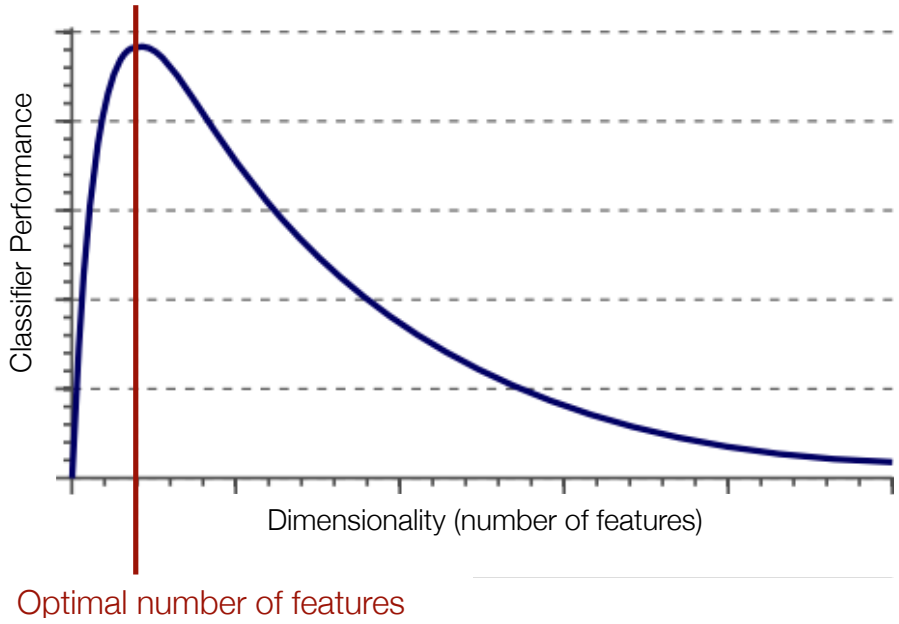
Dimension Reduction

Sarah Jane Delany

With material provided by Padraig Cunningham & Derek Greene

Curse of Dimensionality

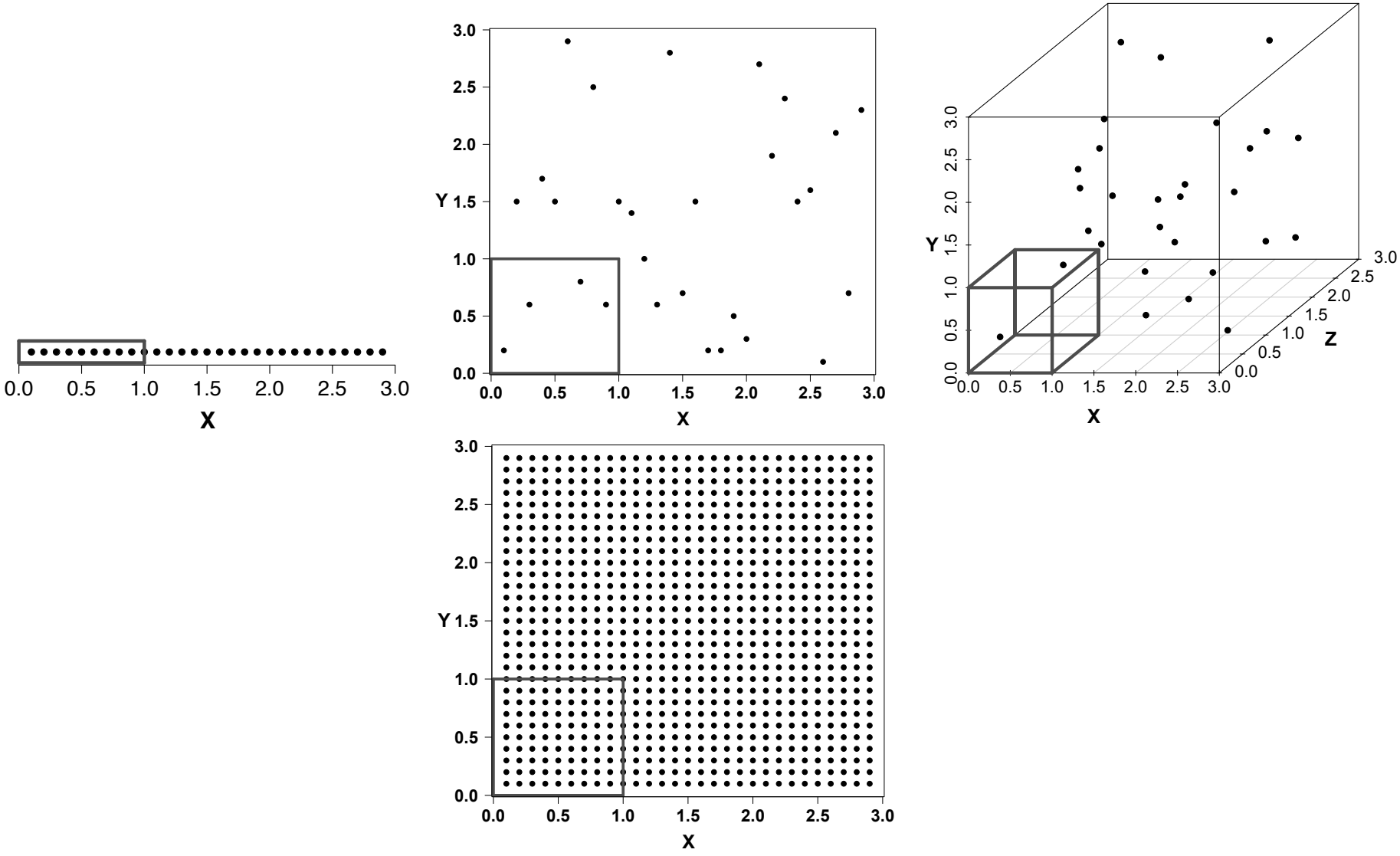
- Intuitively, adding more features (dimensions) to a dataset should provide more information about each example, making prediction easier.
- In reality, we often reach a point where adding more features no longer helps, or can even reduce predictive power.



