CREDIT CARD FRAUD DETECTION USING MACHINE LEARNING MODELS

PRESENTED BY:

KRISHNA RAVALI JAMMALAMADAKA(19MBMB12)

INTRODUCTION

Credit card fraud is a type of identity theft when some unintended user uses a credit card or credit account to make a transaction. Ways in which this can happen are:

- A misplaced credit card

- When someone gets hold of other's account number, pin and security code

- Skimming POS data to make duplicate card

- Hackers using online portals or unsecure imposter websites to hack user information leading to data breach causing card-not-present/CNP fraud

- Fraudulent application for credit card (identity theft)

- Cards stolen from mailboxes

MOTIVATION

CREDIT CARD FRAUD STATISTICS

By J Krishna Ravali

https://spd.group/machine-learning/credit-card-fraud-detection/

03-12-2020
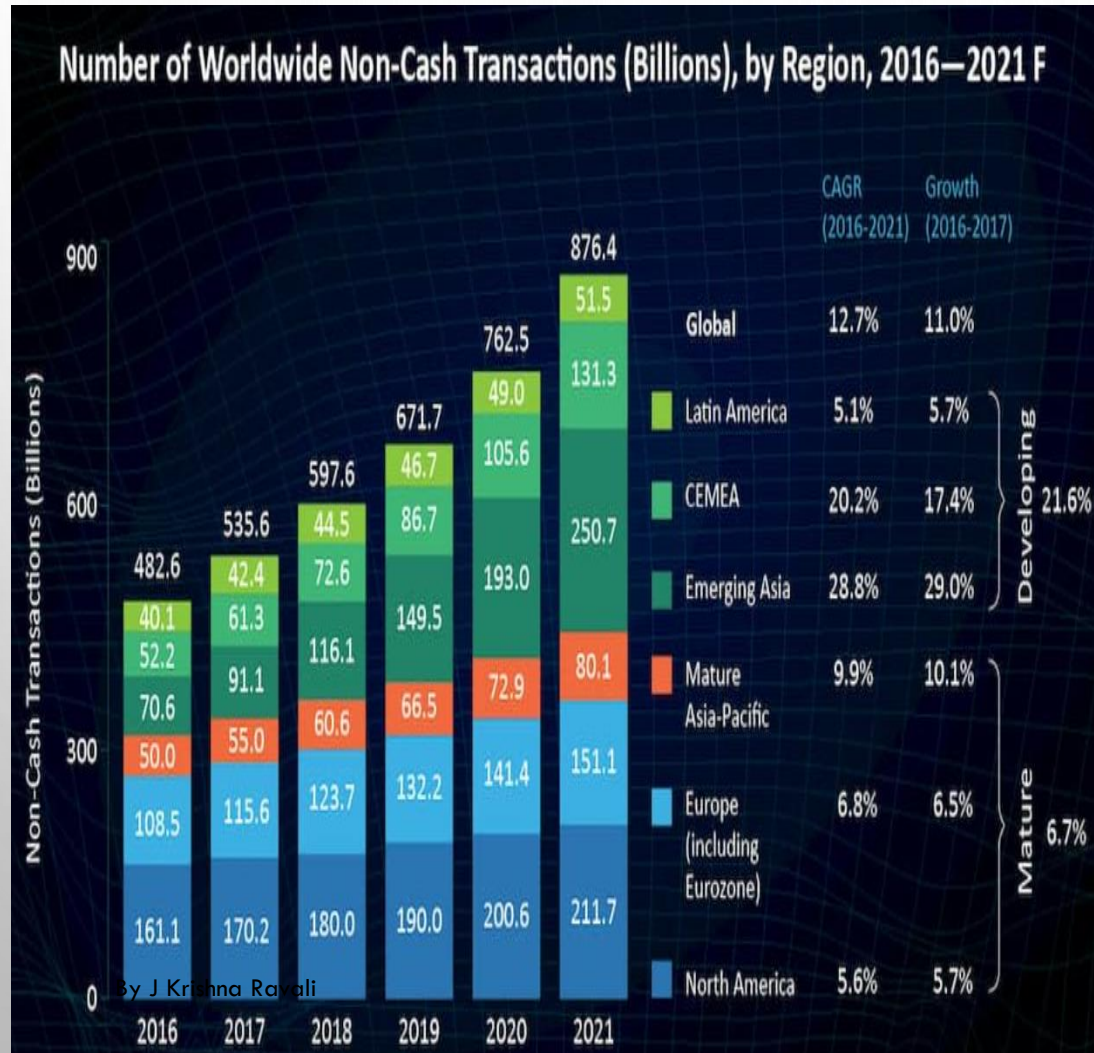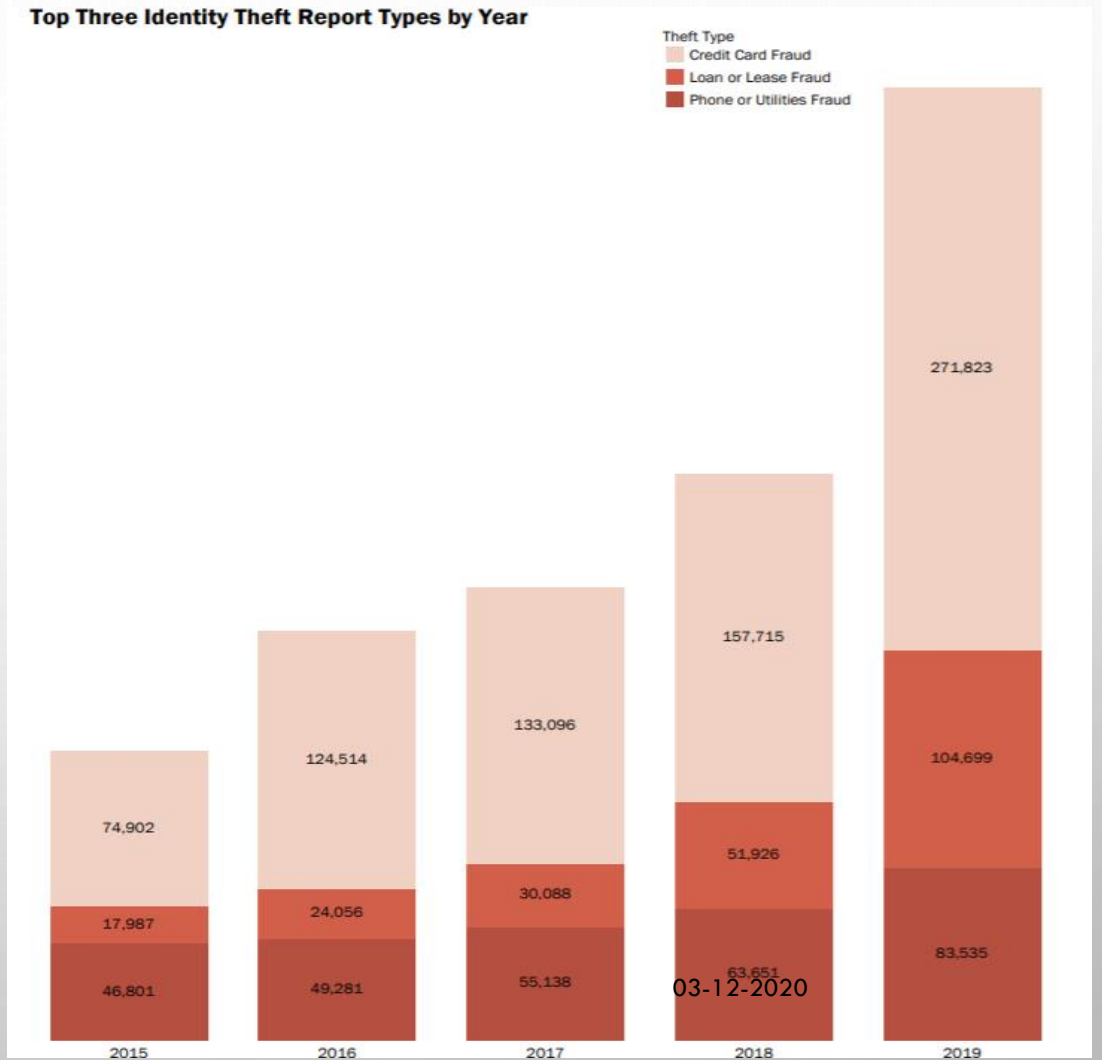
https://www.ftc.gov/system/files/documents/reports/consumer-sentinel-network-data-book-2019/consumer_sentinel_network_data_book_2019.pdf

