Amex Project Final Report Group 20

(Team name in Amex portal: IDA_Group_20)

Aim:

To develop a credit risk model that predicts the likelihood of a customer to default payment. The data present for a customer is at any given point in time. We need to develop a model to predict the likelihood of the customer defaulting after 12 months. In ML terms, it is basically a classification problem.

Procedure:

- **Importing the datasets(train and test datasets) and some python packages**
    - Train dataset
    - Test dataset
    - Python packages for data preprocessing, classification, model selection and hyperparameter tuning

- **Data cleaning and preprocessing (performed on both train and test datasets)**
    - **Checking for missing values and converting all missing values to NaN** - We noticed that all missing values are not given the value 'NaN'. Some of them are named with different placeholders like 'missing' and 'na'. We found out all such

    placeholders and replaced them with NaN (for both train and test datasets)
    - **Label encoding** - By inspection, we found that all columns except mvar47 are numerical and does not need encoding.
    - **Datatype correction** - Now that all the columns are numerical, we converted all of them to float datatype for uniformity. (Except application key which is just an id and default_index which is our classification target)
    - **Checking for class imbalance in the training data** - We noticed that there is some level of class imbalance in the training data. The level of imbalance was saved in a variable so that it could be used as a hyperparameter for some classification algorithms like XGBoost and LightGBM
    - **Imputing the missing values** - We imputed the missing values with the mean of the column it belongs to. We tried other methods like median, dropping rows and columns, etc but mean imputation produced better results.

- **Model selection**
    - We used a min-max scaler
    - We used stratified cross-validation (on the scaled dataset) to find the best model
    - We used F1 score on the test data as the metric.
    - Default hyperparameters were used (except the class imbalance level parameter)
    - Individual models we tried:
        - Logistic Regression
        - KNN
        - Decision Tree
        - Random Forest
        - Bagging
        - XGBoost
        - AdaBoost
        - Light Gradient Boost (LightGBM)

- XGBoost and LightGBM produced the best results, hence we tried using a Voting of these two (both soft and hard voting was tried).
- Finally, the following models gave the best F1 scores:
    1. LightGBM
    2. XGBoost
    3. Soft Voting of LightGBM and XGBoost
    4. Hard Voting of LightGBM and XGBoost

- **Hyperparameter tuning**
    - We performed tuning only for hyperparameters in LightGBM, (Since cross-validation for XGBoost is time consuming).
    - Therefore, tuning for (1) is done completely, (3) and (4) is done partially

- **Reporting the final train and test scores for all 4 models(with tuned hyperparameters)**

  (Included in 'Results')
  All four models, with tuned hyperparameters, gave almost similar F1 scores

- **Predictions on the test dataset and final csv submission**

  We trained on the entire training dataset. We tried all four models (given above, with tuned hyperparameters) for the leaderboard submission. The one that gave our best score (60.6%) for the leaderboard is LightGBM with tuned hyperparameters

MODEL SELECTED FOR FINAL SUBMISSION: LIGHTGBM (WITH TUNED HYPERPARAMETERS)

Selected classification model: LightGBM (Light Gradient Boosting Machine)

LightGBM is a type of gradient boosting model that uses tree-based learning algorithms. It has faster training speed, higher accuracy and lower memory usage.

Hyperparameters that were tuned and used:

- n_estimators: no of boosted trees that need to be fitted
- max_depth: max tree depth for each of the learners
- learning_rate: learning rate used in boosting
- subsample: ratio of subsampling of the training instance
- colsample_bytree: ratio of subsampling of columns when constructing each tree
- scale_pos_weight: weight of labels with positive class (imbalance correction)

Tuned hyperparameter values:
'n_estimators': 760
'max_depth': 85
'learning_rate': 0.02935086260767057
'subsample': 0.3792225838646886
'colsample_bytree': 0.21433910546560286

Untuned hyperparameter values:
‘scale_pos_weight’: value was taken from the training dataset
(default values for other hyperparameters)

Results: (on the tuned LightGBM model)

Train Confusion Matrix

```
[[37022  4621]
 [10294 14463]]
```

Test Confusion Matrix

```
[[8832 1453]
 [2997 3318]]
```

Train Accuracy: 0.7753765060240964
Test Accuracy: 0.7319277108433735

Train F1 score: 0.6597933441299241
Test F1 score: 0.5985928197726863

Insights and takeaways from the project

- The project helped us to understand a typical classification problem(more specifically, an imbalanced classification problem) and by solving it in a step-by-step fashion, we were able to understand the importance of each step and how it contributes to the overall performance of the model.
- We were able to try out different models taught in class and hence gain some practical knowledge. At the same time, we learnt some extra concepts too.
- Even though we were not given the actual names of each variable, working on a practical dataset helped us to understand what level of accuracy is required for companies, etc
- We learnt about advanced models like LGBM and XGBoost

Group members who contributed:

- **Nohan Joemon**
- **Mohsin Sackeer**
- **Sauban Asharaf**
- **Bharatwaajan B**