.9421	0.1033	0.7944	0.7582	0.7763	0.8690	0.4706
LDA	0.7941	0.9418	0.1029	0.8002	0.7505	0.7754	0.8721	0.4731
KNeighborsClassifier	0.7361	0.9270	0.1021	0.7428	0.6882	0.7155	0.8315	0.3911
ADABoostClassifier	0.7781	0.9382	0.1026	0.7837	0.7378	0.7608	0.8610	0.4503
GradientBoostClassifier	0.8088	0.9418	0.1020	0.8195	0.7326	0.7760	0.8826	0.4855
RandomForestClassifier	0.8536	0.9267	0.0974	0.8983	0.5356	0.7169	0.9150	0.4741
ExtraTreeClassifier	0.8581	0.9230	0.0975	0.9090	0.4954	0.7022	0.9183	0.4623
XGBoostClassifier	0.8134	0.9390	0.1008	0.8292	0.7015	0.7653	0.8863	0.4809
LightGBMClassifier	0.8120	0.9425	0.1020	0.8229	0.7344	0.7787	0.8848	0.4905
CatBoostClassifier	0.8192	0.9407	0.1009	0.8347	0.7091	0.7719	0.8901	0.4914

Figure 7: SMOTE-PCAData-70-30Split

	Accuracy	Precision_One	Precision_Zero	Recall_One	Recall_Zero	ROC_AUC_Score	F1_Score_One	F1_Score_Zero
DummyClassifier	0.8768	0.8768	0.1232	1.0000	0.0000	0.5000	0.9344	0.0000
LogisticRegression	0.7176	0.9412	0.1084	0.7005	0.8392	0.7698	0.8131	0.4226
LDA	0.6986	0.9385	0.1085	0.6788	0.8389	0.7589	0.7980	0.4067
KNeighborsClassifier	0.6713	0.9329	0.1079	0.6501	0.8225	0.7363	0.7762	0.3814
ADABoostClassifier	0.6881	0.9370	0.1086	0.6670	0.8379	0.7525	0.7895	0.3982
GradientBoostClassifier	0.7213	0.9411	0.1080	0.7055	0.8336	0.7696	0.8162	0.4242
RandomForestClassifier	0.8004	0.9413	0.1023	0.8092	0.7378	0.7735	0.8767	0.4766
ExtraTreeClassifier	0.8088	0.9400	0.1013	0.8219	0.7156	0.7687	0.8829	0.4797
XGBoostClassifier	0.7553	0.9426	0.1061	0.7483	0.8053	0.7768	0.8428	0.4477
LightGBMClassifier	0.7397	0.9435	0.1078	0.7268	0.8321	0.7794	0.8304	0.4406
CatBoostClassifier	0.7597	0.9440	0.1064	0.7523	0.8123	0.7823	0.8459	0.4543

Figure 8: SMOTEENN-PCAData-70-30Split

'LogisticRegression', 'LDA', 'AdaBoostClassifier', 'GradientBoostingClassifier', 'RandomForestClassifier', 'ExtraTreeClassifier', 'XGBoostClassifier', 'LightGBMClassifier', 'CatBoostClassifier'

	Accuracy	Precision_One	Precision_Zero	Recall_One	Recall_Zero	ROC_AUC_Score	F1_Score_One	F1_Score_Zero
SoftVoting-PCA	0.8799	0.8998	0.1051	0.9702	0.2371	0.6036	0.9340	0.3271
SoftVoting-PCA-RandomUnderSampled	0.7994	0.9447	0.1037	0.8036	0.7695	0.7865	0.8754	0.4858
HardVoting-PCA-RandomUnderSampled	0.7978	0.9444	0.1037	0.8019	0.7686	0.7853	0.8743	0.4836
SoftVoting-PCA-SMOTEENN	0.7651	0.9451	0.1065	0.7582	0.8148	0.7865	0.8499	0.4608
SoftVoting-OriginalScaled	0.8832	0.9055	0.1021	0.9665	0.2901	0.6283	0.9355	0.3795
SoftVoting-OriginalScaled-RandomUnderSampled	0.8210	0.9576	0.1089	0.8161	0.8559	0.8360	0.8889	0.5409

