Project Description

In this project I used the Kaggle Creditcard Fraud data to determine whether the transaction is fraud or not.

Assumptions:

- Here we have data for just two days, but we assume the data is representative of the whole population of credit card transactions.
- We assume the 28 variables V1-V28 are obtained from correct method of PCA and are scaled properly.

Metric Used

- Here 1 means fraud and 0 means no fraud.
- For this imbalanced dataset, false negative (fraud classified as not fraud) is more important than false positive (not-fraud classified as fraud), so we use `Recall` as the metric of evaluation. (`Recall = TP / (TP + FN)`).
- For the imbalanced dataset, AUCROC gives overly optimistic metric, instead we should use `precision_recall_curve` and after looking at the curve we should choose the value that we want for precision and recall.
- We should also note that precision and recall does not involve TN, so we should use them only when specificity (TNR = TN/(TN+FP)) is not important.
- For imbalanced dataset, we can use F_beta metric. If both precision and recall are equally important, we can use F1-score. If we consider recall beta times more important than precision, we can use `F_beta = (1+beta^2) PR/(beta^P + R)` where P is precision and R is recall. (Mnemonic: Look at the denominator and remember that Recall is beta^2 time important than Precision). (Common values are 2 and 0.5. If beta is 2, recall is twice important than precision.)
- We should also note that F_beta depends on Precision and Recall only. It does not depend on TN (true negative), so for imbalanced classification, better metric could be MCC (Mathew's Correlation Coefficient.)

Resampling Techniques

- Our dataset is imbalanced, we can try two sampling: undersampling and oversampling.
- Under-sampling. (We have low number of frauds, choose randomly same number of non-frauds.)
- Oversampling `SMOTE` method. Used external library `imblearn`.

Best Model So Far

Model	Description	Accuracy	Precision	Recall	F1	AUC	Untrue Frauds	Missed Frauds
keras	1 layer, class_weight, early_stopping, scikit api	0.987939	0.111989	0.867347	0.198366	0.927747	674	13
cb_tuned pycaret	fold=5	0.9996	0.9659	0.7865	0.9667	0.8642		
catboost	seed=100,depth=6,iter=1k	0.999631	1.000000	0.785714	0.880000	0.892857	0	21

Undersampling

Recall for all Classifiers with Grid Search for Undersampled Data

	Model	Description	Accuracy	Precision	Recall	F1	Mathews Correlation Coefficient	Cohens Kappa	Area Under Precision Curve	Area Under ROC Curve
0	Logistic Regression	Undersample, Grid Search	0.741117	0.651515	0.945055	0.7713	0.541913	0.495303	0.95003	0.931474
1	Support Vector Classifier	Undersample, Grid Search	0.741117	0.651515	0.945055	0.7713	0.541913	0.495303	0.962546	0.94464
2	Random Forest Classifier	Undersample, Grid Search	0.913706	0.95122	0.857143	0.901734	0.828738	0.825181	0.981182	0.977918
3	KNN	Undersample, Grid Search	0.888325	0.915663	0.835165	0.873563	0.776569	0.773941	0.903222	0.93163
4	Decision Tree Classifier	Undersample, Grid Search	0.898477	0.949367	0.824176	0.882353	0.79999	0.793847	0.903238	0.931371

Logistic Regression, Undersample, Grid Search

	Predicted_No_Fraud	Predicted_Fraud	Total_Frauds	Correct_Frauds	Incorrect_Frauds	Fraud_Detection
No_Fraud	60	46	91	86	5	94.51%
Fraud	5	86	91	86	5	94.51%

SMOTE Oversampling: Logistic Regression

	Model	Description	Accuracy	Precision	Recall	F1	Mathews Correlation Coefficient	Cohens Kappa	Area Under Precision Curve	Area Under ROC Curve
0	Logistic Regression	Train Test Undersample, Grid Search	0.923858	0.910891	0.938776	0.924623	0.848129	0.847735	0.985611	0.97918
1	Logistic Regression	Train Test Undersample	0.944162	0.957895	0.928571	0.943005	0.888717	0.888305	0.991175	0.989281
2	Logistic Regression	Train Oversample SMOTE, Test Imbalanced	0.796831	0.00029432	0.428571	0.000588235	0.00661815	0.00030949	0.00142026	0.715063
3	Logistic Regression	Train Oversample SMOTE, Test Imbalanced, Grid Search from Undersample	0.875775	0.000481386	0.428571	0.000961693	0.0109009	0.000683173	0.00369781	0.730713
4	Logistic Regression Polynomial deg 2	Train Oversample SMOTE, Test Imbalanced, Grid Search from Undersample	0.875775	0.000481386	0.428571	0.000961693	0.0109009	0.000683173	0.00369781	0.730713
5	Logistic Regression	Train Test Imbalanced	0.99986	0	0	0	0	0	0.00305379	0.624277
6	Logistic Regression	Train Test Imbalanced, Grid Search	0.99986	0	0	0	0	0	0.000155879	0.466225
7	Logistic Regression	Train Undersample, Test Imbalanced	0.999482	0	0	0	-0.000229906	-0.000203943	0.000139512	0.492705

