DATA LAB PROJECT

Pre set file directory run->edit configuration to Msc Project

General plan:

1) explore the dataset

2) exploratory data analysis (EDA)

3) modelling -> class prediction (random forest, neural networks)

4) synthesize data and then evaluate with modelling (train on original + synthetic and validate always on original) and statistics (compare distributions, etc. )

Kolmogorov Smirnoff method

Background reading:

* Synthpop
* GANs
* TGANs

To do : thesis + technical blog + presentation for the data lab

# Work log

## Week 1

Setting up the laptop, introductory meetings

Installing the software, background reading on tGANs, synthpop, GANs using Mendeley

Getting familiar with pyCharm

EDA as below

Getting familiar with keras and tensorflow, setting up keras and tensorflow

Neural network practice and on the dataset with keras and tensorflow

## Week 2

Precision-recall curves

Decision tree research – r or python

Construction a decision tree

Synthpop – generating and artificial dataset

Running random forest with synthetic data

Getting access to cloud services

Neural networks unexplainable

SVM very slow on large datasets

* Random forest

Should I optimize parameters after adding synthetic data?

Computer with GPU

Sticking with random forest, reasons

K-S stats test – graphs from synthpop

## Week 3

Clean up the code -> add functions to avoid repeating the code

Decision tree importance scores

Research and implement the methods for analyzing relationships within data and between different features

Try various synthpop data generation methods

Analyze data generated using different methods

Practice VAE

Try out GANs

Conclusions: based on data comparison, it’s clear that non fraud features are much closer to the original data than the fraud and therefore I am trying to synthesize just the fraud data

## Week 4

Consult Jianguo about evaluating synthetic datasets

Make a simple model generating data

Explore mutual information

<https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html>

<https://github.com/scikit-learn/scikit-learn/blob/412996f/sklearn/feature_selection/mutual_info_.py#L365>

run vGAN, WGAN, compare to synthpop, cGAN with k-means

## Week 5

Innovation week

cGANs, WcGANs

Literature review on measuring the quality of synthetic data (Thanos papers)

## Week 6

Synthetic ranking agreement

pMSE

TSTR

Questions for Thanos:

* How do you train the classifier in propensity score
* competition measure requires synthesizing original data and computationally demanding
* cross-validation and re-training
* very small accuracy differences
* are the quality measure corresponding to improved performance?

More evaluation methods

## Week 7

Write up methods

Improve GANs

Evaluate data – a copy or similar dataset?

Revising maths to understand GANs better

Exploring AWS services – learning how to deploy my code in the aws services

Meeting with Thanos:

Discussing cloud services, best approaches

parameters as inputs code

latex, overleaf

cambridge university template

Andrew Ng – courser

## Week 8 (after the 3 week break)

Connect to AWS instance – remote desktop

Finalise the code – improve the functions, eg. Parameters as function inputs

Set up GitHub synchronization

Write up - relevance

Optimize assessment of GANs

<https://towardsdatascience.com/understanding-and-optimizing-gans-going-back-to-first-principles-e5df8835ae18>

## Week 9

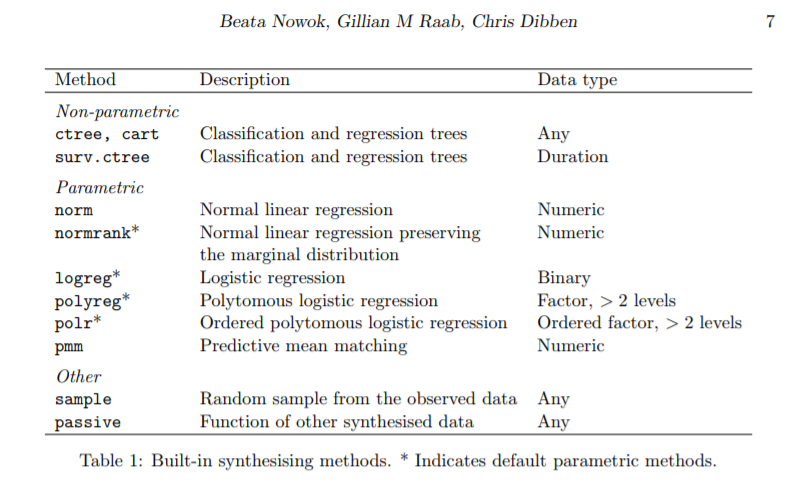
Enhance code

Optimize WcGAN parameters and architecture

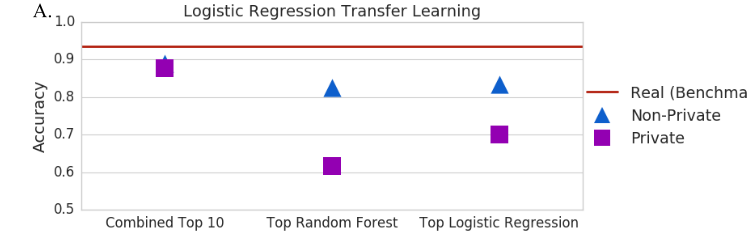
Getting final figures – recall improvement

