Feature Selection / Dimensionality Reduction

Dr Paul Yoo

Dept CSIS

17/10/19

1

Timetable

Week	Date	Lecture (G12, Torrington, UCL)	Lab (MAL 414-417)
1	03/10/19	Introduction, Workflow and Loading	Loading data and descriptive statistics
2	10/10/19	Data pre-processing	Preparing data
3	17/10/19	Feature selection and re-sampling	Selecting features and re-sampling
4	24/10/19	DT and RF	Comparing ML algorithms
5	31/10/19	LR and NN	Automating the process
6	07/11/19	TensorFlow and Keras	MLP with Keras
7	14/11/19	Project Briefing	Project (30%)
8	21/11/19		
9	28/11/19	Image processing	Deep learning - CNN
10	05/12/19	RNN and sequential data	Deep learning - RNN
11	12/12/19	Real-life case	Deep learning - LSTM

Autumn term: 30/09/2019 to 13/12/2019

2

Overview

We covered:
- 5G and ML
- Predictive Modelling - The Analytic Workflow
- Data for ML
- Python
- Prediction types
- Data pre-processing
  - Re-scale, Normalise, Binarise, Standardise
  - Concept hierarchy
  - Noisy data – data smoothing (e.g. binning)

We will cover:
- Feature selection techniques
- Re-sampling

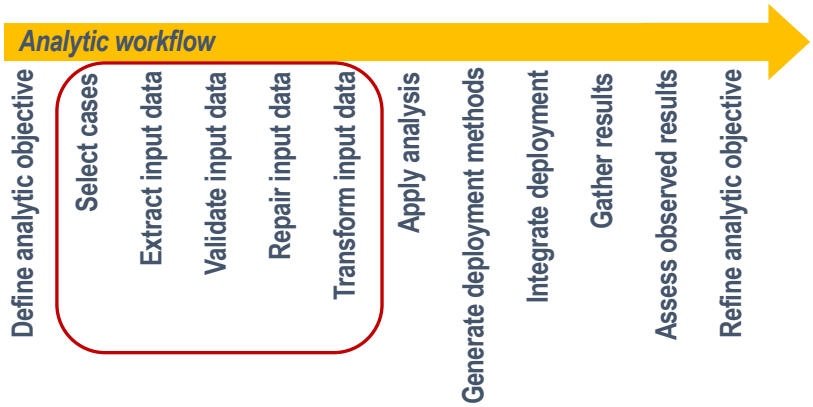
3

Predictive Modelling ML Steps

1. **Define Problem:** Investigate and characterise the problem in order to better understand the goals of the project.
2. **Analyse Data:** Use descriptive statistics and visualisation to better understand the data you have available.
3. **Prepare Data:** Use data transforms in order to better expose the structure of the prediction problem to modeling algorithms.
4. **Evaluate Algorithms:** Design a test harness to evaluate a number of standard algorithms on the data and select the top few to investigate further.
5. **Improve Results:** Use algorithm tuning and ensemble methods to get the most out of well-performing algorithms on your data.
6. **Present Results:** Finalise the model, make predictions and present results.

4

The Analytic Workflow



Source: SAS.com

Main Principles

The feature selection methods are typically presented in three classes based on how they combine the selection algorithm and the model building.

- **Filter (e.g. chi2)** - pick up the intrinsic properties of the features (i.e., the "relevance" of the features) measured via univariate statistics.
- **Wrapper (e.g. RFE)** - measures the "usefulness" of features based on the classifier performance (computationally more expensive due to repeated learning steps)
- **Embedded (e.g. DT)** – similar to wrapper but an intrinsic model building metric is used during learning.

FS Techniques

1. Percent missing values
2. Amount of variation
3. Pairwise correlation
4. Pincipal Component Analysis
5. Cluster anlaysis
6. Correlation (with the target)
7. Forward selection
8. Backward elimination
9. Stepwise selection
10. Embedded (e.g. DT)

Information

Redundancy

Predictive Power

Greedy Selection

Embedded

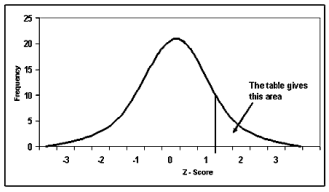
Percent Missing Values

- Drop variables that have a very high % of missing values
  - # of records with missing values / # of total records
- Create binary indicators (encode) to denote missing (or non-missing) values
- Review or visualise variables with high % of missing values. Why?

