2022AI511ML Project Report
Credit Card Fraud Detection Kaggle (It’s a Fraud)

Team: AshTea

Members -

Ashish Gatreddi (IMT2020073)

Chaitanya Manas Cheedella (IMT2020053)

PROBLEM : Given details about online transactions, detect whether they’re a fraud or not.

DATASET

- TransactionDT : timedelta from a given reference datetime (not an actual timestamp)
  TransactionDT : corresponds to the number of seconds in a day.
- TransactionAMT : transaction payment amount in USD
- ProductCD : product code, the product for each transaction
- card1 - card6: payment card information, such as card type, card category, issue bank, country, etc.
- addr : address
  “both addresses are for purchaser
  addr1 as billing region
  addr2 as billing country”
- dist : distances between (not limited) billing address, mailing address, zip code, IP address, phone area, etc.”
- P_ and (R__) emaildomain : purchaser and recipient email domain
  “ certain transactions don't need a recipient, so R_emaildomain is null.”
- C1-C14 : counting, such as how many addresses are found to be associated with the payment card, etc.
- D1-D15 : timedelta, such as days between previous transactions, etc.
- M1-M9 : match, such as names on card and address, etc.
- Vxxx : Vesta engineered rich features, including ranking, counting, and other entity relations.
- “id01 to id11 are numerical features for identity.
  Other columns such as network connection information (IP, ISP, Proxy, etc),digital signature (UA/browser/os/version, etc) associated with transactions are also present.

Datasets link -
https://drive.google.com/drive/folders/1r3frBYSr0wEjdGW4zdhbx5U9IkTgdyih?usp=share_link

PREPROCESSING

The major steps in preprocessing are :

- Dealing with null/missing values (Dropping / Imputation)
- Datapoints grouping (Similar values in categorical data)
- Dealing with categorical and non-numeric data (One shot / Label encoding)
- Feature Selection (Correlation)
- Normalisation / Standardization.
- Outlier Detection and removal.
- Under/Over Sampling

Link to Graph and image snippets - AshTea Slide Deck

1. Dropping of Rows/Columns

a) If a column has 10,000 or lesser null values, we remove the rows.
b) If a column has 90 % null values, we drop the column.
c) If more than 90% of the data points in a column are skewed towards a single value, the column is dropped.
d) During feature selection, more non-required columns were dropped.

2. Imputation of NULL values

For columns having 10 - 50 % NULL values,

a) If a particular column has its 50th percentile value that is very close to the mean value, we replace the NULL values of that column with the mean.
b) If a column has its 50th percentile value far away from the mean value, then for numerical columns we use the median and for categorical columns we use the mode to fill in the empty cells.

For columns having 50-90% NULL values, we use imputation by regression, i.e fitting a statistical model on a variable with missing values. Predictions of this regression model are used to substitute the missing values in this variable. (eg. The columns having no null values are taken as reference, a linear regression model is fit to them, and the corresponding empty value cells are filled based on this regression pattern.)

3. Feature Selection

1) First, sets of similar features are grouped together (eg. C1 - C14, V237 - V245, etc.) and their numerical columns are considered. Categorical columns can't be considered in a correlation matrix to determine overlap with numerical columns.
2) Co - relation heat maps were drawn for these groups to identify similar columns. The features having either lesser null values, or less skew to a gaussian distribution (checked with histograms), or have a large overlap with most columns of the group were retained. Rest of them were dropped.

4. Categorical Value Grouping

Columns having similar categorical values were modified to reduce the number of unique values in the column. For eg. in the column 'id_30', the values 'Mac OS X 10_12_6', 'Mac OS X 10_11_6', 'Mac OS X 10_10_5', 'Mac OS X 10_13_2' were all replaced by 'Mac OS'.

5. Categorical to Numerical Conversion

1) For categorical columns having less than 10 unique values, we use 'One - Hot' encoding.
2) For categorical columns having greater than 10 unique values, we use 'Label' encoding.

6. Dealing with Outliers

Columns having outliers, identified through their box-plots, are analyzed for how much of their data points are outliers. For the columns having less than 5% outliers, if not many of them are 'Fraud', then the outliers are simply removed. In all other cases since we have very less number of 'Fraud' data points, we retain the outliers, and set their values to the 75th or 25th percentile values, whichever is closer.

7. Sampling

As the number of 'Fraud' values in the dataset are much lesser in comparison to the 'Not Fraud' values, we try to increase the number of 'Fraud' data points by generating new points by Over - Sampling. At the same time, we reduce the 'Not Fraud' values too. For the same, we use a Random Over Sampler and Random Under Sampler to get an 80:20 ratio between the two labels.

8. Standardization

In order to get all of our numerical data columns into a comparable range of values, we use standardization. In our case, we used Z-score normalization on all columns.