- Curse of dimensionality: the phenomenon whereby many machine learning algorithms can perform poorly on high-dimensional data (Bellman, 1961).
- In theory, to build a good model, the number of examples required per feature increases exponentially with number of features.

Curse of Dimensionality

- Trade off between the number of descriptive features and the density of instances in the feature space



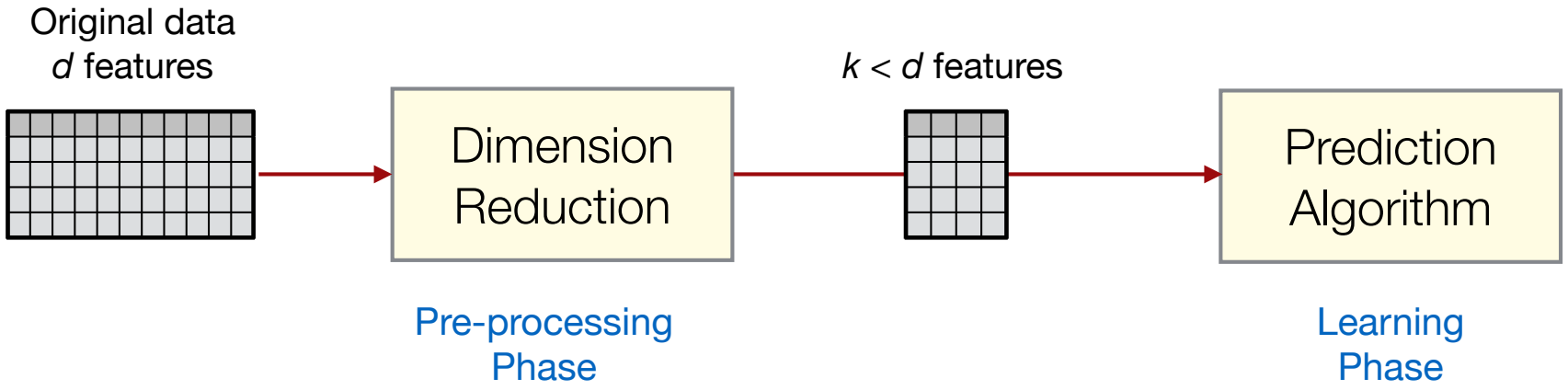
Dimension Reduction

- There are often other reasons why we might want to reduce the number of features used to represent data:
  1. **Computational cost**: For many algorithms, having a large number of features can significantly increase the running time and memory usage.
  2. **Financial cost**: In certain domains (e.g. clinical medicine, manufacturing), running experiments to generate feature values can be expensive. In such cases, we only want to generate the minimum number of features required to build a good model.
  3. **Interpretability**: A feature set which is more compact can help to give a better understanding of the underlying process that generated the data.