Develop pipeline for assessing the performance improvement

Compare all the main data generators using mesures



Tried ctree and normrank



# Codes

Forest\_model\_bal

Forest\_model\_unbal

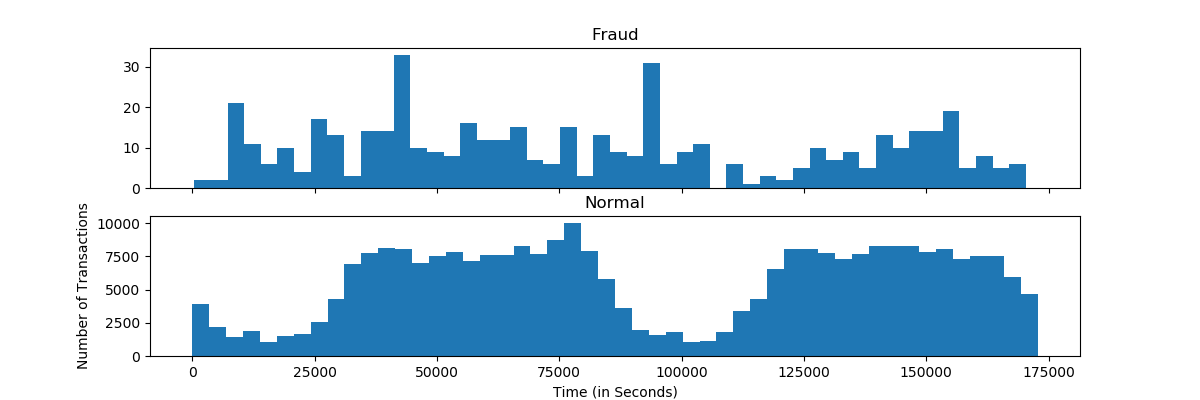
So – synthetic only

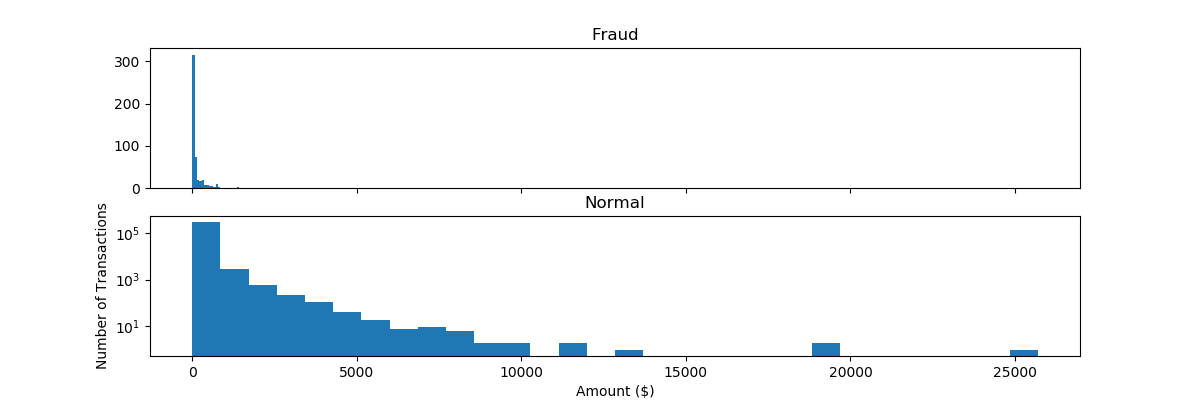
Mix

ori

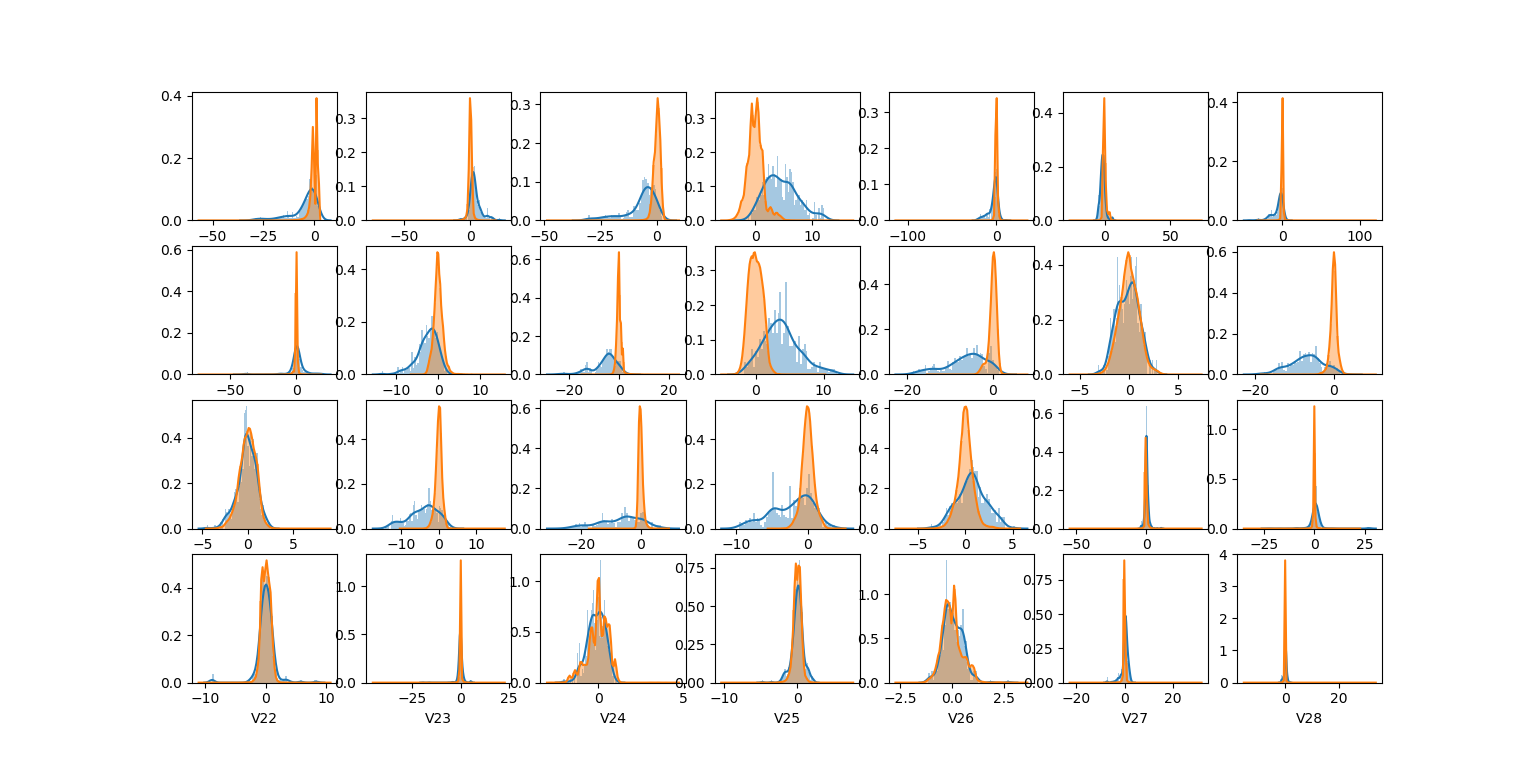
# EDA

python -m pip install pandas





For the features - compare the histograms to decide which ones give a good separation of fraud and not fraud



Maybe remove the features which do not give a good separation?

T-sne

There are several ways to approach this classification problem taking into consideration this unbalance.

* Collect more data? Nice strategy but not applicable in this case
* Changing the performance metric:
  + Use the confusio nmatrix to calculate Precision, Recall
  + F1score (weighted average of precision recall)
  + Use Kappa - which is a classification accuracy normalized by the imbalance of the classes in the data
  + ROC curves - calculates sensitivity/specificity ratio.
* Resampling the dataset
  + Essentially this is a method that will process the data to have an approximate 50-50 ratio.
  + One way to achieve this is by OVER-sampling, which is adding copies of the under-represented class (better when you have little data)
  + Another is UNDER-sampling, which deletes instances from the over-represented class (better when he have lot's of data)

Precision-recall curve

Lung x-ray pictures dataset: <https://www.kaggle.com/nih-chest-xrays/sample>

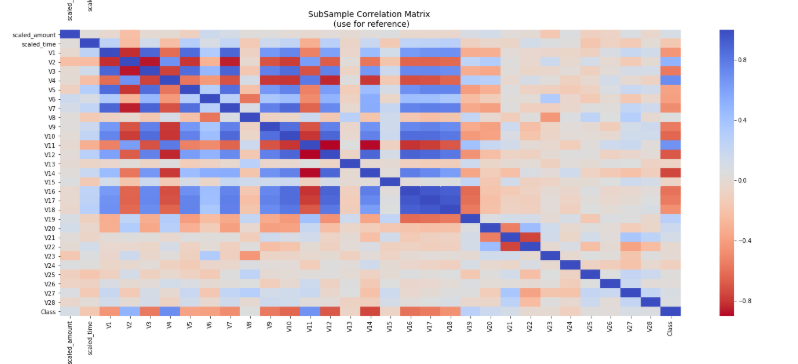
<https://www.kaggle.com/uciml/pima-indians-diabetes-database>