RATIONALE FOR MACHINE LEARNING BASED FRAUD DETECTION

Traditional method of fraud detection	Machine Learning-based Fraud Detection
The rules for decision making are set manually	Automatic fraud detection
Takes lot of time	Ability to detect frauds in real time
Inconvenient for the user, since it needs multiple verifications	Low verification time
Detects only obvious frauds	Identifies hidden data patterns and correlations

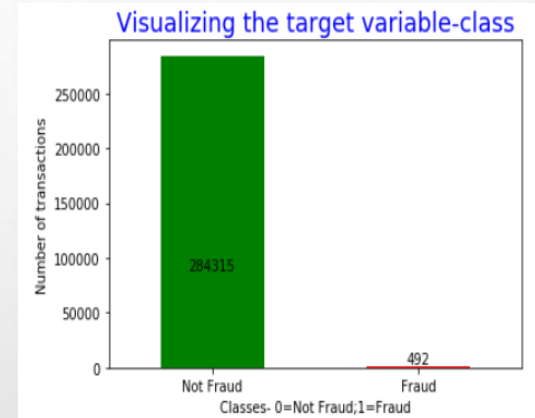# CHALLENGES IN CREDIT FRAUD DETECTION

- Unavailability of real data

- Skewed data

- Size of the data

- Choosing appropriate metric

- Random patterns in frauds

PROBLEM STATEMENT

- IDENTIFY FRAUDULENT CREDIT CARD TRANSACTIONS.

- GIVEN THE CLASS IMBALANCE RATIO, MEASURING THE MODEL PERFORMANCE IN TERMS OF SENSITIVITY AND SPECIFICITY.

DATA DESCRIPTION

- Source: https://www.kaggle.com/mlg-ulb/creditcardfraud

- It contains the details of credit card transactions of European card holders in 2013 that occurred in two days

- Due to privacy reasons, data is completely encrypted for the fields from v1-v28 by the data provider

- Dataset has 31 fields in total. Predictors V1-V28, time, amount

- Target field is 'class' which is binomial in nature 1(fraud) 0(not fraud)

- Total transactions=284807; out of which only 492 records are fraudulent transactions.

Visualizing the target variable-class

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
2	0	-1.35981	-0.07278	2.53635	1.37816	-0.33832	0.46239	0.2396	0.0987	0.36379	0.09079	-0.5516	-0.6178	-0.99139	-0.31117	1.46818	-0.4704	0.20797	0.02579	0.40399	0.25141	-0.01831	0.27784	-0.11047	0.06693	0.12854	-0.18911	0.13356	-0.02105	149.62	0
3	0	1.19186	0.26615	0.16648	0.44815	0.06002	-0.08236	-0.0788	0.0851	-0.25543	-0.16697	1.61273	1.06524	0.4891	-0.14377	0.63556	0.46392	-0.1148	-0.18336	-0.14578	-0.06908	-0.22578	-0.63867	0.10129	-0.33985	0.16717	0.12589	-0.00898	0.01472	2.69	0
4	1	-1.35835	-1.34016	1.77321	0.37978	-0.5032	1.8005	0.79146	0.24768	-1.51465	0.20764	0.6245	0.06608	0.71729	-0.16595	2.34586	-2.89008	1.10997	-0.12136	-2.26186	0.52498	0.248	0.77168	0.90941	-0.68928	-0.32764	-0.1391	-0.05535	-0.05975	378.66	0
5	1	-0.96627	-0.18523	1.79299	-0.86329	-0.01031	1.2472	0.23761	0.37744	-1.38702	-0.05495	-0.22649	0.17823	0.50776	-0.28792	-0.63142	-1.05965	-0.68409	1.96578	-1.23262	-0.20804	-0.1083	0.00527	-0.19032	-1.17558	0.64738	-0.22193	0.06272	0.06146	123.5	0
6	2	-1.15823	0.87774	1.54872	0.40303	-0.40719	0.09592	0.59294	-0.27053	0.81774	0.75307	-0.82284	0.5382	1.34585	-1.11967	0.17512	-0.45145	-0.23703	-0.03819	0.80349	0.40854	-0.00943	0.79828	-0.13746	0.14127	-0.20601	0.50229	0.21942	0.21515	69.99	0
7	2	-0.42597	0.96052	1.14111	-0.16825	0.42099	-0.02973	0.4762	0.26031	-0.56867	-0.37141	1.34126	0.35989	-0.35809	-0.13713	0.51762	0.40173	-0.05813	0.06865	-0.03319	0.08497	-0.20825	-0.55982	-0.0264	-0.37143	-0.23279	0.10591	0.25384	0.08108	3.67	0
8	4	1.22966	0.141	0.04537	1.20261	0.19188	0.27271	-0.00516	0.08121	0.46496	-0.09925	-1.41691	-0.15383	-0.75106	0.16737	0.05014	-0.44359	0.00282	-0.61199	-0.04558	-0.21963	-0.16772	-0.27071	-0.1541	-0.78006	0.75014	-0.25741	0.03451	0.00517	4.99	0
9	7	-0.64427	1.41796	1.07438	-0.4922	0.94893	0.42812	1.12063	-3.80786	0.61537	1.24938	-0.61947	0.29147	1.75796	-1.32387	0.68613	-0.07613	-1.22213	-0.35822	0.3245	-0.15674	1.94347	-1.01545	0.0575	-0.64971	-0.41527	-0.05163	-1.20692	-1.08534	40.8	0
10	7	-0.89429	0.28616	-0.11319	-0.27153	2.6696	3.72182	0.37015	0.85108	-0.39205	-0.41043	-0.70512	-0.11045	-0.28625	0.07436	-0.32878	-0.21008	-0.49977	0.11876	0.57033	0.05274	-0.07343	-0.26809	-0.20423	1.01159	0.3732	-0.38416	0.01175	0.1424	93.2	0
11	9	-0.33826	1.11959	1.04437	-0.22219	0.49936	-0.24676	0.65158	0.06954	-0.73673	-0.36685	1.01761	0.83639	1.00684	-0.44352	0.15022	0.73945	-0.54098	0.47668	0.45177	0.20371	-0.24691	-0.63375	-0.12079	-0.38505	-0.06973	0.0942	0.24622	0.08308	3.68	0
12	10	1.44904	-1.17634	0.91386	-1.37567	-1.97138	-0.62915	-1.42324	0.04846	-1.72041	1.62666	1.19964	-0.67144	-0.51395	-0.09505	0.23093	0.03197	0.25341	0.85434	-0.22137	-0.38723	-0.0093	0.31389	0.02774	0.50051	0.25137	-0.12948	0.04285	0.01625	7.8	0