➡ Understanding which features in our data are informative and which are not is an important knowledge discovery task.

Dimension Reduction

- Basic idea to try to beat the curse of dimensionality:
  1. Apply pre-processing techniques to reduce the number of features used to represent a dataset.
  2. Then build a model on the smaller feature set.
- In many (but not all) cases, the additional information that is lost by removing some features is (more than) compensated by higher prediction accuracy in the lower dimensional space.



Dimension Reduction Strategies

There are two general strategies for dimension reduction:

1. Feature Transformation

   Transforms the original features of a dataset to a completely new, smaller, more compact feature set, while retaining as much information as possible.
   e.g. Principal Components Analysis (PCA), Linear Discriminant Analysis (LDA)

2. Feature Selection

   Tries to find a minimum subset of the original features that optimises one or more criteria, rather than producing an entirely new set of dimensions for the data.

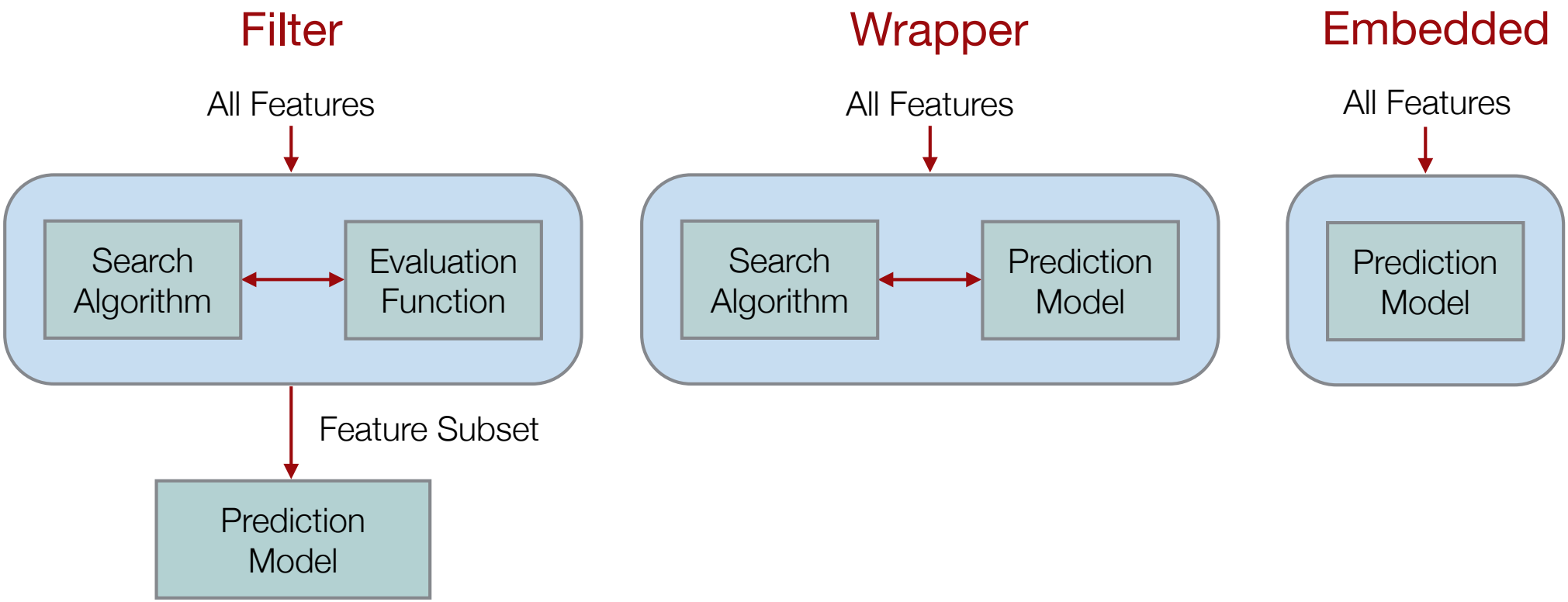
   e.g. Information Gain filter, Wrapper with sequential forward selection

Feature Selection

- Feature Subset Selection: Find the best subset of all available features, which contains the smallest number of features that most contribute to accuracy. Discard the remaining, unimportant features.
- **Why select subset of original features?**
  1. Building a better predictor - Redundant or noisy features can damage accuracy.
  2. Knowledge discovery - Identifying useful features helps us learn more about the data.
  3. Features expensive to obtain - Test a large number of features, select a few for the final system (e.g. sensors, manufacturing).
  4. Interpretability - Selected features still have meaning.