Figure 9: Voting Classifier - With different Input Data

Promising Light GBM

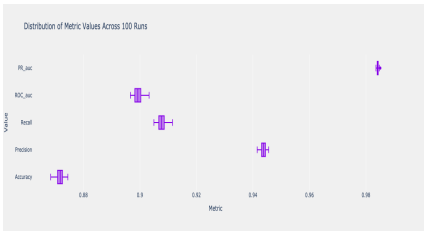

Figure 10: LightGBM - Optimizer1

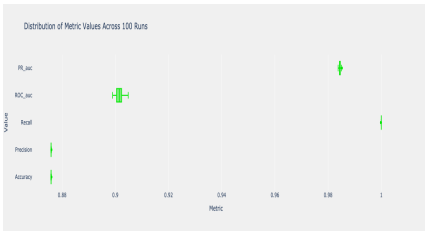

Figure 11: LightGBM - Optimizer2

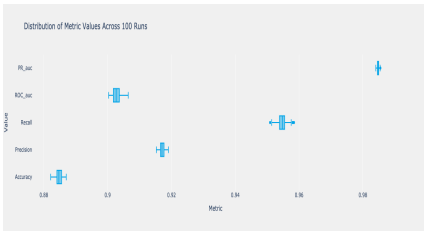

Figure 12: LightGBM - Optimizer3

Keras - Tensorflow

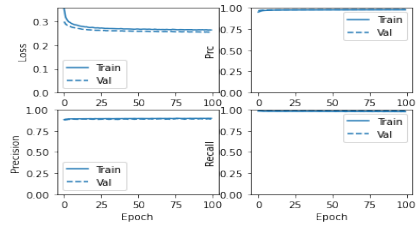

Figure 13: Keras - Baseline Model

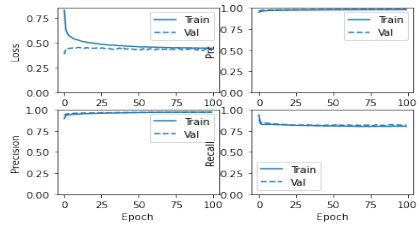

Figure 14: Keras - Weighted Class

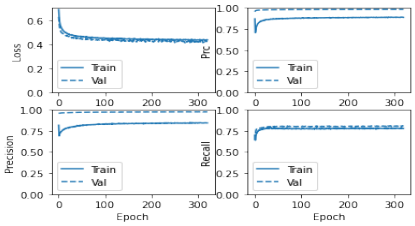

Figure 15: Keras - Over Sampled

Submission:
Although I had a couple of models with similar ROC-AUC scores, I've chosen Light GBM for my output prediction. Light GBM outputs probablities.

Best Model Metrics -

- **Accuracy** : 0.8868033851880465
- **Precision**: 0.9099151743638078
- **Recall** : 0.9665992510863253
- **ROC-AUC** : 0.9042868864519347
- **PR-AUC** : 0.9850967709983607

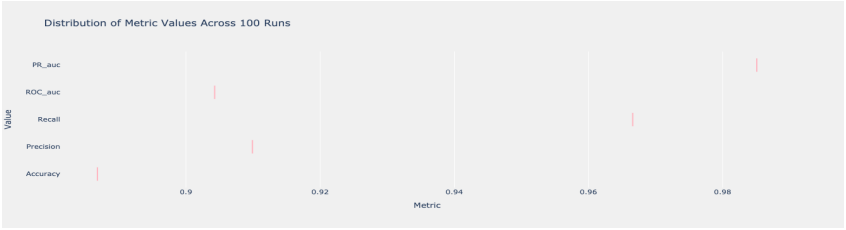

Figure 16: Best Light GBM - Used for Output Submission

Output File:
File: lightgbm_model_predicted_probs.csv

	ifa	Predicted_Probs
0	8e932096849088aa76a953ce2f69728db6b6cdc056cd91...	0.946926
1	9ba004d2e45aa4ae05935634f6198bf81e8766cb76a2bc...	0.774377
2	17a74969d7681ff8e206aeb7ba0e83300c8d4ed1f6db80...	0.995480
3	c8f3ddc5e2b0b3deacdb49181bc170363e5d01c992a660...	0.996855
4	9c0c1acf953e443c577a30350a723ddf0a818e0763dce3...	0.998650

Figure 17: Best Light GBM - Used for Output Submission

Code for output evaluation

```
accuracy = accuracy_score(y_actual, predictions[:] >= 0.5)
roc_auc = roc_auc_score(y_actual, predictions[:])
precision = precision_score(y_actual, predictions[:] >= 0.5)
recall = recall_score(y_actual, predictions[:] >= 0.5)
pr_auc = average_precision_score(y_actual, predictions[:])
```

P.S: Please refer to the notebooks attached along with this report for entire data preprocessing, EDA, model steps and additional information.