By J Krishna Rayali

03-12-2020

EXPLORATORY DATA ANALYSIS

- Summary of target class

```
frauds.Amount.describe()

count      492.000000
mean       122.211321
std        256.683288
min          0.000000
25%          1.000000
50%          9.250000
75%        105.890000
max       2125.870000
Name: Amount, dtype: float64
```

```
nonfrauds.Amount.describe()

count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```


- Average amount involved in fraud transactions in this dataset is slightly higher than that of normal transactions

- Maximum amount involved in normal transaction is huge when compared to that of fraudulent transactions

- Correlation matrix:
  - Input variable are not correlated in the dataset, which implies data is good enough to proceed with data modelling since there is no multicollinearity.
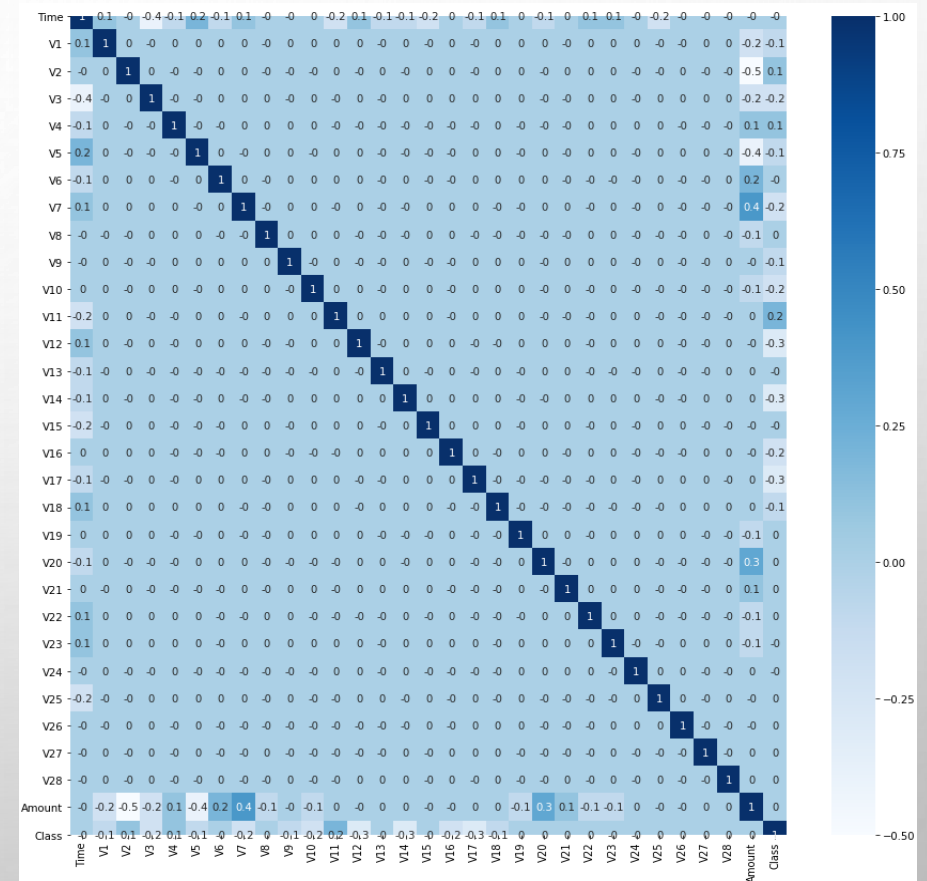
contd…

- Fraud cases are seen over entire period of two days, so this field doesn't add much to the analysis, hence this column can be dropped

- More number of transactions are observed with lesser amounts and vice versa.

By J Krishna Ravali

03-12-2020
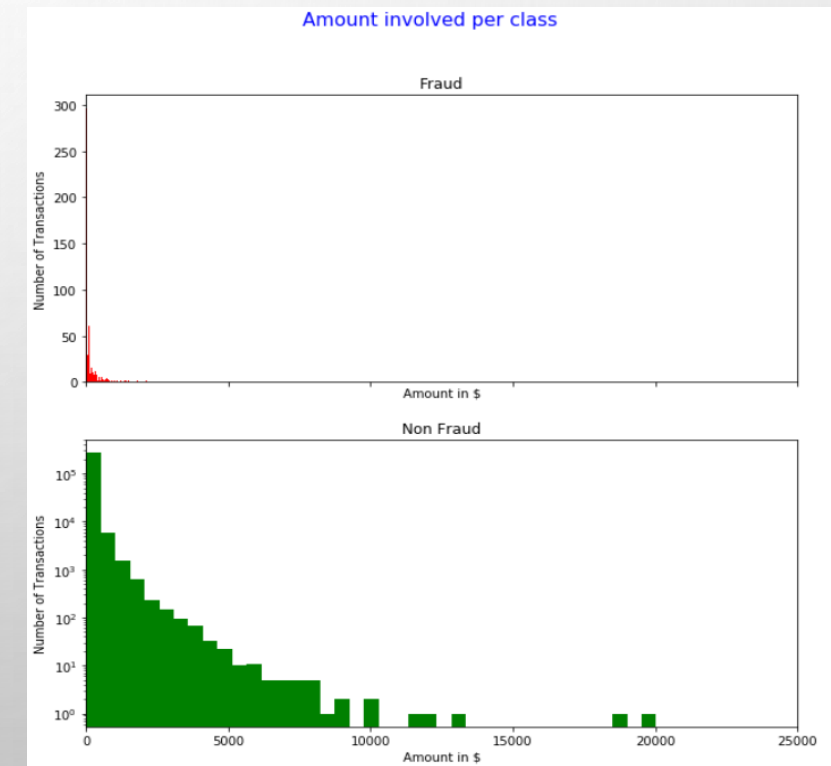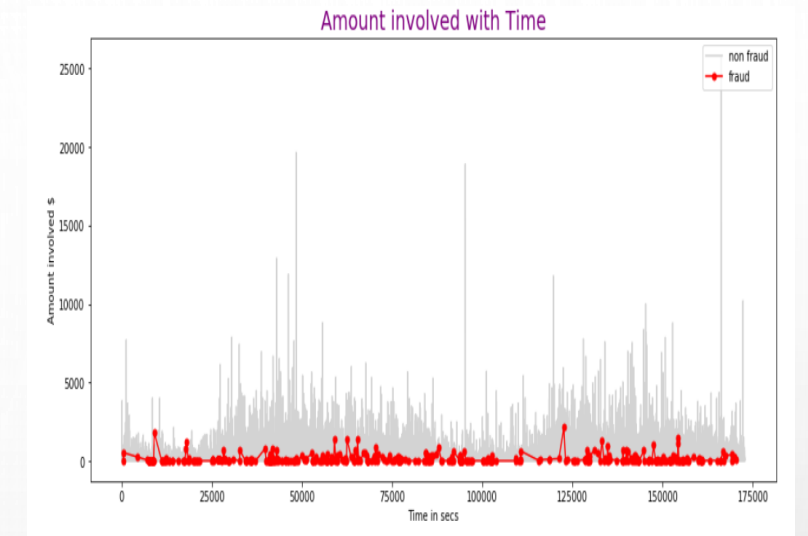
TRAIN-TEST SPLIT

- **Handling imbalanced data**: adaptive synthetic sampling (ADASYN) technique is used which generates synthetic samples inversely proportional to the density of minority class i.e., More synthetic data is generated for minority class samples that are harder to learn compared to those minority samples that are easier to learn. It helps to maintain the proportion of majority and minority class in train and test data after split.