Feature Selection Strategies

- Finding the optimal feature subset for a given dataset is difficult.
- Brute force evaluation of all feature subsets involves $\binom{d}{k}$ combinations if k is fixed, or 2^d combinations if not fixed.
- Three broad strategies for feature selection:



Filter Feature Selection

- Pre-processing step that ranks and “filters” features independently of the choice of classifier that will be subsequently applied.
- **Evaluation function:** How does a filter algorithm score different feature subsets to produce an overall ranking?
- Generally score the predictiveness of the features.
  - Information theoretic analysis e.g. Information Gain, Gini index
  - Statistical tests e.g. Chi-square statistic
- Score using models, e.g. Random Forests, SVMs, …

How many features?

- Filter approaches can either
  - Rank features where you have to determine a threshold

    e.g. Information Gain
  - Provide a set of features due to some intrinsic element of the approach

    e.g. Chi-square statistic (p value)

Recap: Information Gain

- IG for descriptive feature d that splits a dataset D of examples into subsets or partitions $\{D_1, D_2, \ldots, D_k\}$

$$IG(d, D) = (\text{original entropy}) - (\text{entropy after split})$$

The entropy remaining after the dataset is split using descriptive feature *d*

$$IG(d, D) = H(D) - rem(d, D)$$

$$H(D) = -\sum_{i=1}^{l} (P(t_i) \times log_s(P(t_i)))$$

The entropy on the full dataset wrt the target feature *t*

$$rem\,(d, \mathcal{D}) = \sum_{i}^{k} \underbrace{\frac{|\mathcal{D}_i|}{|\mathcal{D}|}}_{\text{weighting}} \times \underbrace{H\,(\mathcal{D}_i)}_{\substack{\text{entropy of} \\ \text{partition } \mathcal{D}_i}}$$

Each partition is weighted in proportion to its size

Information Gain Filter

- A feature d that is predictive of the target will have significant Information Gain (i.e. a reduction in uncertainty):

$$IG(d, \mathcal{D}) = H(\mathcal{D}) - \sum_{i}^{k} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} \times H(t, \mathcal{D}_i)$$

- **IG Filter approach**:
  1. Score all features based on their Information Gain (IG).
  2. Rank features based on their scores.
  3. Select a subset of the top ranked features to use for prediction.