Correlation matrix for fraud:

<https://www.kaggle.com/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets>

<http://di.ulb.ac.be/map/adalpozz/pdf/Dalpozzolo2015PhD.pdf>

The main issue with "Random Under-Sampling" is that we run the risk that our classification models will not perform as accurate as we would like to since there is a great deal of **information loss** (bringing 492 non-fraud transaction from 284,315 non-fraud transaction)



# Datasets

<https://archive.ics.uci.edu/ml/datasets/adult>

# Neural networks in python

<https://www.youtube.com/watch?v=83vR1Nz3dHA>

No Frauds 99.83 % of the dataset

Frauds 0.17 % of the dataset

First attempt at the NN while training with the whole dataset resulted in the confusion matrix like this:

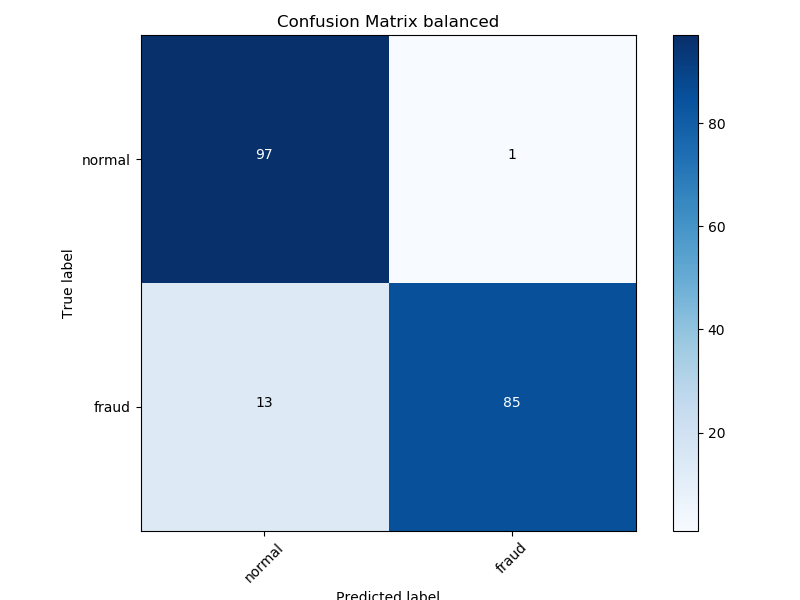
[ 3 56860]

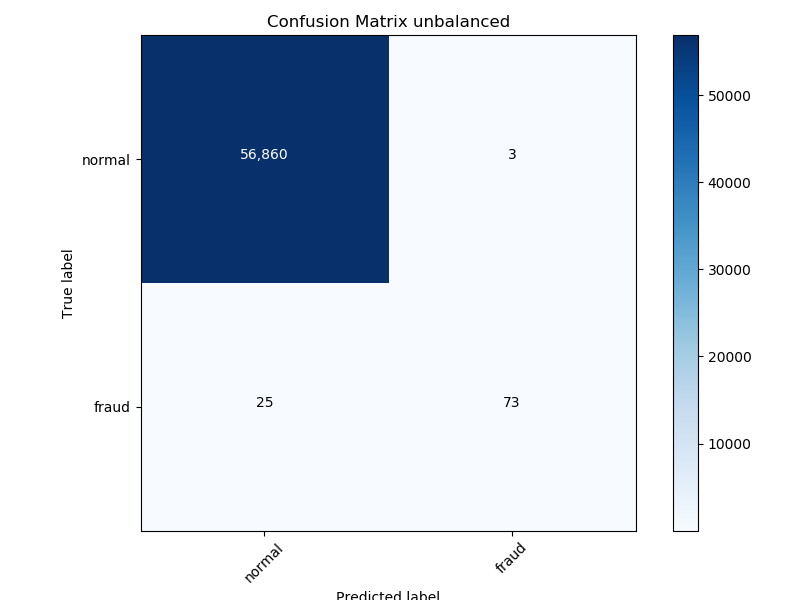
[ 73 25]

And on balanced dataset:

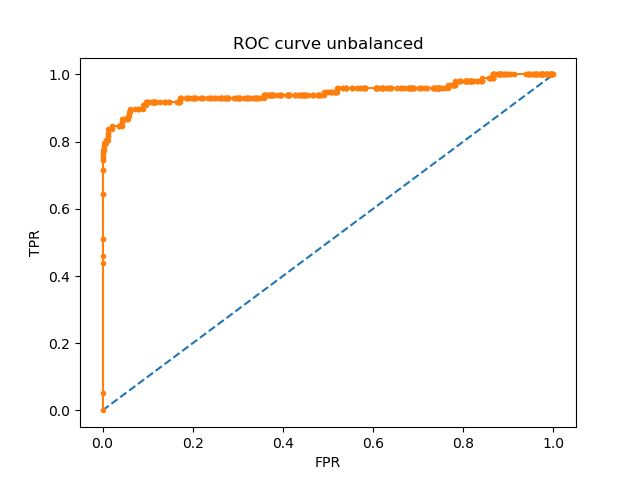
[ 1 97]

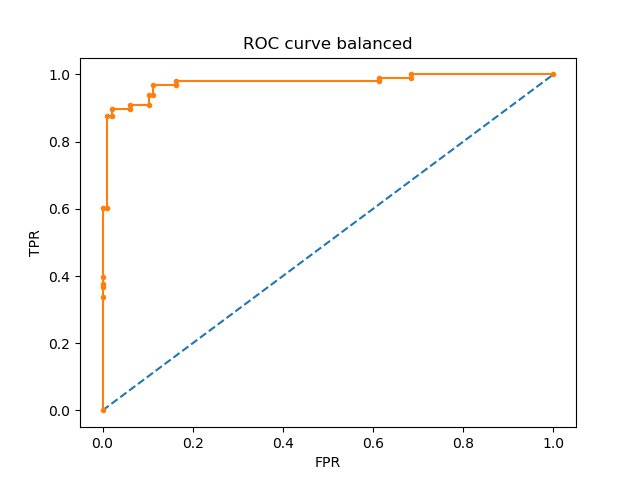
[85 13]