```
Original dataset shape Counter({0: 284315, 1: 492})
Resampled dataset shape Counter({0: 284315, 1: 284240})
```

- **Data split**: Data is split in to 80:20 ratio for training and testing respectively using train_test_split() function.

```
train dataset shape Counter({0: 227562, 1: 227282})
test dataset shape Counter({1: 56958, 0: 56753})
```

SUPERVISED LEARNING

Three models are tested and compared under this segment: **Logistic Regression, Naïve Bayes, KNN**
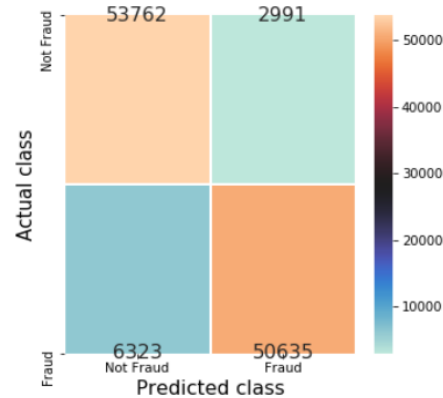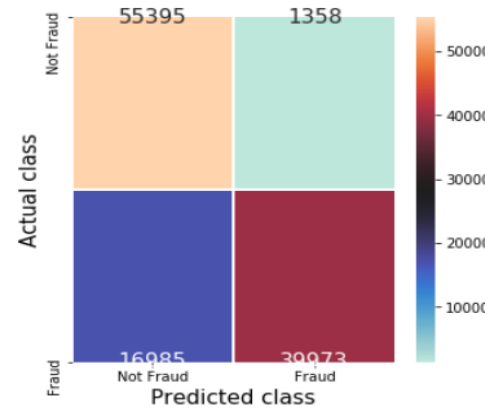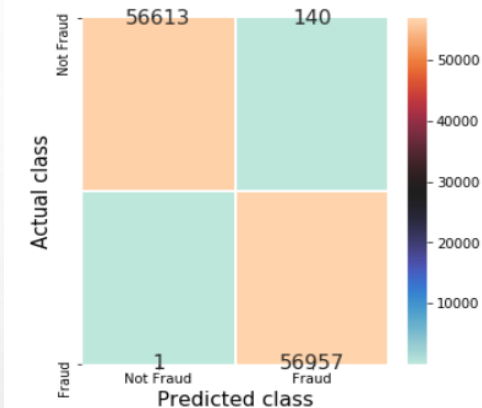
Results are validated through stratified cross validation with different variants taking cv folds=5,10
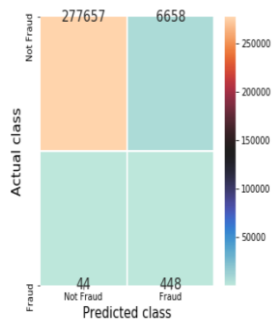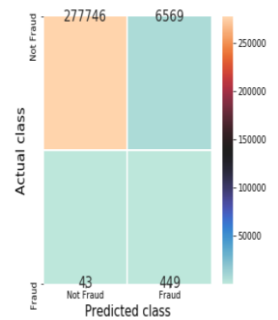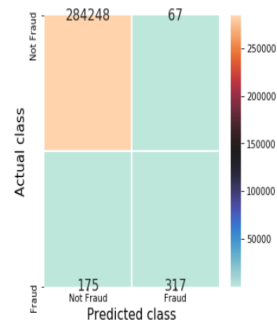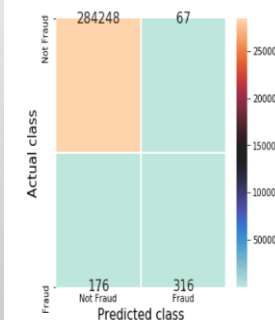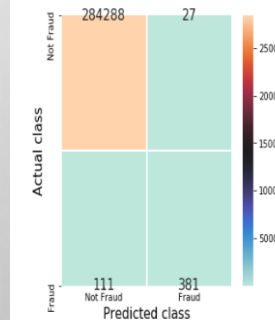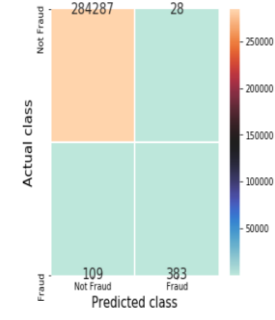
UNSUPERVISED MODELS

- Two models are tested and compared under this segment: **Mini Batch K-Means, K-Means**

- Number of segments/clusters in the data are chosen using elbow curves (n = 2)

- Mini Batch K-Means is faster than K-Means as the data size increases

- Both the models have given same performance on this dataset

- Total FPs = 32 FNs = 239

- Even, in case of unsupervised learning, model is able to detect around half of the fraudulent cases correctly, by further tuning, models can be deployed for prediction along with detection in case of new frauds.

https://www.geeksforgeeks.org/ml-mini-batch-k-means-clustering-algorithm/

By J Krishna Ravali
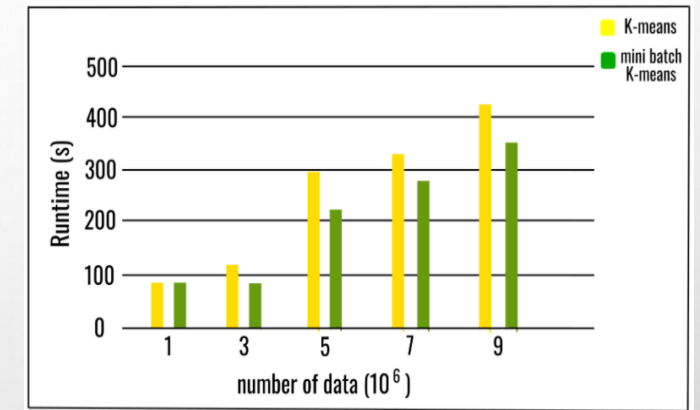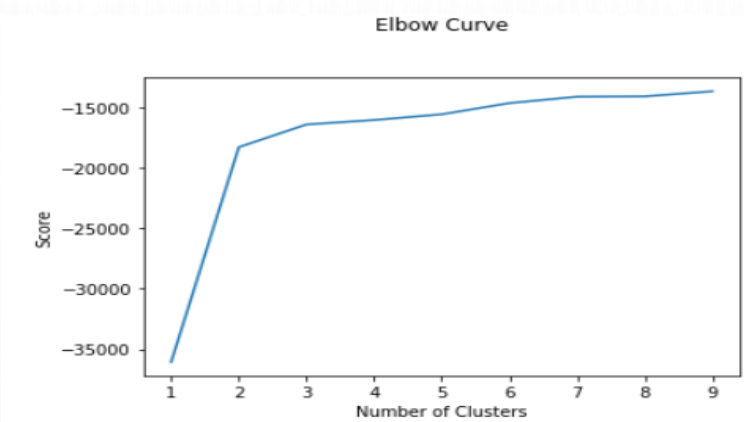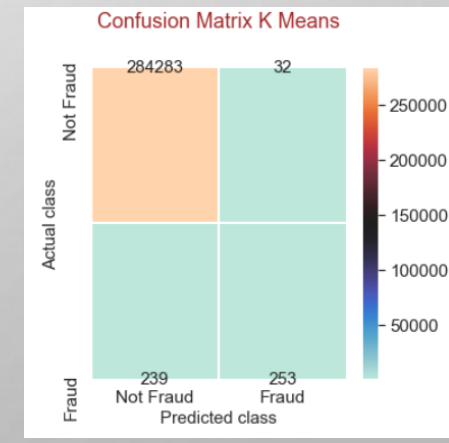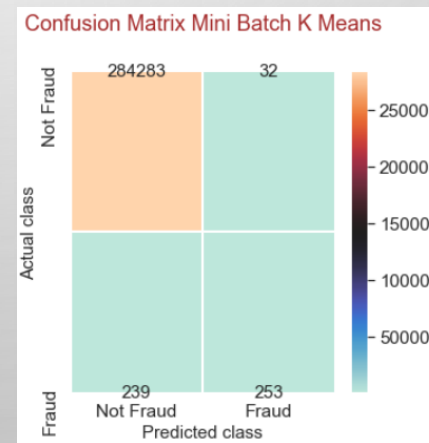
AUTO ENCODERS (ARTIFICIAL NEURAL NETWORKS)

- Auto encoders are usually trained with normal data and are fed with anomalous or abnormal data during validation phase which will give higher reconstruction error when compared with normal data which helps to detect/identify them.

- Optimizer- adam *(ADAPTIVE MOMENT ESTIMATION)* fast, low memory requirement, computationally efficient. Adam is a stochastic gradient descent method that computes individual adaptive learning rates for different parameters from estimates of first- and second-order moments of the gradients