Amount of Variation

Drop or review variables that have a very low variation

- $VAR(x) = \sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$
- Either standardise all variables, or use standard deviation σ to account for variables with different scales
- Z-scores ($\boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{\mu}) / \boldsymbol{\sigma}$) are a way to compare results from a test to a "normal" population.
- Drop variables with zero variation (unary)



Pairwise Correlations

Many variables are often correlated with each other, and hence are redundant.

- If two variables are highly correlated, keeping only one will help reduce dimensionality without much loss of information.
- Which variable to keep?
  - The one that has a higher correlation coefficient with the target.

X^2 Correlation Test

Null hypothesis: the distribution of data is due to chance (independent).

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

- The larger the X^2 value, the more likely the variables are related.
- The cells that contribute the most to the X^2 value are those whose actual count is very different from the expected count.
- Correlation does not imply causality
  - Causality means the second event is understood as a consequence of the first.
  - # of hospitals and # of car-theft in a city are correlated
  - Both are causally linked to the third variable: population

X^2 Correlation Test cont.

What does this mean?

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

1) Subtract the expected count (E) from the observed count (O) to find the difference between the two (also called the "residual").
2) Calculate the square of that number to get rid of positive and negative values (because the squares of 5 and –5 are, both 25).
3) Divide the result by the expected frequency to normalise bigger and smaller counts (because we don't want a formula that will give us a bigger X^2 value just because you're working with a bigger set of data).
4) The sigma, the sum of every *i* for which you calculate this relationship
   - Calculate this for each cell in the table, then add it all together. And that's it!

X^2 Correlation Test cont.

	Democrat	Republican	Total
Male	20 (25)	30 (25)	50
Female	30 (25)	20 (25)	50
Total	50	50	100

- X^2 value for our gender/party example is
  - ((20-25)^2/25) + ((30-25)^2/25) + ((30-25)^2/25) + ((20-25)^2/25), or
  - (25/25) + (25/25) + (25/25) + (25/25), or
  - 1 + 1 + 1 + 1,
  - which comes out to **4**.
- What does that mean??
  - The X^2 value serves as input for the more interesting piece of information: the p-value.
  - Calculating a p-value is less intuitive than a X^2 value
  - We simply use tools for calculating this data.

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

X^2 Correlation Test cont.

- X^2 value of 4, and
- Degree of freedon (df) of 1
  - For 2x2 table,
    - df=1=(2-1)(2-1)
- Use the p-value calculator: https://www.socscistatistics.com/pvalues/
- Gives us a p-value of 0.0455.
- This is interpreted as a 4.6% likelihood that the null hypothesis is correct.
- To put it best, if the distribution of this data is due entirely to chance, then you have a 4.6% chance of finding a discrepancy between the observed and expected distributions that is at least this extreme.
- x^2 value needed to reject the hypothesis is 3.84 (95% confidence level)
- Critical value of 3.84 < X^2 of 4
- So reject! The two attributes are dependent

X^2 Correlation Test cont.

- Look up the critical chi-square statistic value for $p = 0.05$ (95% confidence level) with 1 degree of freedom → 3.84

Degrees of Freedom	Probability										
	0.95	0.90	0.80	0.70	0.50	0.30	0.20	0.10	0.05	0.01	0.001
1	0.004	0.02	0.06	0.15	0.46	1.07	1.64	2.71	3.84	6.64	10.83
2	0.10	0.21	0.45	0.71	1.39	2.41	3.22	4.60	5.99	9.21	13.82
3	0.35	0.58	1.01	1.42	2.37	3.66	4.64	6.25	7.82	11.34	16.27
4	0.71	1.06	1.65	2.20	3.36	4.88	5.99	7.78	9.49	13.28	18.47
5	1.14	1.61	2.34	3.00	4.35	6.06	7.29	9.24	11.07	15.09	20.52
6	1.63	2.20	3.07	3.83	5.35	7.23	8.56	10.64	12.59	16.81	22.46
7	2.17	2.83	3.82	4.67	6.35	8.38	9.80	12.02	14.07	18.48	24.32
8	2.73	3.49	4.59	5.53	7.34	9.52	11.03	13.36	15.51	20.09	26.12
9	3.32	4.17	5.38	6.39	8.34	10.66	12.24	14.68	16.92	21.67	27.88
10	3.94	4.86	6.18	7.27	9.34	11.78	13.44	15.99	18.31	23.21	29.59
	Nonsignificant								Significant		

Critical values for the X^2 Distribution

[Source] https://www.itl.nist.gov/div898/handbook/eda/section3/eda3674.htm

Principal Component Analysis (PCA)

Dimensionality reduction technique which emphasises variation.