Information Gain Filter in scikit learn

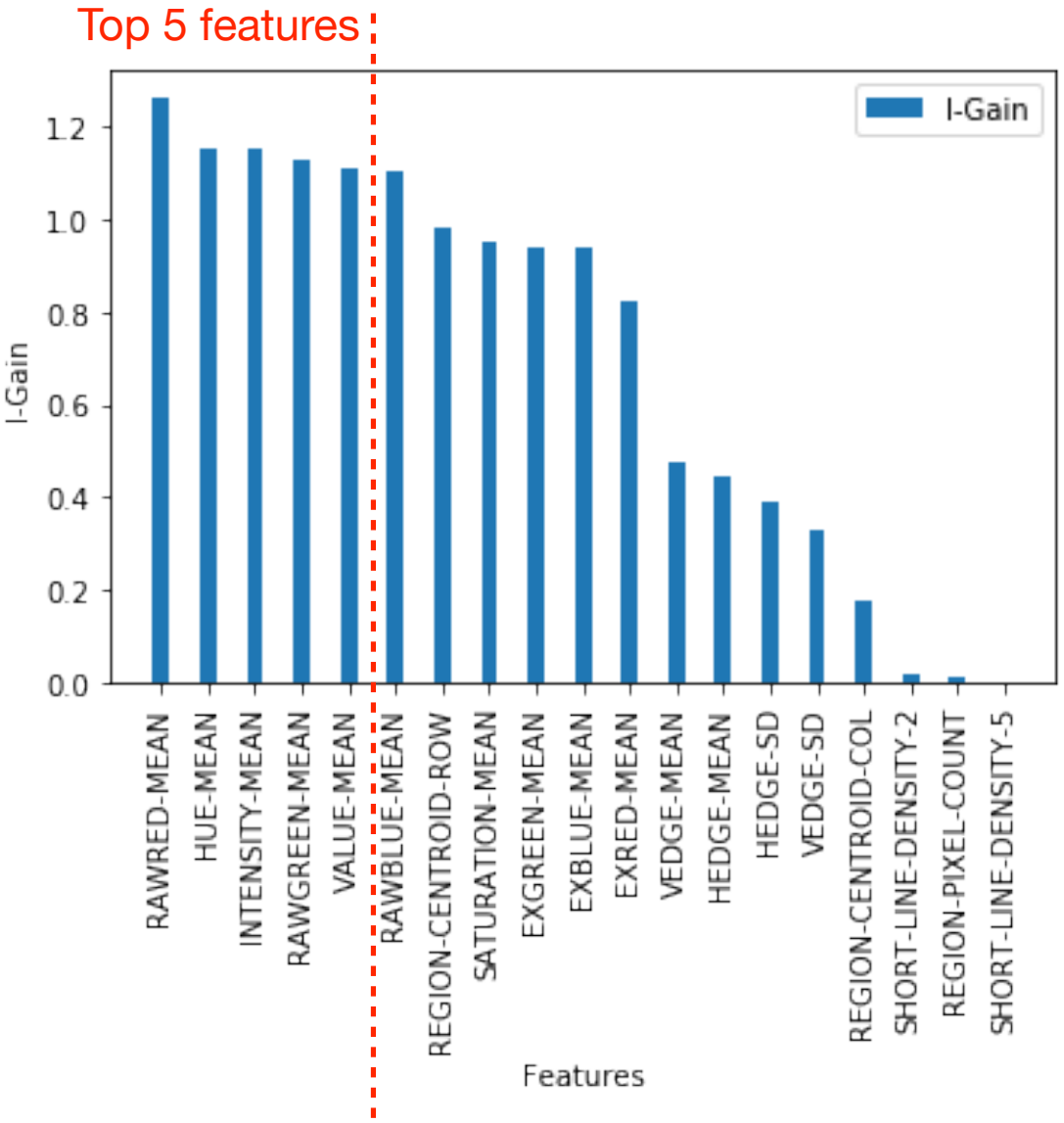
- Score each feature on information gain
- Sort data frame on this column

```
from sklearn.feature_selection import SelectKBest, mutual_info_classif

mi = dict()

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    random_state=2,
                                                    test_size=1/2)
i_scores = mutual_info_classif(X_train, y_train)

for i,j in zip(seg_data.columns,i_scores):
    mi[i]=j

df = pd.DataFrame.from_dict(mi,orient='index',columns=['I-Gain'])
df.sort_values(by=['I-Gain'],ascending=False,inplace=True)
df.head(10)
```