- Batch size:32 epochs:100.

By J Krishna Ravali

Histogram of Normal transactions

Histogram of Fradulent transactions
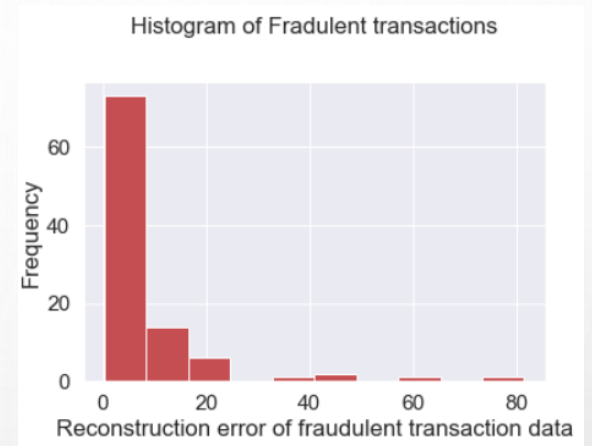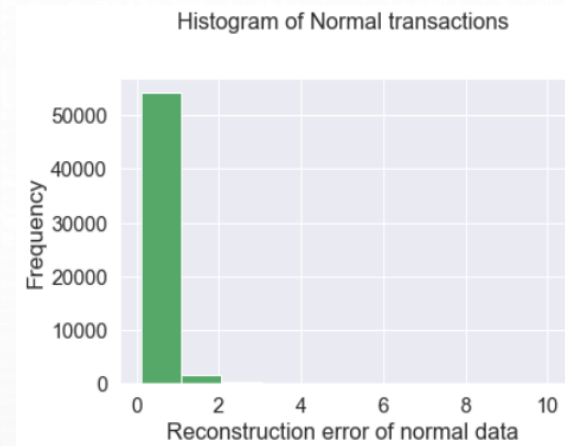
normal class

	reconstruction_error	true_class
count	56864.000000	56864.0
mean	0.556202	0.0
std	1.696124	0.0
min	0.096891	0.0
25%	0.285996	0.0
50%	0.391433	0.0
75%	0.559599	0.0
max	124.325271	0.0

fraud class

|	reconstruction_error	true_class
count	98.000000	98.0
mean	6.764667	1.0
std	12.660082	0.0
min	0.270467	1.0
25%	0.897144	1.0
50%	1.955423	1.0
75%	7.880527	1.0
max	81.554176	1.0

contd…..
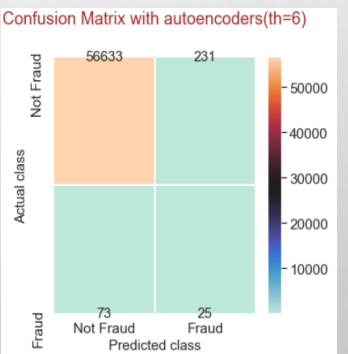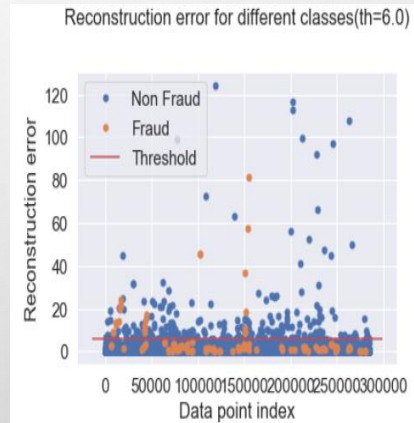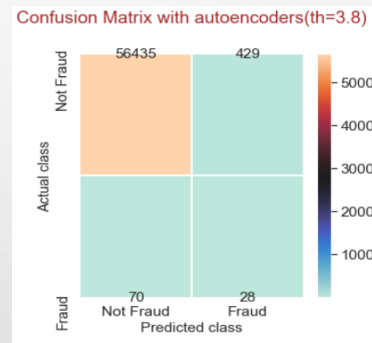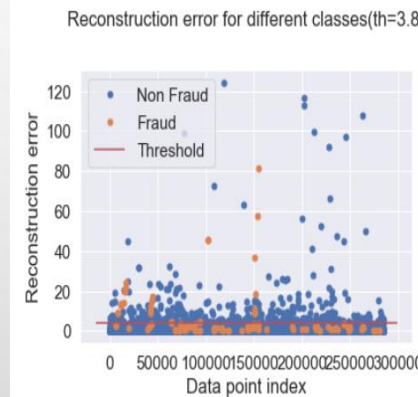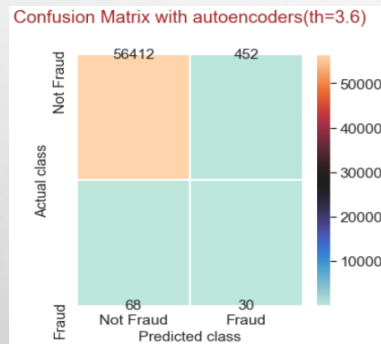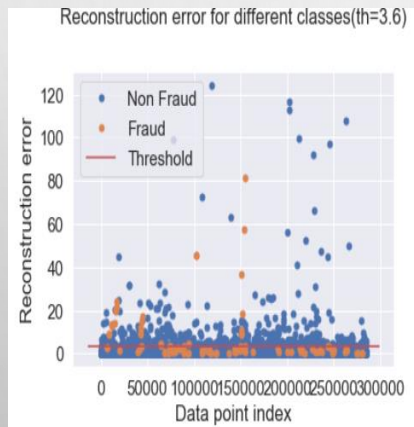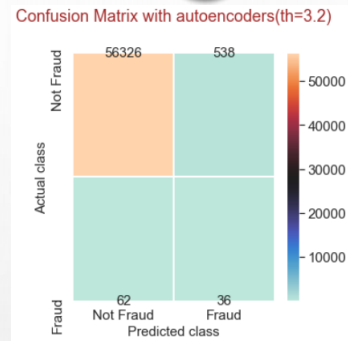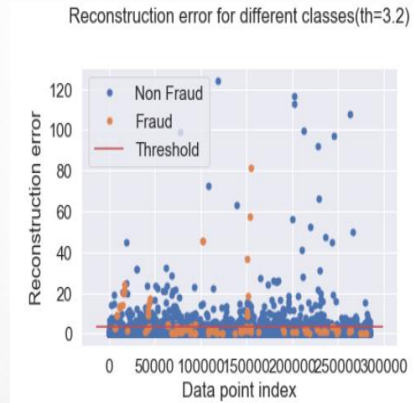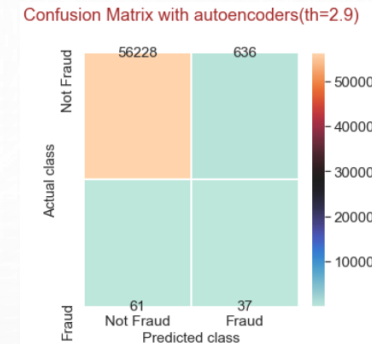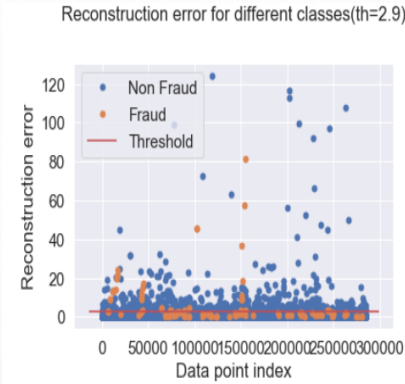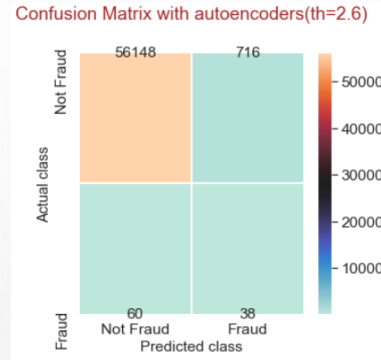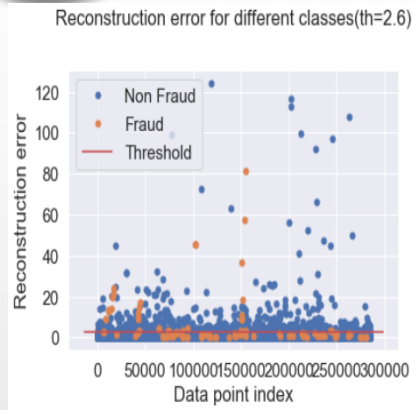
- Comparison of model performance for difference reconstruction error thresholds

- As threshold rises (th=2.6 to 6.0), count of FNs increased, count of FPs reduced

- Thence an optimum threshold value can be set depending upon organization's objective

ENSEMBLED METHODS

- Random Forest(bagging), XGBoost classifier(boosting) are the two ensemble techniques applied over the dataset
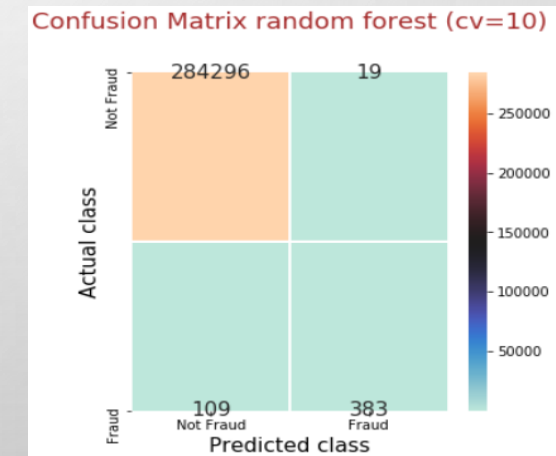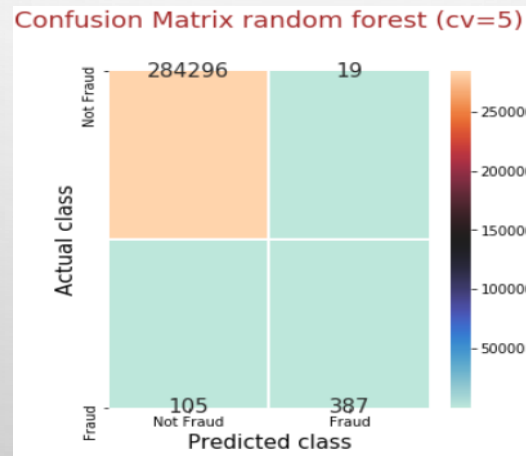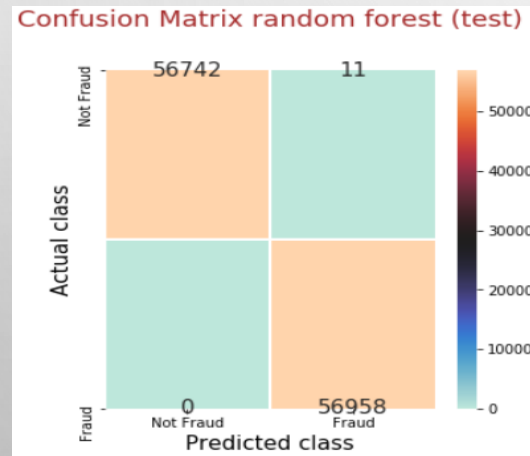
AUTO ML

It is a process of automating the complete machine learning process for real world scenarios.

Advantages:

- Improve efficiency by automatically running repetitive tasks. This allows data scientists to focus more on problems instead of models.

- Automated ml pipelines also help avoid potential errors caused by manual work.

- Automl is a big step toward the democratization of machine learning and allows everyone to use ml features.

contd…..

- Using h2o.ai, automl is implemented on the source data in this project