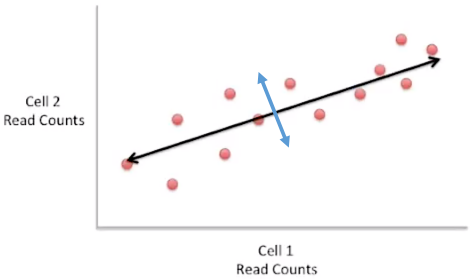
- Uses orthogonal transformation
- When to use:
  - Excessive multicollinearity - high intercorrelations or inter-associations among the independent variables.
  - Explanation of the predictors is not important
  - A light overhead in implementation is okay
  - More suitable for unsupervised learning

A PCA example

We'll start with just two cells.

Gene	Cell1 reads	Cell2 reads
a	10	8
b	0	2
c	14	10
d	33	45
e	50	42
f	80	72
g	95	90
h	44	50
i	60	50
... (etc)	... (etc)	... (etc)

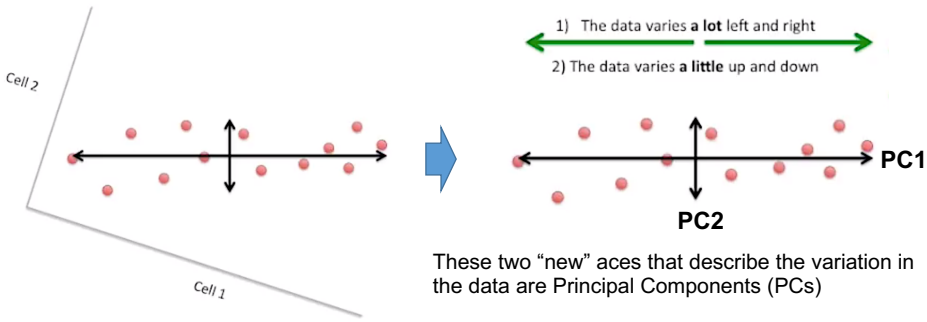
The dots are also spread out a little above and below the first line.



The 2nd largest amount of variation is at the endpoints of the newline.

Rotate the whole graph

The two lines that we drew make new X and Y axes.



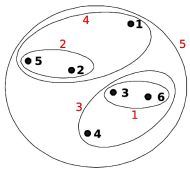
These two "new" aces that describe the variation in the data are Principal Components (PCs)

- PC1 is the axis that spans the most variation
- PC2 is the axis that spans the 2nd most variation.

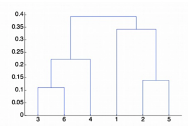
Cluster Analysis

- Dimensionality reduction technique which emphasizes correlation/similarity.
  - Identify groups of variables that are as correlated as possible among themselves and as uncorrelated as possible with variables in other clusters
- Produces a set of nested clusters organised as a hierarchical tree.
- Can be visualised as a dendrogram
- When to use:
  - Excessive multicollinearity
  - Explanation of the predictors is important

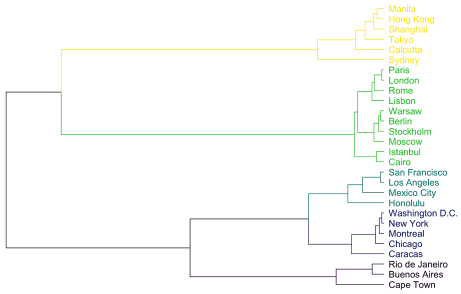
Cluster Analysis



Nested Clusters



Dendrogram



Correlation with the Target

Drop variables that have a very low correlation with the target.

- If a variable has a very low correlation with the target, it's not going to useful for the model (prediction).

Forward/Backward/Stepwise Selection

- **Forward Selection**
  - Identify the best variable (e.g., based on model accuracy)
  - Add the next best variable into the model
  - And so on until some predefined criteria is satisfied.
- **Backward Elimination (Recursive Feature Elimination)**
  - Start with all variables included in the model.
  - Drop the least useful variable (e.g. based on the smallest drop in model accuracy)
  - And so on until some predefined criteria is satisfied.
- **Stepwise Selection (combination of above)**
  - Similar to forward selection process, but a variable can also be dropped if its deemed as not useful any more after a certain number of steps.

Tree-based

Forests of trees to evaluate the importance of features

- Fit a number of randomized decision trees on various sub-samples of the dataset and use averaging to rank order features.