MODELS

After preprocessing, our dataset had 265 columns and 5,40,252 rows (train dataset). We fit this data set to multiple different classification models -

1. **XGBoost** - Model used for regularized Gradient boosting. For this model, GridSearchCV even with GPU usage frequently exceeded the Kaggle memory limit even on low parameters. Hence, we tried to logically do the hyperparameter tuning for this model. Here are some of the iterations of the parameters we evaluated the model on :

Num of estimators	Max depth	Learning rate	Sub Samples	Col Sample by tree	Val fit Accuracy	Test fit Accuracy
2000	12	0.02	0.8	0.4	99.9%	82.6%
300	12	0.01	0.3	0.8	96.4%	85.4%
1000	12	0.02	0.3	0.8	99.7%	83.6%
100	12	0.02	0.3	0.8	91.2%	84.7%
500	12	0.02	0.3	0.8	98.4%	86.2%
500	10	0.02	0.3	0.8	95.4%	86.5%
500	8	0.02	0.3	0.8	90.9%	86.3%
500	6	0.01	0.3	0.8	86.2%	84.4%
750	7	0.01	0.4	0.8	90.7%	86.0%
750	6	0.02	0.4	0.8	88.2%	85.7%
1000	7	0.02	0.4	0.8	91.6%	86.1%
1000	7	0.1	0.4	0.8	85.2%	81.1%
1000	7	0.02	0.9	0.7	95.0%	86.0%
800	7	0.02	0.9	0.8	95.3%	86.2%
1300	7	0.02	0.7	0.8	95.9%	86.4%
800	8	0.02	0.8	0.8	96.1%	86.7%

Best Score Parameters -
N_estimators = 800, max depth = 8, learning rate = 0.02, Sub Samples = 0.8, Col Sample by tree = 0.8

Score = 86.7% (without probabilities) and 95.2% (with probabilities)

2. **LGBM Classifier -** The size of the dataset is increasing rapidly. It has become very difficult for traditional data science algorithms to give accurate results. Light GBM is prefixed as Light because of its high speed. Light GBM can handle the large size of data and takes lower memory to run. We have done hyperparameter tuning using optuna as well as manually.

Optuna :

max_ depth	Num estimators	Num leaves	learning rate	lambda_l1	lambda_l2	min_data _in_leaf	Accuracy (on validation set) - stratified cv
9	1500	20	0.126	2	0	1000	0.88
6	1500	20	0.151	1	1	8000	0.83
11	1500	60	0.095	1	1	1000	0.95
5	1500	60	0.187	2	0	10000	0.79
8	1500	60	0.052	1	1	10000	0.75

Manual :

max_ depth	Num estimators	Num leaves	learning_rate	feature_fraction	subsample	Accuracy (non-proba train validation split {train,validation}
12	1000	31	0.1	0.8	0.2	(0.9170,0.9208)
20	1200	31	0.1	0.8	0.2	(0.9241.0.9288)
20	1200	40	0.1	0.8	0.4	(0.9389,0.9448)
20	1500	45	0.1	0.8	0.6	(0.9562,0.9631)
21	1500	50	0.1	0.8	0.7	(0.9618,0.9689)

Best Score Parameters -

N_estimators = 1500, max depth = 11, learning rate = 0.095, $\lambda 1 = 1$, $\lambda 2 = 1$, min_Data_In_Leaf = 1000

Score (With test dataset) = 91.92% (without probabilities) and 97.3% (with probabilities)

3. **Random Forests -** Random forests algorithms are used for classification and regression. The random forest is an ensemble learning method, composed of multiple decision trees. By averaging out the impact of several decision trees, random forests tend to improve prediction. Random forests tend to shine in scenarios where a model has a large number of features that individually have weak predictive power but much stronger power collectively.

For hyperparameter tuning we used GridSearchCV.

Parameters - {number of estimators, max depth}

- num_estimators - [100, 200, 300, 400, 500]
- max_depth - [2, 3, 4, 7, 9]

Best Score Parameters -
N_estimators = 400, max depth = 9

Score (With validation set) = 88.01% (without probabilities)

4. **Neural Networks -** Neural networks mimic the working of biological neurons, with a forward-only mechanism having nodes in layers and weights connecting the previous layers to the next ones. We use the help of Pytorch to implement our neural network model.

Following parameters were defined :

a) Number of Epochs - 50
b) Batch Size (data samples) - 64
c) Learning Rate - 0.001
d) Activation Function - Sigmoid

Score (With test set) = 80.29% (without probabilities)

5. **K Nearest Neighbours -** The K-nearest Neighbors (KNN) algorithm is a type of supervised machine learning algorithm used for classification, regression as well as outlier detection. It is extremely easy to implement in its most basic form but can perform fairly complex tasks. It is a lazy learning algorithm since it doesn't have a specialized training phase.

We defined our model to classify points into a class by setting the number of neighbors k to be 7. This model however is the slowest and toughest to modify and run for our large dataset.

Score (With train/validation set) = 97.2% (without probabilities)

6. **Gaussian Naive Bayes -** The Naive Bayes method makes the assumption that the predictors contribute equally and independently to selecting the output class. Although the Naive Bayes model's assumption that all predictors are independent of one another is unfeasible in real-world circumstances, this assumption produces a satisfactory outcome in the majority of instances.

Score (With train/validation set) = 62.2% (without probabilities)