Code Notebook:
09 Feature Selection

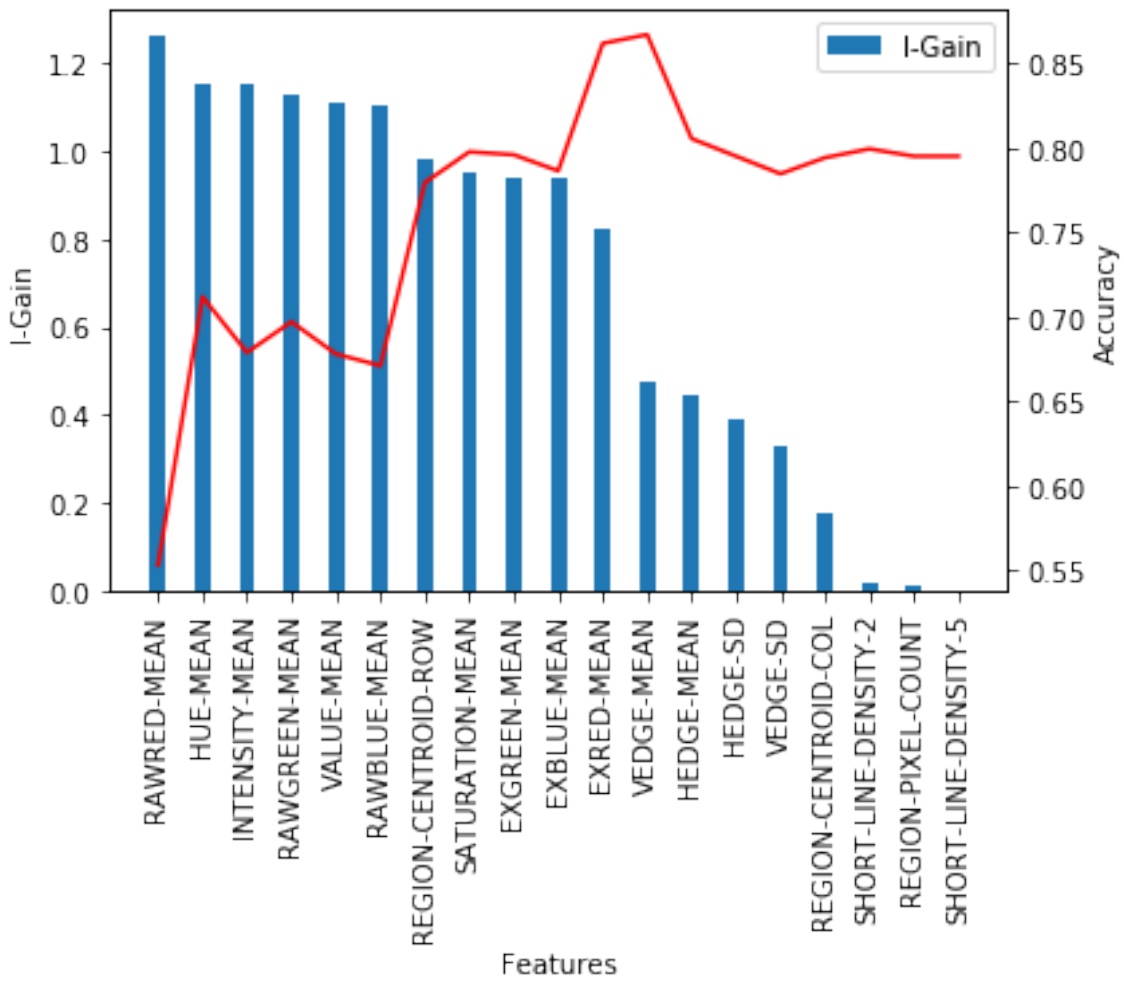
Filters - Selecting Top Features

- Several basic options for step 3…
  - Select the top ranked k features.
  - Select top x%, e.g. 50%
  - Select features with IG > 50% of max IG score.
  - Subset of features with non-zero IG scores.



Filters - Selecting Top Features

- Better strategy for selecting k top features: evaluate classification performance using feature subsets of increasing size.
- Start with feature with highest Information Gain, then add the next feature.
- Measure the accuracy for each subset using cross-validation.
- Choose the final subset giving the highest accuracy.