Labs (scikit-learn)

- Univariate Selection
- Recursive Feature Elimination
- Principal Component Analysis
- Feature Importance (Tree-based)

Univariate Selection

See also:

- **f_classif** — ANOVA F-value between label/feature for classification tasks.
- **mutual_info_classif** — Mutual information for a discrete target.
- **chi2** — Chi-squared stats of non-negative features for classification tasks.
- **f_regression** — F-value between label/feature for regression tasks.
- **mutual_info_regression** — Mutual information for a continuous target.
- **SelectPercentile** — Select features based on percentile of the highest scores.
- **SelectFpr** — Select features based on a false positive rate test.
- **SelectFdr** — Select features based on an estimated false discovery rate.
- **SelectFwe** — Select features based on family-wise error rate.
- **GenericUnivariateSelect** — Univariate feature selector with configurable mode.

SelectKBest class

```python
# Feature Extraction with Univariate Statistical Tests (Chi-
from pandas import read_csv
from numpy import set_printoptions
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
# load data
filename = 'pima-indians-diabetes.data.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pe
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
# feature extraction
test = SelectKBest(score_func=chi2, k=4)
fit = test.fit(X, Y)
# summarize scores
set_printoptions(precision=3)
print(fit.scores_)
features = fit.transform(X)
# summarize selected features
print(features[0:5,:])
```

```
[ 111.52  1411.887   17.605   53.108 2175.565  127.669    5.393
   181.304]
[[ 148.    0.   33.6  50. ]
 [  85.    0.   26.6  31. ]
 [ 183.    0.   23.3  32. ]
 [  89.   94.   28.1  21. ]
 [ 137.  168.   43.1  33. ]]
```

[Source] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest

28

Recursive Feature Elimination

RFE class

```python
# Feature Extraction with RFE
from pandas import read_csv
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
# load data
filename = 'pima-indians-diabetes.data.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
# feature extraction
model = LogisticRegression()
rfe = RFE(model, 3)
fit = rfe.fit(X, Y)
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
```

```
Num Features: 3
Selected Features: [ True False False False False True True False]
Feature Ranking: [1 2 3 5 6 1 1 4]
```

[Source] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html#sklearn.feature_selection.RFE

29

Principal Component Analysis

PCA class

```
# Feature Extraction with PCA
from pandas import read_csv
from sklearn.decomposition import PCA
# load data
filename = 'pima-indians-diabetes.data.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
# feature extraction
pca = PCA(n_components=3)
fit = pca.fit(X)
# summarize components
print("Explained Variance: %s" % fit.explained_variance_ratio_)
print(fit.components_)
```

```
Explained Variance: [ 0.88854663 0.06159078 0.02579012]
[[ -2.02176587e-03 9.78115765e-02 1.60930503e-02 6.07566861e-02
   9.93110844e-01 1.40108085e-02 5.37167919e-04 -3.56474430e-03]
 [  2.26488861e-02 9.72210040e-01 1.41909330e-01 -5.78614699e-02
  -9.46266913e-02 4.69729766e-02 8.16804621e-04 1.40168181e-01]
 [ -2.24649003e-02 1.43428710e-01 -9.22467192e-01 -3.07013055e-01
   2.09773019e-02 -1.32444542e-01 -6.39983017e-04 -1.25454310e-01]]
```

[Source] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

Feature Importance (Tree-based)

ExtraTreesClassifier() class - fits a number of randomised decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

```
# Feature Importance with Extra Trees Classifier
from pandas import read_csv
from sklearn.ensemble import ExtraTreesClassifier
# load data
filename = 'pima-indians-diabetes.data.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
# feature extraction
model = ExtraTreesClassifier()
model.fit(X, Y)
print(model.feature_importances_)
```

```
[ 0.11070069 0.2213717  0.08824115 0.08068703 0.07281761 0.14548537 0.12654214 0.15415431]
```

[Source] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html

Quiz

Why can't you prepare your machine learning algorithm on your training dataset and use predictions from this same dataset to evaluate performance?

Re-sampling

- Train and Test Sets
- *k*-fold Cross Validation
- Leave One Out Cross Validation
- Repeated Random Test-Train Splits

Train and Test Sets - train_test_split class

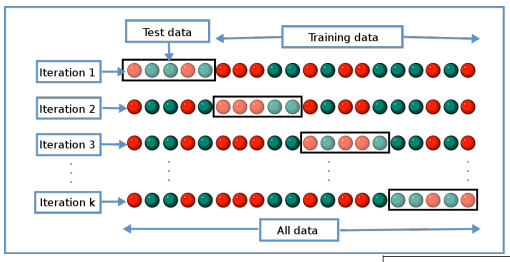
The simplest method - train the algorithm on the first part, make predictions on the second part and evaluate the predictions against the expected results.