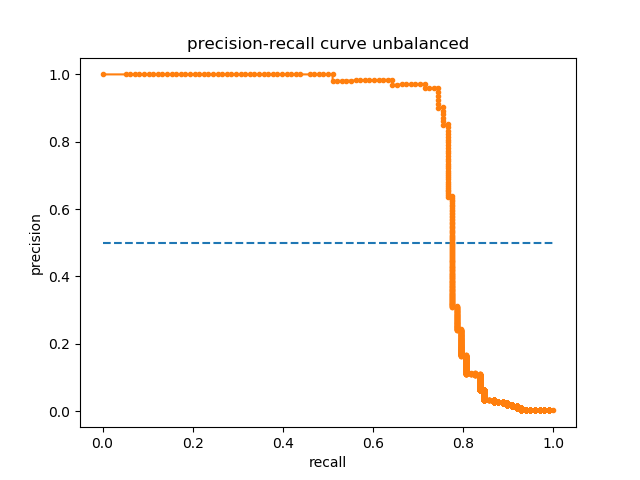


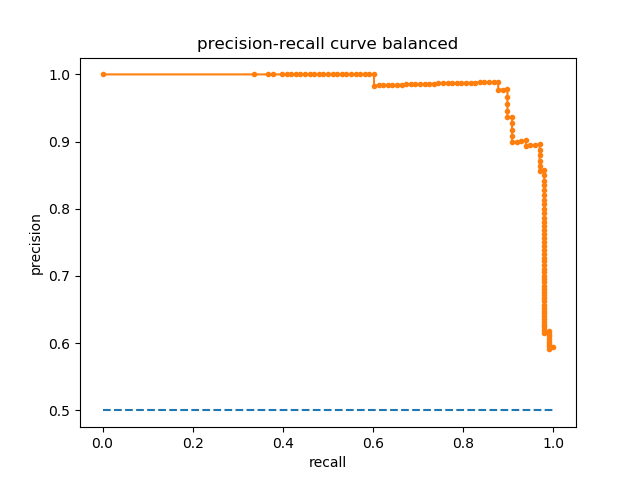


Synthpop workshop: <https://www.geos.ed.ac.uk/homes/graab/notes.pdf>









Precision is a ratio of the number of true positives divided by the sum of the true positives and false positives. It describes how good a model is at predicting the positive class. Precision is referred to as the positive predictive value.

Precision = True Positives / (True Positives + False Positives)

Recall = True Positives / (True Positives + False Negatives)

# Decision trees

Sklearn labelencoder

Problems with visualising the tree in python, R seems ato be a better tool

<https://www.datacamp.com/community/tutorials/decision-trees-R>

ROC curve for decision trees??

balanced model: precision recall f1-score support

0 0.94 0.92 0.93 97

1 0.92 0.94 0.93 100

micro avg 0.93 0.93 0.93 197

macro avg 0.93 0.93 0.93 197

weighted avg 0.93 0.93 0.93 197

unbalanced model: precision recall f1-score support

0 1.00 1.00 1.00 56875

1 0.73 0.71 0.72 87

micro avg 1.00 1.00 1.00 56962

macro avg 0.86 0.86 0.86 56962

weighted avg 1.00 1.00 1.00 56962

[[89 8]

[ 6 94]]

[[56852 23]

[ 25 62]]

After grid search

[[96 1]

[ 9 91]]

[[56867 8]

[ 23 64]]

balanced model: precision recall f1-score support

0 0.91 0.99 0.95 97

1 0.99 0.91 0.95 100

micro avg 0.95 0.95 0.95 197

macro avg 0.95 0.95 0.95 197

weighted avg 0.95 0.95 0.95 197

unbalanced model: precision recall f1-score support

0 1.00 1.00 1.00 56875

1 0.89 0.74 0.81 87

micro avg 1.00 1.00 1.00 56962

macro avg 0.94 0.87 0.90 56962

weighted avg 1.00 1.00 1.00 56962

<https://stackoverflow.com/questions/35097003/cross-validation-decision-trees-in-sklearn>

one hot encoding: [https://scikit-learn.org/stable/modules/preprocessing.html#encoding-categorical-features](https://scikit-learn.org/stable/modules/preprocessing.html)

# SVM

Grid search

Parameters

[  
 {**'C'**: [1, 10, 100, 1000], **'kernel'**: [**'linear'**]},  
 {**'C'**: [1, 10, 100, 1000], **'gamma'**: [0.001, 0.0001], **'kernel'**: [**'rbf'**]},  
 ]

Scoring parameters <https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter>

# Synthpop

> syn\_lin\_reg$method

V1 V2 V3 V4 V5 V6 V7 V8 V9

"sample" "normrank" "normrank" "normrank" "normrank" "normrank" "normrank" "normrank" "normrank"

V10 V11 V12 V13 V14 V15 V16 V17 V18

"normrank" "normrank" "normrank" "normrank" "normrank" "normrank" "normrank" "normrank" "normrank"

V19 V20 V21 V22 V23 V24 V25 V26 V27

"normrank" "normrank" "normrank" "normrank" "normrank" "normrank" "normrank" "normrank" "normrank"

V28 scaled\_amount scaled\_time class

"normrank" "normrank" "normrank" "constant"

Method "cart" is not valid for a variable without predictors (V1)

Method has been changed to "sample"

> syn\_cart$method

V1 V2 V3 V4 V5 V6 V7 V8 V9

"sample" "cart" "cart" "cart" "cart" "cart" "cart" "cart" "cart"

V10 V11 V12 V13 V14 V15 V16 V17 V18

"cart" "cart" "cart" "cart" "cart" "cart" "cart" "cart" "cart"

V19 V20 V21 V22 V23 V24 V25 V26 V27

"cart" "cart" "cart" "cart" "cart" "cart" "cart" "cart" "cart"

V28 scaled\_amount scaled\_time class

"cart" "cart" "cart" "constant"

> syn\_default$method

V1 V2 V3 V4 V5 V6 V7 V8 V9

"sample" "cart" "cart" "cart" "cart" "cart" "cart" "cart" "cart"

V10 V11 V12 V13 V14 V15 V16 V17 V18

"cart" "cart" "cart" "cart" "cart" "cart" "cart" "cart" "cart"

V19 V20 V21 V22 V23 V24 V25 V26 V27

"cart" "cart" "cart" "cart" "cart" "cart" "cart" "cart" "cart"

V28 scaled\_amount scaled\_time class

"cart" "cart" "cart" "cart"

Data team meeting:

Amazon synthetic data – healthcare

Fraud detection - mobile market, labara

One sample Kolmogorov-Smirnov test – comparing each line to a normal distribution