- H2O driverless AI is an artificial intelligence (AI) platform for automatic machine learning

- Driverless AI automates some of the most difficult data science and machine learning workflows such as feature engineering, model validation, model tuning, model selection, and model deployment

- It aims to achieve highest predictive accuracy

- Driverless AI runs on commodity hardware. It was also specifically designed to take advantage of graphical processing units (gpus), including multi-gpu workstations and servers such as IBM power9-gpu AC922 server and the NVIDIA DGX-1 for order-of-magnitude faster training.

H2O.AI (v1.9.0, 1 GPU)

By J Krishna Ravali

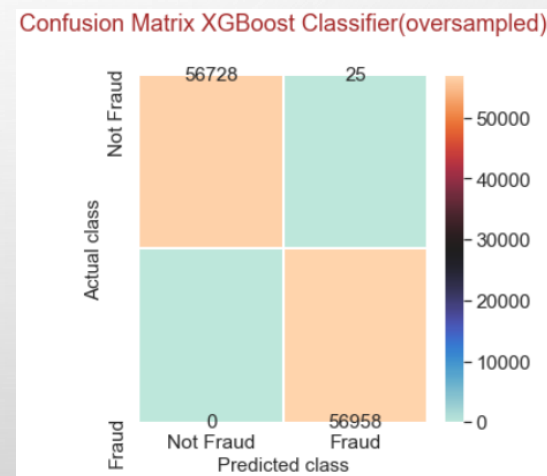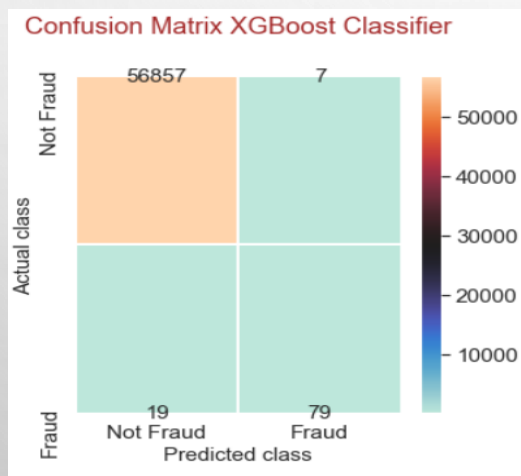03-12-2020

FINAL VIEW

XGB MODEL REPLICATION IN PYTHON

- The same model parameters are considered from the results of optimized model generated from h2o.ai which is replicated both on oversample and normal data set in python only considering the parameters suiting desktop version processing.

RESULTS

Metric/ Model	Supervised			Unsupervised		Ensembled		
	Logistic	Naïve Bayes	KNN	Mini Batch K Means	K Means	Random Forest	XGBoost	XGBoost (oversampled)
Accuracy	91.81	83.37	99.88	99.9	99.9	99.99	99.95	99.98
Precision	94.42	96.71	99.75	88.77	88.77	99.98	91.86	99.96
Sensitivity	88.9	70.18	100	51.42	51.42	100	80.61	100
Specificity	94.73	97.61	99.75	99.99	99.99	99.98	99.99	99.96
F1 score	91.58	81.34	99.88	65.12	65.12	99.99	85.87	99.98
Classification error	8.19	16.13	0.12	0.1	0.1	0.01	0.05	0.02

Metric/Model	LG (CV=5)	LG (CV=10)	RDF (CV=5)	RDF (CV=10)	NB (CV=5)	NB (CV=10)	KNN (CV=5)	KNN (CV=10)
Accuracy	97.65	97.68	99.96	99.96	99.92	99.91	99.95	99.95
Precision	6.3	6.4	95.32	95.27	82.55	82.51	93.38	93.19
Sensitivity	91.06	91.26	78.66	77.85	64.43	64.23	77.44	77.85
Specificity	97.66	97.69	99.99	99.99	99.98	99.98	99.99	99.99
F1 score	11.79	11.96	86.19	85.68	72.37	72.23	84.67	84.83
Classification error	2.35	2.32	0.04	0.04	0.08	0.09	0.05	0.05

KEY OBSERVATIONS

- *In case of supervised models, cross validation results are almost consistent with test data results implying no overfitting in model.*