- The size of the split varies - common to use 67%/33% splits

```
# Evaluate using a train and a test set
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
filename = 'pima-indians-diabetes.data.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
test_size = 0.33
seed = 7
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=test_size,
    random_state=seed)
model = LogisticRegression()
model.fit(X_train, Y_train)
result = model.score(X_test, Y_test)
print("Accuracy: %.3f%%" % (result*100.0))
```


k-Fold Cross Validation

Kfold class - Cross-validation is an approach that you can use to estimate the performance of a ML algorithm with less variance than a single train-test set split.




```
# Evaluate using Cross Validation
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
filename = 'pima-indians-diabetes.data.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
num_folds = 10
seed = 7
kfold = KFold(n_splits=num_folds, random_state=seed)
model = LogisticRegression()
results = cross_val_score(model, X, Y, cv=kfold)
print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))
```

Sample mean $\bar{x} = \frac{\sum x_i}{n}$

Variance $\sigma^2 = \frac{\sum (x_i - \bar{x})^2}{n-1}$

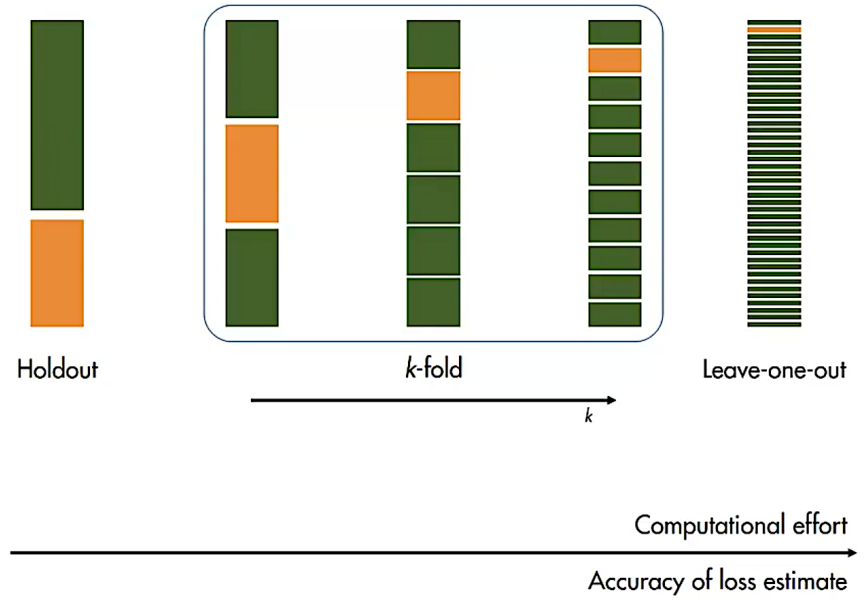
Standard deviation σ

holdout (10%)	cross-validation (10-fold)
75.3	73.8
77.9	75.0
80.5	75.5
74.0	75.5
71.4	74.4
70.1	75.6
79.2	73.6
71.4	74.0
80.5	74.5
67.5	73.0
$\bar{x} = 74.8$	$\bar{x} = 74.5$
$\sigma = 4.6$	$\sigma = 0.9$

Leave One Out Cross-Validation

- n-fold cross-validation ($n = total\ \#\ of\ instances$)
  - Predict each instance, training on all $(n-1)$ other instances
- Pros and cons:
  - Best possible learned: n-1 training examples
  - High computational cost: re-learn everything n times
  - Classes are not balanced in training / testing sets

```
# Evaluate using Leave One Out Cross Validation
from pandas import read_csv
from sklearn.model_selection import LeaveOneOut
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
filename = 'pima-indians-diabetes.data.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
loocv = LeaveOneOut()
model = LogisticRegression()
results = cross_val_score(model, X, Y, cv=loocv)
print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))
```

38

Repeated Random Test-Train Splits

Create a random split of the data like the train/test split, but repeat the process of splitting and evaluation of the algorithm multiple times, like CV.

- Pros and cons:
  - the reduction in variance in the estimated performance of k-fold cross-validation.
  - repeat the process many more times as needed to improve the accuracy.
  - repetitions may include much of the same data in the train or the test split from run to run - introducing redundancy into the evaluation.

39

Repeated Random Test-Train Splits

```
# Evaluate using Shuffle Split Cross Validation
from pandas import read_csv
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
filename = 'pima-indians-diabetes.data.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
n_splits = 10
test_size = 0.33
seed = 7
kfold = ShuffleSplit(n_splits=n_splits, test_size=test_size, random_state=seed)
model = LogisticRegression()
results = cross_val_score(model, X, Y, cv=kfold)
print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))
```


Questions?

paul@dcs.bbk.ac.uk

