CREDIT CARD FRAUD DETECTION

- PINAKI BHAGAT
- SYDNEY CORREA

Learning Unbalanced Data

Agenda

- Credit Card use case overview
- Dataset description
- Techniques: Traditional/Deep Learning/ML Platforms
- Findings
- Future enhancements

Overview

Credit card fraud detection (Learning unbalanced data)

- Annual global fraud losses reached $21.8 billion in 2015.
- Approx 12 cents per $100 were stolen in the US during the same year
- Future trends show moving toward fraud prediction (and prevention) rather than fraud detection
- The cost of false positive fraud labelling was $118 billion dollars of which $9 billion were actual fraud cases
- Trends are moving from more traditional approaches to deep learning approaches (including hybrids w/ graph dbs)

Source: experoinc.com

Project objective

Objective: Try different approaches to compare model performance

A. Traditional methods

a. K-means, Random Forest

b. Ensemble Learning: Random forest, ExtraTrees, AdaBoost, GradientBoost, XGBoost

B. Deep Learning

a. Autoencoders

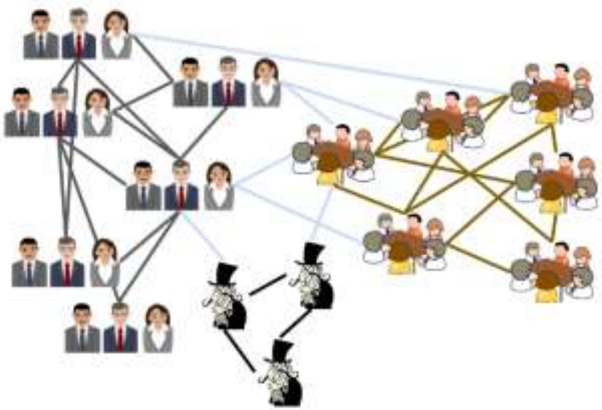
C. ML Platforms

a. h2o.ai

Research some of the leading approaches and cutting edge technologies

Credit card Dataset

The creditcard.csv datasets is one of 4 datasets on Kaggle and contains transactions made by credit cards in September 2013 by European cardholders.



Features:

-transactions that occurred in two days

-contains 492 frauds out of 284,807 transactions

-highly unbalanced (frauds account for 0.172% of all transactions)

-dataset contains only numerical input variables which are the result of a PCA transformation.

-features "Time" and "Amount" are not transformed

Sources: experoinc.com; Kaggle.com

Techniques: KNN, RF

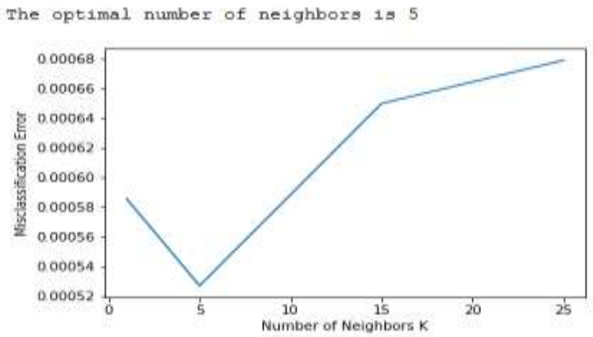
Classification problem using shallow algorithms to train individual models using KNN and RF

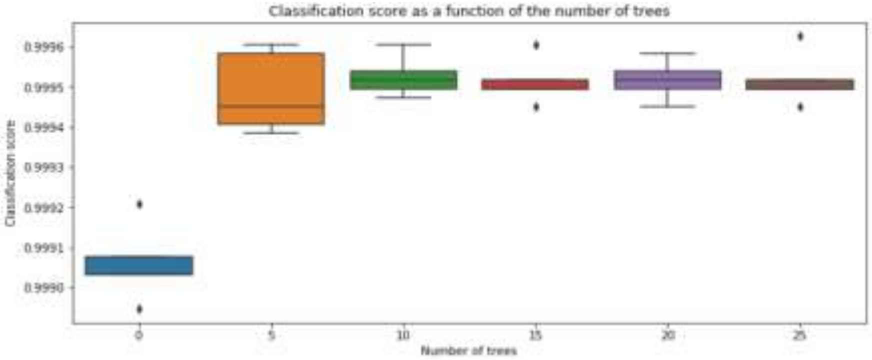
Checked for correlation between features