Kolmogorov-Smirnov two sample test to check if the two distributions differ

Chi-square test for association and Jaccard Similarity Index

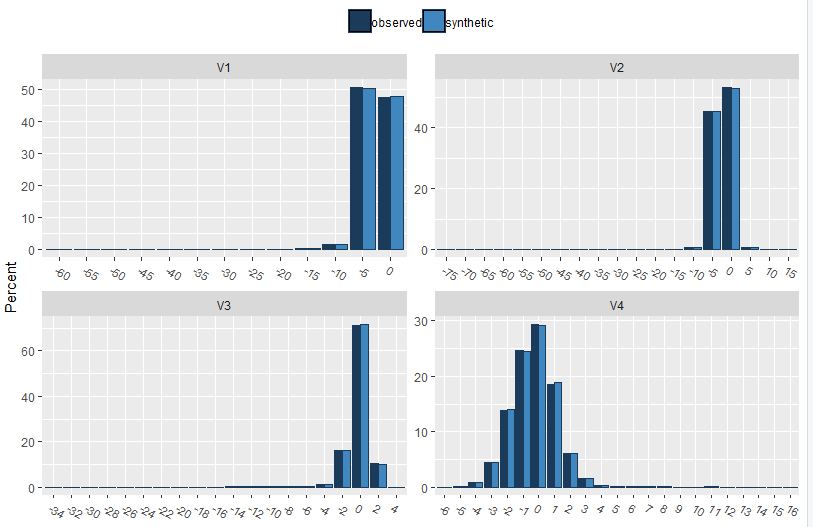
Kolmogorov-smirnov for continuous data

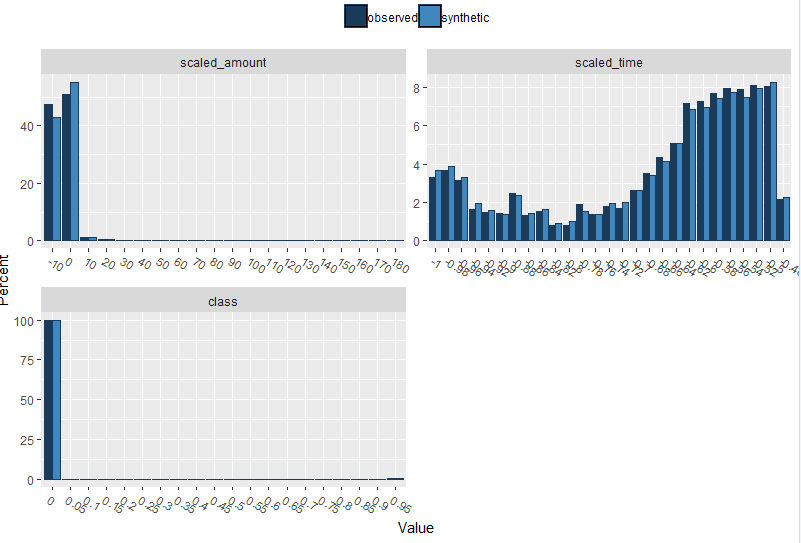
Mann-Whitney test

Chi-square

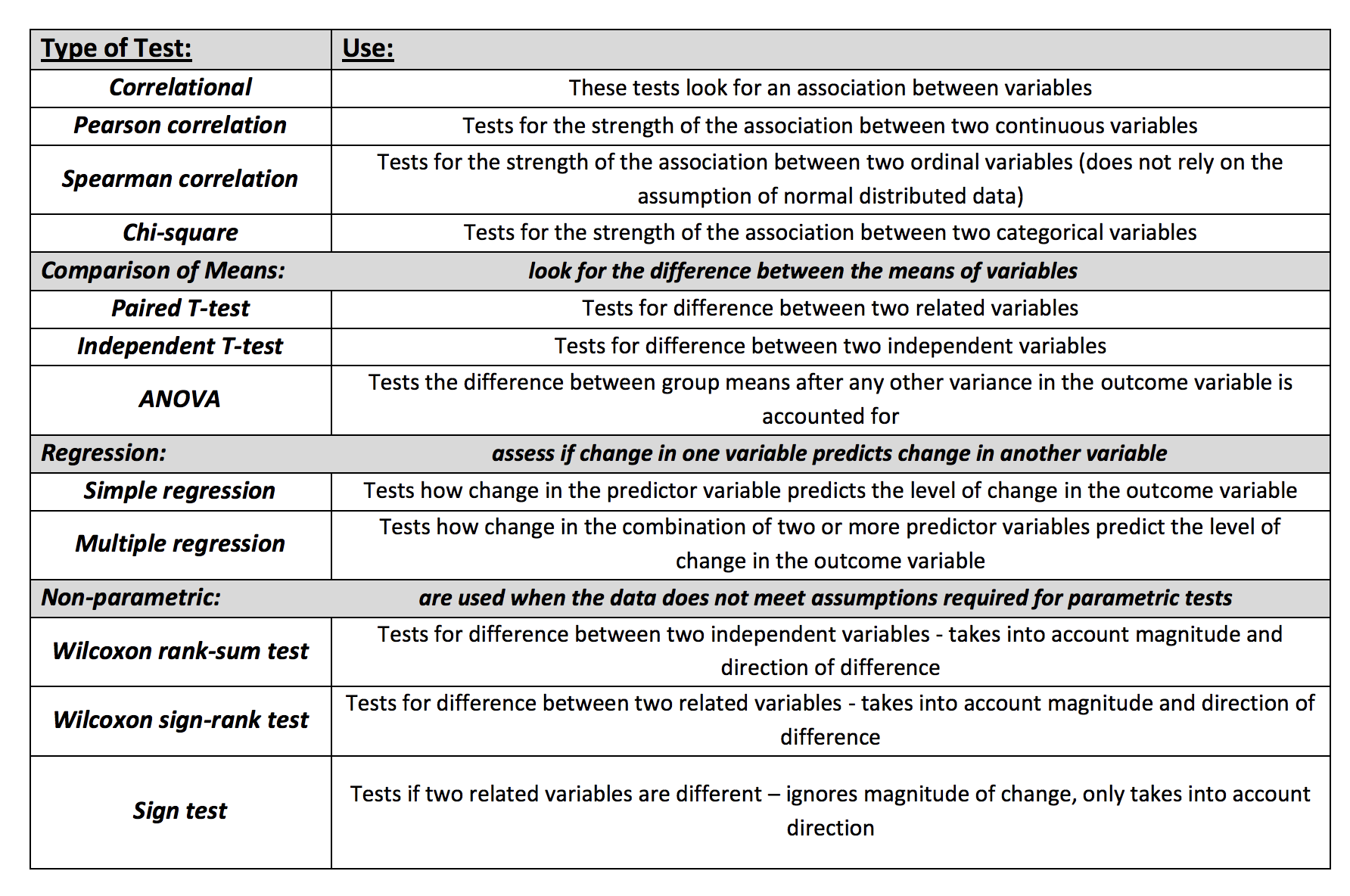
<https://en.wikipedia.org/wiki/Statistical_distance>

If instead you mean "I calculate my test statistic on a sample drawn from a discrete distribution and then use a suitable critical value/calculate a suitable p-value for my situation" (say via a permutation test, for example), then the test is certainly valid in the sense that you'll get the right type I error rate -- up to the discreteness of the test statistic itself, of course. (Though there may well be better tests for your particular purpose, just as there usually are in the continuous case.)