Logistic Regression Oversampling SMOTE Grid Search from Undersampling

	Predicted_No_Fraud	Predicted_Fraud	Total_Frauds	Correct_Frauds	Incorrect_Frauds	Fraud_Detection
No_Fraud	43,939	6,229	7	3	4	42.86%
Fraud	4	3	7	3	4	42.86%

Anomaly Detection Methods

Model	Description	Accuracy	Precision	Recall	F1(Weighted)
Isolation Forest	default	0.997384	0.261682	0.285714	0.997442
Local Outlier Factor	default	0.996331	0.025641	0.030612	0.996493

Gradient Boosting Modelling

Model	Description	Accuracy	Precision	Recall	F1	AUC
lightgbm	grid search optuna	0.999315	0.873418	0.704082	0.779661	0.851953
lightgbm	default	0.997367	0.275862	0.326531	0.299065	0.662527
Xgboost	default, imbalanced	0.999263	0.850000	0.693878	0.764045	0.846833
Xgboost	default, undersampling	0.999263	0.850000	0.693878	0.764045	0.846833
Xgboost	n_estimators=150, imbalanced	0.999263	0.850000	0.693878	0.764045	0.846833
Xgboost	undersample, hpo1	0.999298	0.881579	0.683673	0.770115	0.841758
Xgboost	imbalanced, hpo	0.999245	0.898551	0.632653	0.742515	0.816265
xgboost	grid search optuna	0.999333	0.875000	0.714286	0.786517	0.857055
catboost	seed=100,depth=6,iter=1k	0.999631	1.000000	0.785714	0.880000	0.892857

Automatic Modelling: pycaret

Model	Description	Accuracy	AUC	Recall	Precision	F1	Kappa
cb_tuned	fold=5	0.9996	0.9659	0.7865	0.9667	0.8642	0.8639
lda_tuned	fold=5	0.9995	0.9833	0.7760	0.9217	0.8423	0.8420
xgb	default	0.9994	0.9585	0.7345	0.9102	0.8047	0.8044
cb	default	0.9995	0.9554	0.7345	0.9548	0.8215	0.8212
lda	default	0.9992	0.9677	0.7255	0.8340	0.7661	0.7657
xgb_tuned	tuned	0.9992	0.9677	0.7255	0.8340	0.7661	0.7657
lda_tuned	n_iter=100,fold=10	0.9992	0.9677	0.7255	0.8340	0.7661	0.7657

Big Data Modelling: PySpark

	model_name	desc	f1	weightedPrecision	weightedRecall	accuracy	areaUnderROC	areaUnderPR
0	Logistic Regression	log features	0.846066	0.730487	0.999300	0.999294	0.999337	0.999337
3	Decision Tree Classifier	log features	0.868053	0.765536	0.999398	0.999393	0.999424	0.999424
4	Random Forest Classifier	log features	0.890040	0.800147	0.999494	0.999490	0.999511	0.999511
6	Random Forest Classifier	log features, grid search	0.884580	0.836982	0.999541	0.999548	0.999564	0.999564

Deep Learning Models

Model	Description	Accuracy	Precision	Recall	F1	AUC	Missed Frauds	Untrue Frauds
keras	3 layers, 2 dropouts, class_weight	0.983744	0.081818	0.826531	0.148897	0.905273	17	909
keras	1 layer, dropout, early_stopping	0.984990	0.090811	0.857143	0.164223	0.921177	14	841
keras	1 layer, dropout, steps_per_epoch, oversampling	0.982796	0.080000	0.857143	0.146341	0.920077	14	966
keras	1 layer, class_weight, early_stopping, scikit api	0.987939	0.111989	0.867347	0.198366	0.927747	13	674

References

- https://www.tensorflow.org/tutorials/structured_data/imbalanced_data
- https://keras.io/examples/structured_data/imbalanced_classification/
- https://www.kaggle.com/residentmario/using-keras-models-with-scikit-learn-pipelines#
- https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/