Hyper tuned the RF model for n_estimators (n=16)

Ran KNN for different intervals (k=5)

Checked model performance using:
-AUPRC (area under precision recall curve)
-ROC
-Confusion matrix





Techniques: Ensemble

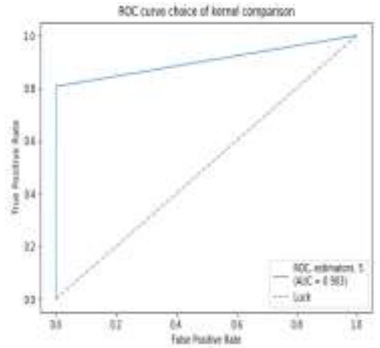
Following algorithms were used to train the model stack:
a. Random Forest
b. Extra Trees
c. AdaBoost
d. GradientBoost

Co-relation between models were checked and extra trees was eliminated

Ensemble had XGBoost as the final model

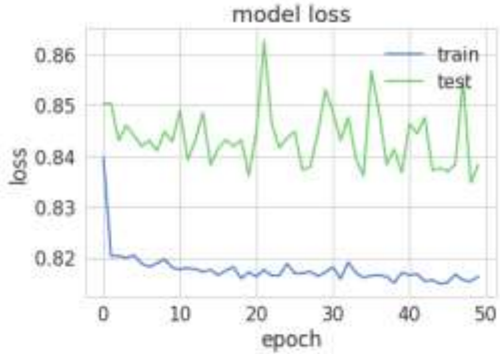
```
Accuracy score:  99.95%

Classification report:
             precision    recall  f1-score   support

   Output 0       1.00      1.00      1.00    113726
   Output 1       0.93      0.76      0.83       197

avg / total       1.00      1.00      1.00    113923
```

	AdaBoost	ExtraTrees	GradientBoost	RandomForest
0	0.278610	0.000171	0.000346	0.000128
1	0.042268	0.008257	0.000353	0.000136
2	0.280778	0.000221	0.000360	0.000144
3	0.291032	0.000237	0.000376	0.000304
4	0.276816	0.000298	0.000363	0.000098



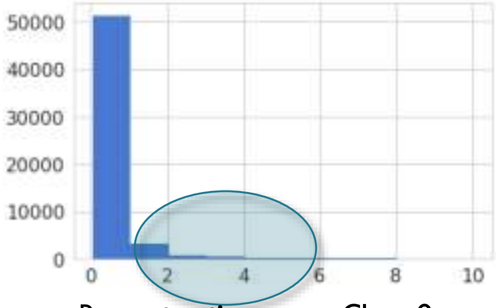
Techniques: Autoencoders

Anomaly detection by training the network on non-fraudulent transactions, fraudulent transactions to be detected by flagging them based on higher than normal MSE values (reconstruction error)

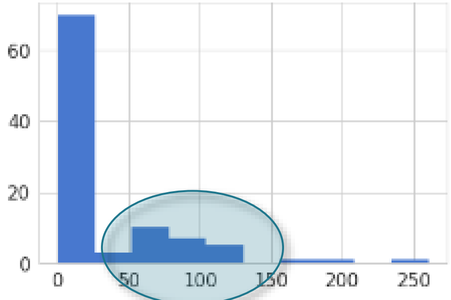


Features:
- 4 fully connected layers (two encoders and 2 decoders)
- Activation filters: tanh, relu

Reconstruction error for Autoencoder: $L(x,x') = ||x - x'||^2$



Reconstruction error: Class 0



Reconstruction error: Mixed

Techniques: h2o.ai

We explored using the popular H2O.ai platform, specifically the AutoML package.