- *In case of random forest classification, model has given better results with oversampled data, but after cross validation results (stratified k-fold) with cv=5 and 10, there was a slight performance drop.*

- *XGBoost classifier model which is a replication of optimized model given by automl, has given optimum performance relative to other models.*

DEPLOYMENT

- In this project XGBoost model was deployed to cloud and local environments

- Pickle file is created for the model to be deployed

- To local using flask which is triggered from anaconda prompt

- Flask is an API of python that allows us to build up web-applications
  - http://127.0.0.1:5000/

- To cloud using Heroku through flask

- Heroku is a platform as a service (paas) that enables developers to build, run, and operate applications entirely in the cloud.
 - https://credit-fraud-detection.herokuapp.com/

contd...

LOCAL   **CLOUD**

FUTURE SCOPE

- Deployment process can be replicated for other models like random forest

- Implementing the deployment process in AWS

- Building real time integrated application using spark

- Hybrid model for fraud detection by integrating unsupervised and supervised machine learning models

LITERATURE SURVEY

- Fraud detection in credit card transaction using neural networks i. sadgali, n. sael, f. benabbou https://www.semanticscholar.org/paper/fraud-detection-in-credit-card-transaction-using-sadgali-sael/99c256642184077fefb3dd084086b2593e73a166

- An artificial intelligence approach to financial fraud detection under iot environment: a survey and implementation dahee choi1 and kyungho lee1 https://www.hindawi.com/journals/scn/2018/5483472/

- Analysis on credit card fraud detection methods s. benson edwin raj; a. annie portia

  - https://ieeexplore.ieee.org/abstract/document/5762457

- Integration of unsupervised and supervised machine learning algorithms for credit risk assessment wang baoa , ning lianjua,∗ , kong yue https://www.researchgate.net/publication/331872177_integration_of_unsupervised_and_supervised_machine_learning_algorithms_for_credit_risk_assessment

- Application of credit card fraud detection: based on bagging ensemble classifier masoumehzareapoor[a] pouryashamsolmoali

  - https://www.sciencedirect.com/science/article/pii/s1877050915007103