<https://stats.stackexchange.com/questions/278527/testing-whether-two-data-sets-are-statistically-different>



<https://www.graphpad.com/support/faqid/1790/>

<https://influentialpoints.com/Training/kolmogorov-smirnov_test-principles-properties-assumptions.htm>

# Random Forest

<https://towardsdatascience.com/random-forest-in-python-24d0893d51c0>

=============================================

unbalanced\_model on unbalanced test

---------------------------------------------

confusion matrix

[[56869 6]

[ 21 66]]

---------------------------------------------

precision recall f1-score support

0 1.00 1.00 1.00 56875

1 0.92 0.76 0.83 87

micro avg 1.00 1.00 1.00 56962

macro avg 0.96 0.88 0.91 56962

weighted avg 1.00 1.00 1.00 56962

=============================================

=============================================

unbalanced\_model\_syn on unbalanced test

---------------------------------------------

confusion matrix

[[56864 11]

[ 19 68]]

---------------------------------------------

precision recall f1-score support

0 1.00 1.00 1.00 56875

1 0.86 0.78 0.82 87

micro avg 1.00 1.00 1.00 56962

macro avg 0.93 0.89 0.91 56962

weighted avg 1.00 1.00 1.00 56962

=============================================

=============================================

balanced\_model on balanced test

---------------------------------------------

confusion matrix

[[84 3]

[ 9 78]]

---------------------------------------------

precision recall f1-score support

0 0.90 0.97 0.93 87

1 0.96 0.90 0.93 87

micro avg 0.93 0.93 0.93 174

macro avg 0.93 0.93 0.93 174

weighted avg 0.93 0.93 0.93 174

=============================================

=============================================

balanced\_model\_syn on balanced test

---------------------------------------------

confusion matrix

[[78 9]

[ 4 83]]

---------------------------------------------

precision recall f1-score support

0 0.95 0.90 0.92 87

1 0.90 0.95 0.93 87

micro avg 0.93 0.93 0.93 174

macro avg 0.93 0.93 0.93 174

weighted avg 0.93 0.93 0.93 174

=============================================

=============================================

balanced\_model on unbalanced test

---------------------------------------------

confusion matrix

[[55273 1602]

[ 9 78]]

---------------------------------------------

precision recall f1-score support

0 1.00 0.97 0.99 56875

1 0.05 0.90 0.09 87

micro avg 0.97 0.97 0.97 56962

macro avg 0.52 0.93 0.54 56962

weighted avg 1.00 0.97 0.98 56962

=============================================

=============================================

balanced\_model\_syn on unbalanced test

---------------------------------------------

confusion matrix

[[54001 2874]

[ 4 83]]

---------------------------------------------

precision recall f1-score support

0 1.00 0.95 0.97 56875

1 0.03 0.95 0.05 87

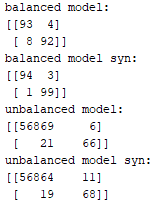
micro avg 0.95 0.95 0.95 56962

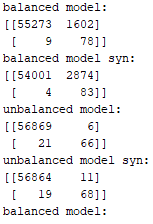
macro avg 0.51 0.95 0.51 56962

weighted avg 1.00 0.95 0.97 56962

=============================================

Results from random forest





# Relationship within data testing

<https://datasciencecampus.ons.gov.uk/projects/synthetic-data-for-public-good/> - explanation of pearson correlation

privacy-preserving paper – nice correlation

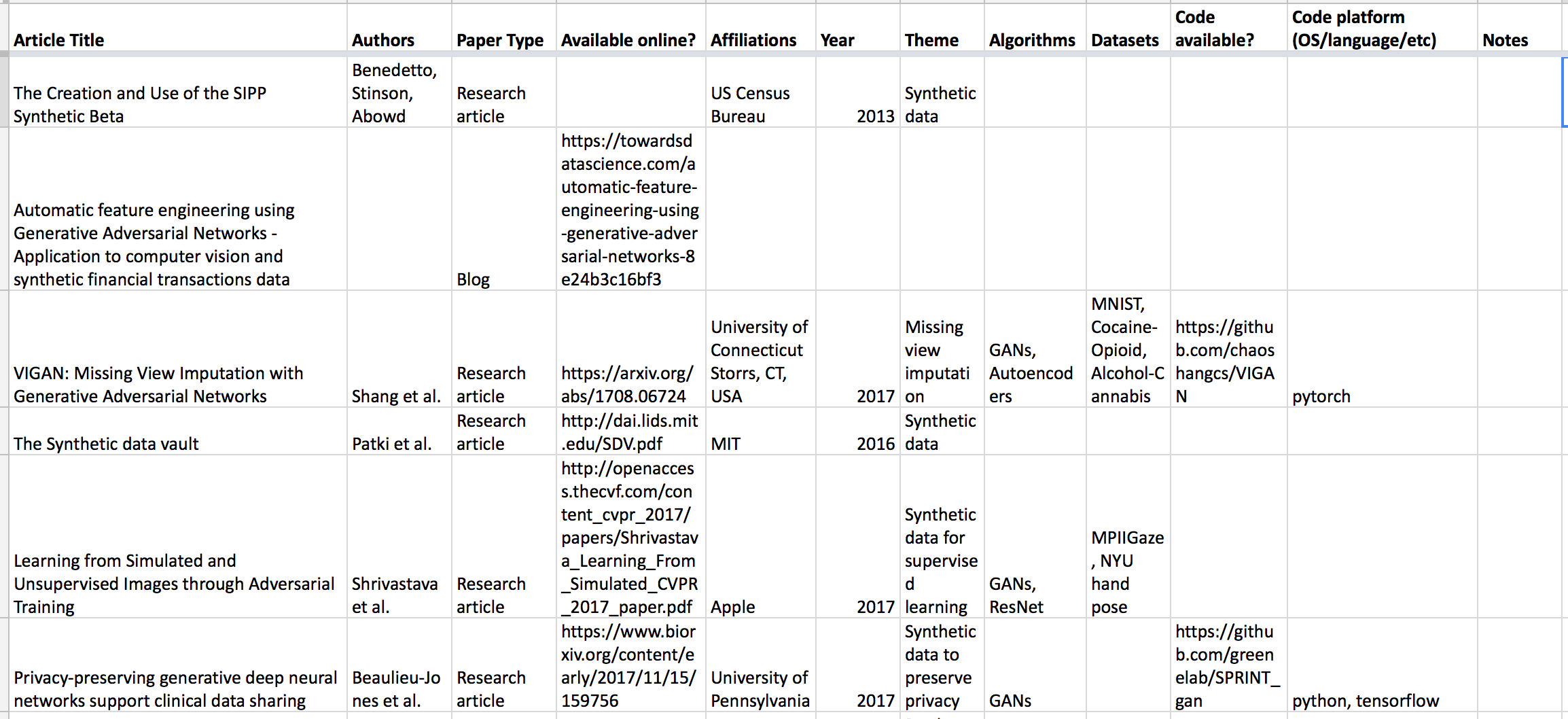
<https://datasciencecampus.ons.gov.uk/projects/generative-adversarial-networks-gans-for-synthetic-dataset-generation-with-binary-classes/>

performance of algorithms trained only on synthetic data

use random forest feature importance scores

source code for the above:

<https://github.com/datasciencecampus/Synthetic_data>

literature review

Repeated measures correlation: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5383908/>

**Pearson product moment correlation**

The Pearson correlation evaluates the linear relationship between two continuous variables. A relationship is linear when a change in one variable is associated with a proportional change in the other variable.

For example, you might use a Pearson correlation to evaluate whether increases in temperature at your production facility are associated with decreasing thickness of your chocolate coating.