AutoML packages – Random Forest (DRF, XRT), XGBoost GBM, Deep Neural Net, …

Hyperparameter tuning – max_models (5,10,15), balancing

```
aml = H2OAutoML(max_models=10, seed=1234, balance_classes = True, max_after_balance_size=0.8)
aml.train(x = predictors, y = response, training_frame = train, validation_frame = valid)
```

model_id	auc	logloss	mean_per_class_error	rmse	mse
XGBoost_1_AutoML_20181219_234301	0.984103	0.00250716	0.101853	0.0200578	0.000402316
GLM_grid_1_AutoML_20181219_234301_model_1	0.973893	0.00411683	0.100894	0.0265999	0.000707553
XGBoost_2_AutoML_20181219_234301	0.971338	0.00324639	0.10298	0.0203073	0.000412386
DRF_1_AutoML_20181219_234301	0.956017	0.00393489	0.0894407	0.0207101	0.000428908
XRT_1_AutoML_20181219_234301	0.952861	0.00359792	0.10298	0.0206141	0.000424941
StackedEnsemble_BestOfFamily_AutoML_20181219_234301	0.94077	0.00311164	0.0950754	0.0207072	0.000428788
StackedEnsemble_AllModels_AutoML_20181219_234301	0.93763	0.00312438	0.09733	0.020708	0.000428822

Sources: h2o.ai

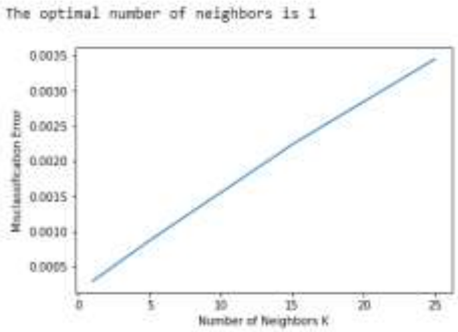
Techniques: Balancing

Considering the nature of dataset transactions to be unbalanced (0.172%) we tried multiple under sampling and oversampling techniques

Packages tried:

Oversampling - imblearn (SMOTE, SMOTEEEN);

Undersampling – RandomUnderSampler, H2o.ai



Models tried: KNN and RF and h2o AutoML

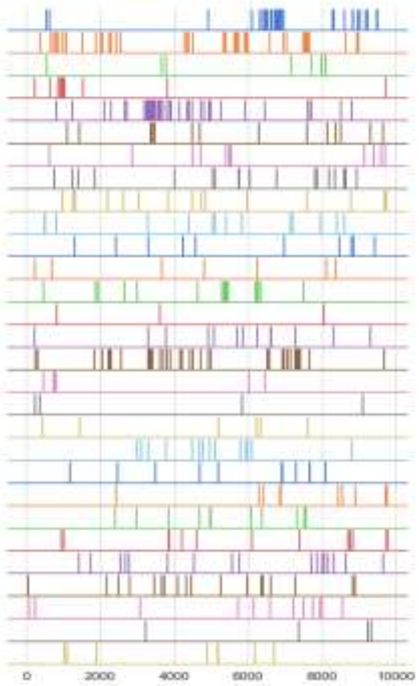
Does not apply to Autoencoders and

Outcomes: Model performance was significantly worse with balanced data.

Pic Source: freepik.com

Findings

Fraud transactions: ridge line graph using seaborn



Future enhancements

Citation & Useful links

- https://www.slideshare.net/databricks/deep-learning-for-recommender-systems-with-nick-pentreath
- https://github.com/maciejkula/spotlight/tree/master/examples/movielens_explicit
- https://recsys.acm.org/recsys18/tutorials/#content-tab-1-0-tab
- https://code.fb.com/core-data/recommending-items-to-more-than-a-billion-people/
- https://www.fast.ai/2017/07/28/deep-learning-part-two-launch/
- https://www.ethanrosenthal.com/2015/11/02/intro-to-collaborative-filtering/
- https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0
- https://www.youtube.com/watch?v=h9gpufJFF-0
- https://en.wikipedia.org/wiki/Collaborative_filtering