Filter Feature Selection in scikit learn

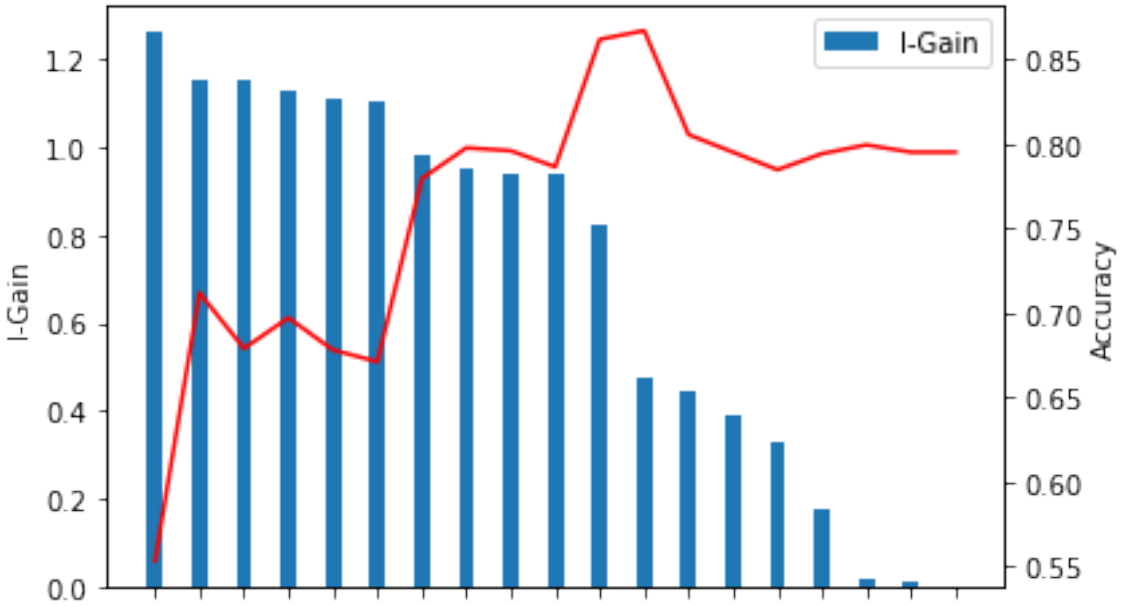
Code Notebook:
09 Feature Selection

Set up data transformer

Transform train & test data

Test feature subset

```
acc_scores = []
for kk in range(1, X.shape[1]+1):
    FS_trans = SelectKBest(mutual_info_classif,
                           k=kk).fit(X_train, y_train)
    X_tR_new = FS_trans.transform(X_train)
    X_tS_new = FS_trans.transform(X_test)
    seg_NB = mnb.fit(X_tR_new, y_train)
    y_dash = seg_NB.predict(X_tS_new)
    acc = accuracy_score(y_test, y_dash)
    acc_scores.append(acc)

df['Accuracy'] = acc_scores
df.head(10)
```



	I-Gain	Accuracy
RAWRED-MEAN	1.259301	0.553247
HUE-MEAN	1.152770	0.711688
INTENSITY-MEAN	1.152625	0.678788
RAWGREEN-MEAN	1.126327	0.696970
VALUE-MEAN	1.107112	0.677922
RAWBLUE-MEAN	1.099933	0.670996
REGION-CENTROID-ROW	0.982181	0.779221
SATURATION-MEAN	0.952746	0.797403
EXGREEN-MEAN	0.938691	0.795671
EXBLUE-MEAN	0.938304	0.786147