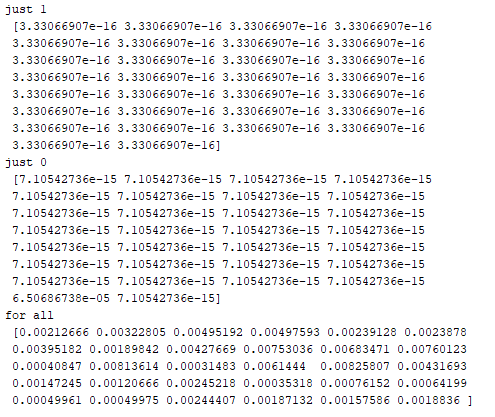
**Spearman rank-order correlation**

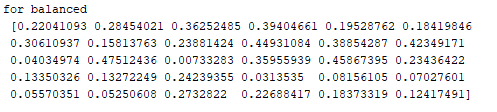
The Spearman correlation evaluates the monotonic relationship between two continuous or ordinal variables. In a monotonic relationship, the variables tend to change together, but not necessarily at a constant rate. The Spearman correlation coefficient is based on the ranked values for each variable rather than the raw data.

Spearman correlation is often used to evaluate relationships involving ordinal variables. For example, you might use a Spearman correlation to evaluate whether the order in which employees complete a test exercise is related to the number of months they have been employed.

# Mutual information

sklearn.feature\_selection.mutual\_info\_classif(X, y)





Mutual information is a measure between two (possibly multi-dimensional) random variables X and Y, that quantifies the amount of information obtained about one random variable, through the other random variable.

# GANs

GitHub repository of GAN implementations in Keras, Keras-GAN

Unlike regular ReLU function which maps any negative input to 0, Leaky ReLU allows a small positive gradient.

Tanh produces crispier images

ADAM a go to optimizer

Evaluation:

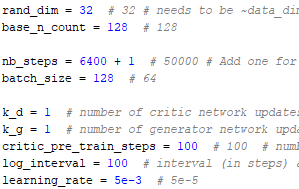
Inception score

Frechet Inception Score

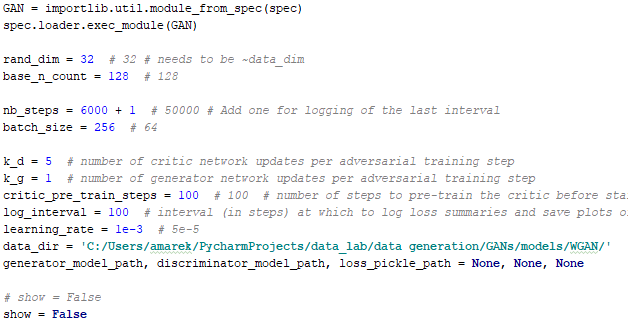
KL divergence and jennsen-shannon divergence

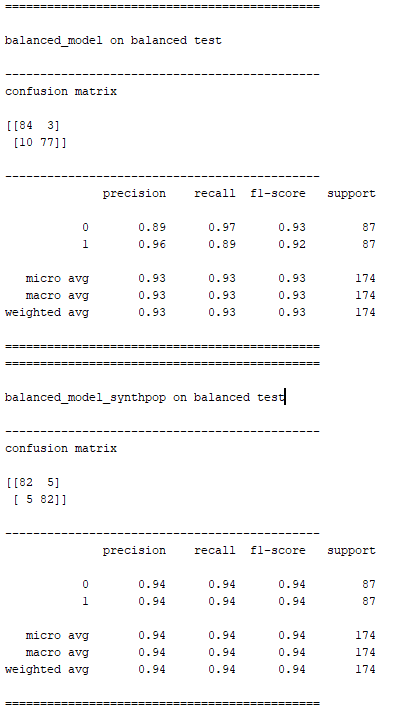
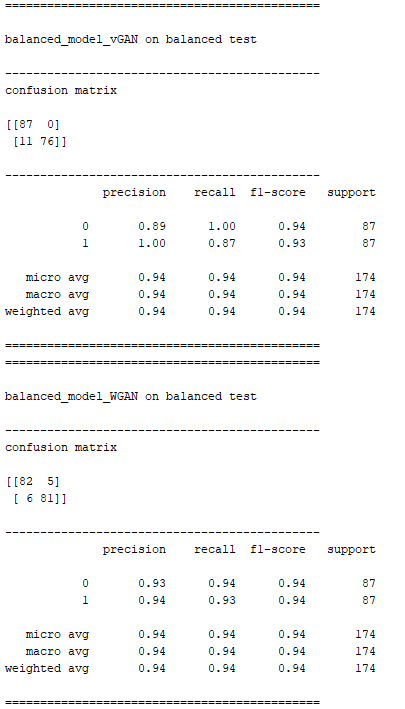
<https://www.toptal.com/machine-learning/generative-adversarial-networks>

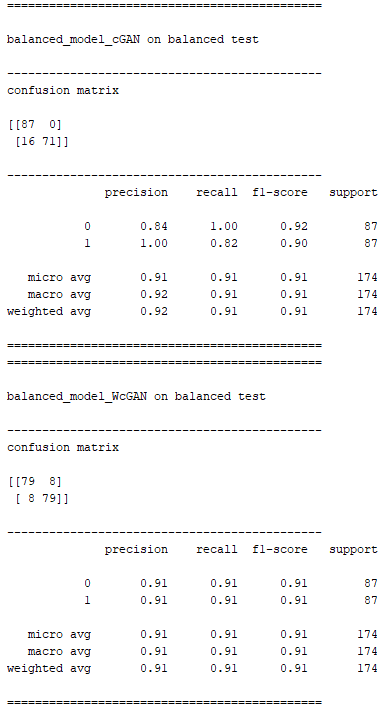
vGAN6000 best

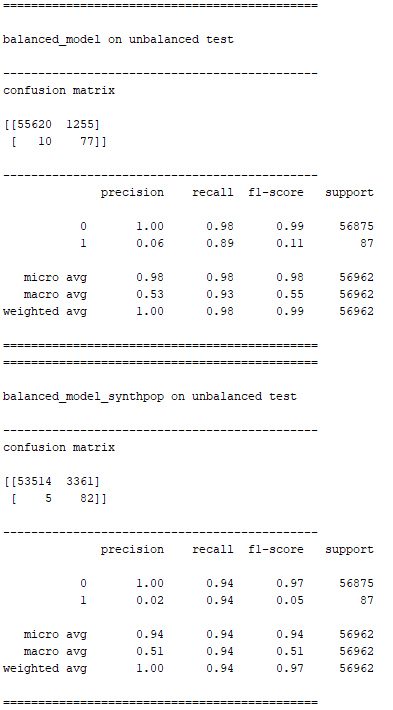
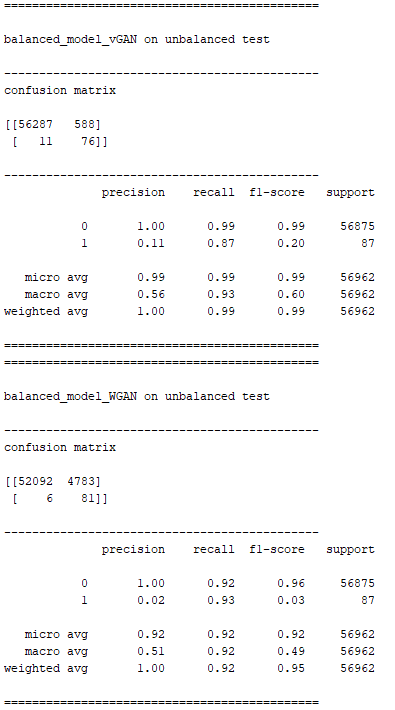