- towards digital staining using imaging mass spectrometry and random forests-technical report

  Michael hanselmann, ullrich köthe, marc kirchner, russell h morgan https://www.researchgate.net/figure/the-random-forest-classifier-is-an-ensemble-of-decision-trees-where-the-single-trees-are_fig1_228540194

REFERENCES

- https://github.com/

- https://www.kaggle.com/

- https://medium.com/

- https://machinelearningmastery.com/

- https://analyticsvidhya.com

- https://www.youtube.com/watch?v=mrExsjcvF4o

APPENDIX

- Code for xgb model deployment: https://github.Com/krishnarj422/credit-fraud-detection

- Full code: https://github.Com/krishnarj422/credit-card-fraud-detection

THANK YOU.

By J Krishna Ravali

03-12-2020

> *Using large models and datasets, we demonstrate Adam can efficiently solve practical deep learning problems.*
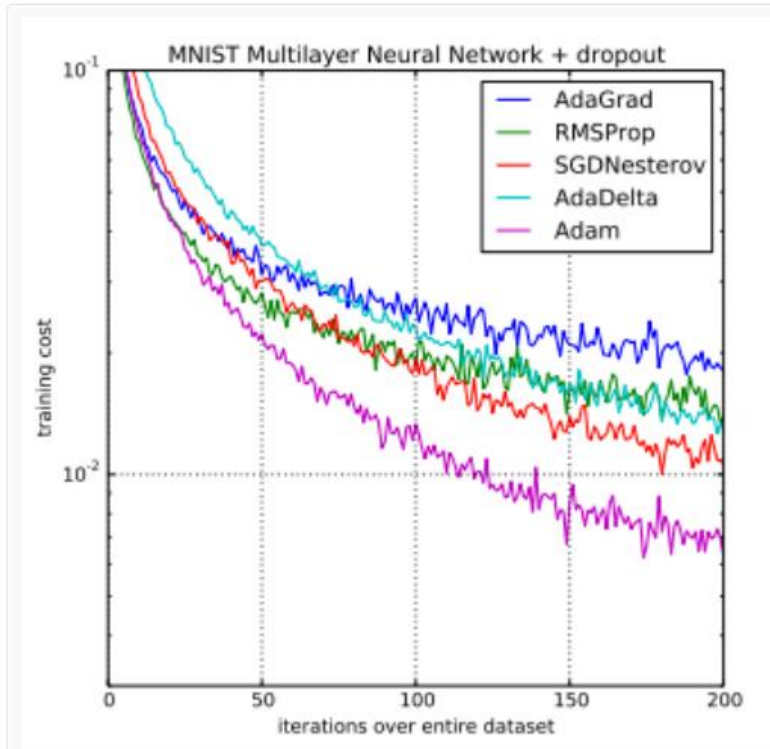
Comparison of Adam to Other Optimization Algorithms Training a
Multilayer Perceptron
Taken from Adam: A Method for Stochastic Optimization, 2015.

Sebastian Ruder developed a comprehensive review of modern gradient descent optimization algorithms titled "An overview of gradient descent optimization algorithms" published first as a blog post, then a technical report in 2016.

OTHER REFERENCES

- https://machinelearningmastery.com/cross-validation-for-imbalanced-classification/

- https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

- https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/#:~:text=adam%20is%20an%20optimization%20algorithm,iterative%20based%20in%20training%20data.&text=the%20algorithm%20is%20called%20adam,derived%20from%20adaptive%20moment%20estimation.

- https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/#:~:text=each%20node%20in%20the%20decision,to%20generate%20the%20final%20output.&text=the%20random%20forest%20algorithm%20combines,to%20generate%20the%20final%20output.

- https://analyticsindiamag.com/random-forest-vs-xgboost-comparing-tree-based-algorithms-with-codes/

- https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/

- https://www.alibabacloud.com/blog/6-top-automl-frameworks-for-machine-learning-applications-may-2019_595317

- https://realpython.com/python-pickle-module/