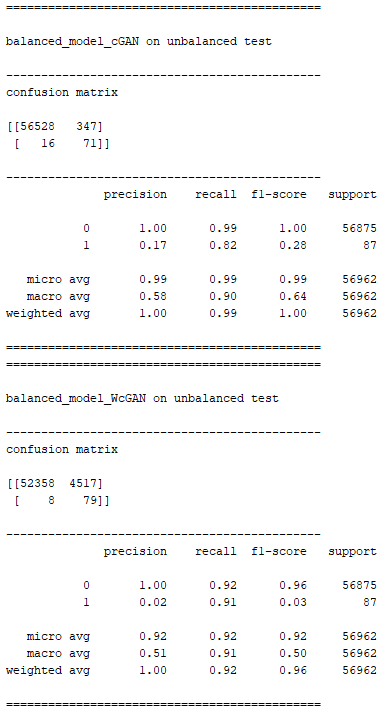
WGAN best







# Measuring the quality of synthetic data

## pMSE

<https://weina.me/nested-cross-validation/>

<https://scikit-learn.org/stable/tutorial/statistical_inference/model_selection.html>

<https://stackoverflow.com/questions/55808504/how-to-speed-up-nested-cross-validation-in-python>

## Competition measure

Cross-validation Accuracy GBM: 94.814815 (+/- 3.064117 )

Test set accuracy: 94.83 %

Cross-validation Accuracy RFC: 94.938272 (+/- 2.116843 )

Test set accuracy: 92.53 %

Cross-validation Accuracy Keras: 94.197531 (+/- 2.087844 )

Test set accuracy: 93.68 %

Cross-validation Accuracy SVM: 93.086420 (+/- 2.322826 )

Test set accuracy: 91.38 %

Cross-validation Accuracy GBM: 51.604938 (+/- 3.461196 )

Test set accuracy: 51.15 %

Cross-validation Accuracy RFC: 50.370370 (+/- 2.425541 )

Test set accuracy: 51.72 %

Cross-validation Accuracy Keras: 51.111111 (+/- 2.080531 )

Test set accuracy: 49.43 %

Cross-validation Accuracy SVM: 51.481481 (+/- 2.013519 )

Test set accuracy: 55.17 %

SRA: 0.16666666666666666

SRA: 0.8333333333333333

Canc: <https://medium.com/jungle-book/towards-data-set-augmentation-with-gans-9dd64e9628e6>

# AWS

EC2

<https://www.youtube.com/watch?v=M2Wc8JIS-p8>

<https://stackoverflow.com/questions/6030115/how-to-run-a-code-in-an-amazones-ec2-instance>

Virtual machine: <https://aws.amazon.com/getting-started/tutorials/launch-windows-vm/?trk=gs_card>

Remote desktop: <https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/connecting_to_windows_instance.html#rdp-prereqs>

blog set up for machine learning: <https://towardsdatascience.com/boost-your-machine-learning-with-amazon-ec2-keras-and-gpu-acceleration-a43aed049a50>

ec2 beginner tutorial: <https://www.datacamp.com/community/tutorials/aws-ec2-beginner-tutorial>

how to enable python, etc: <https://www.google.com/search?q=use+software+in+windows+instance+ec2&rlz=1C1GCEV_enGB848GB848&oq=use+software+in+windows+instance+ec2&aqs=chrome..69i57j33.8128j0j4&sourceid=chrome&ie=UTF-8#kpvalbx=1>

create image to save provisioning

how to create ec2 instance with python code: <https://blog.ipswitch.com/how-to-create-an-ec2-instance-with-python>

Boto3

<https://aws.amazon.com/pycharm/>

Google cloud with keras: <http://liufuyang.github.io/>

# tGAN

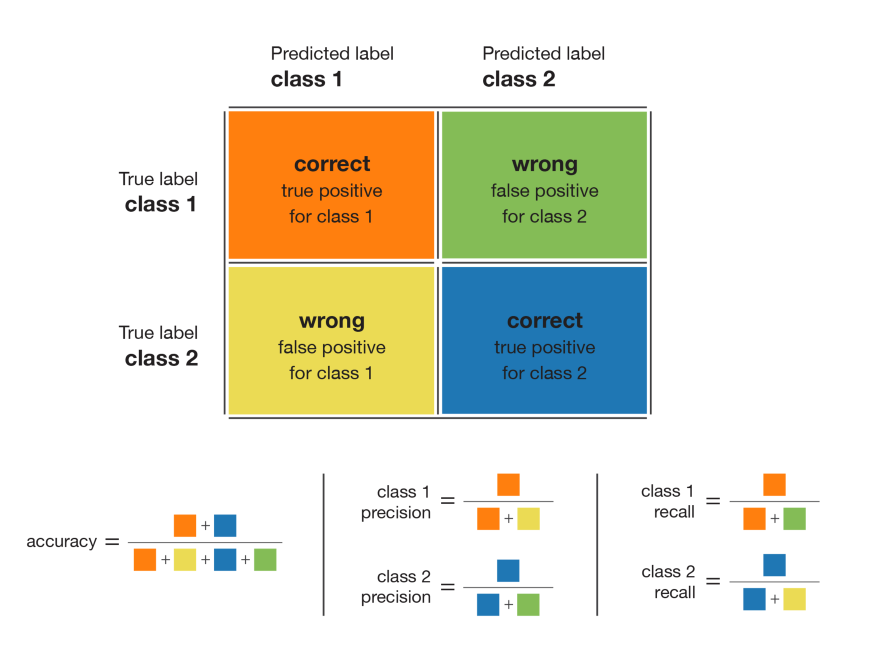
<https://github.com/DAI-Lab/TGAN>

# Explanation of recall, precision, etc.

<https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28>

 The accuracy of the model is basically the total number of correct predictions divided by total number of predictions. The precision of a class define how trustable is the result when the model answer that a point belongs to that class. The recall of a class expresses how well the model is able to detect that class. The F1 score of a class is given by the harmonic mean of precision and recall (2×precision×recall / (precision + recall)), it combines precision and recall of a class in one metric.

When using a resampling method (for example to get as much data from C0 than from C1), **we show the wrong proportions of the two classes to the classifier during the training**. The classifier learned this way will then have a lower accuracy on the future real test data than the classifier trained on the unchanged dataset. Indeed, the true proportions of classes are important to know for classifying a new point and that information has been lost when resampling the dataset.



* high recall + high precision : the class is perfectly handled by the model
* low recall + high precision : the model can’t detect the class well but is highly trustable when it does
* high recall + low precision : the class is well detected but the model also include points of other classes in it
* low recall + low precision : the class is poorly handled by the model

dataset: <https://link.springer.com/article/10.1007/s00521-017-3242-y>