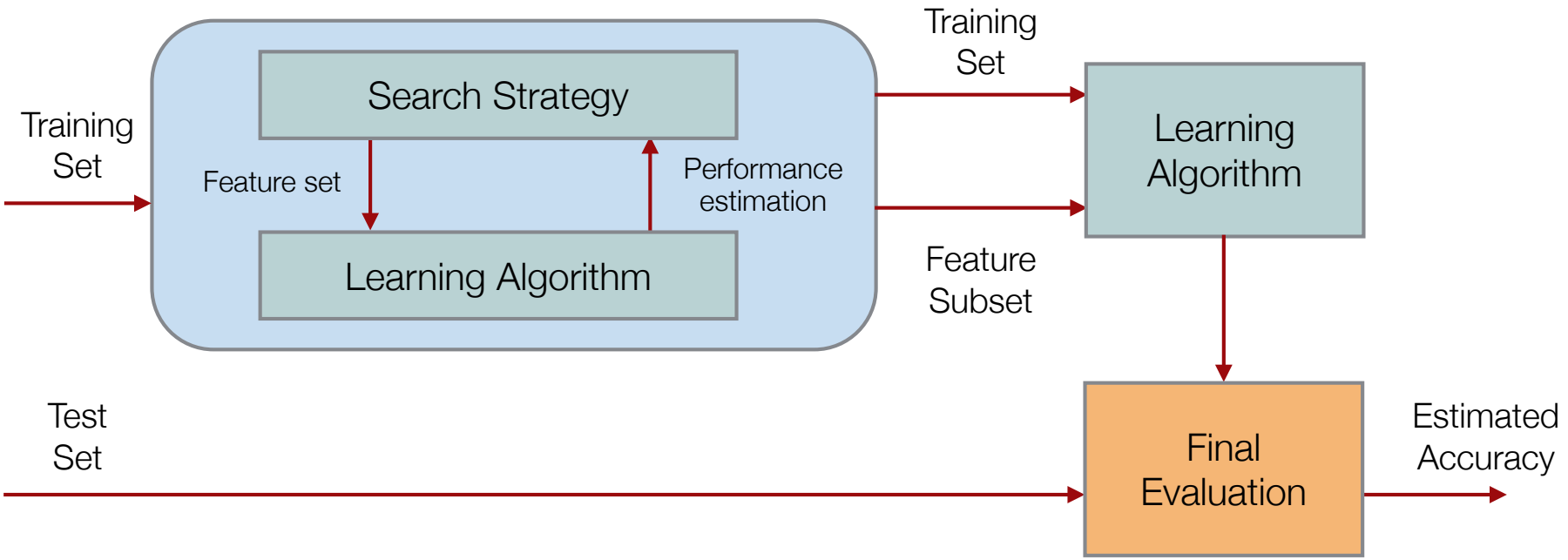
Filters - Disadvantages

Key problems with filter feature selection:

1. **No Model Bias**
   - Different features may suit different learning algorithms (Neural networks, Decision Trees, K-NN, etc.), but filters do not take this into account.
2. **No Feature Dependencies**
   - In filters, the features are considered in isolation from one another, and are not considered in context.
   - In some cases, a filter might select two predictive but correlated features, where one would be sufficient.
   - In other cases, one feature needs another feature to boost accuracy, but a filter cannot discover this.

Wrappers

- Alternative strategy: the classifier is “wrapped” in the feature selection mechanism. Feature subsets are evaluated directly based on their performance when used with that specific classifier.
- Key advantages:
  1. Takes bias of specific learning algorithm into account.
  2. Considers features in context - i.e. feature dependencies.



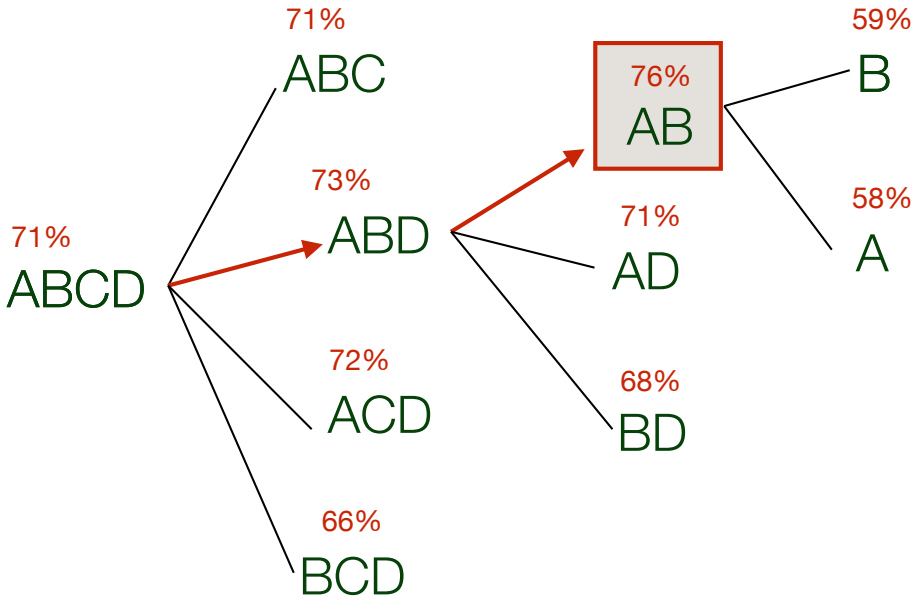
Feature Subset Search

- A key aspect of wrappers is the search strategy which generates the candidate feature subsets for evaluation.
- There are many different ways to search the space of all possible subsets. The choice of a suitable search strategy often depends on the number of features *d* in the data. Several general approaches:
  - Exhaustive search: Evaluate every possible subset of features. For *d* features there are 2^d potential subsets. For even small values of *d*, an exhaustive search over this huge space is intractable.
  - Exponential search: Returns the optimal feature subset, but uses heuristics so that not every one of the 2^d possible subsets needs to be evaluated.
  - Sequential search: Fast search algorithms that choose a subset by adding or removing one feature at a time. Not guaranteed to final the optimal feature subset, but widely used.

Sequential Search

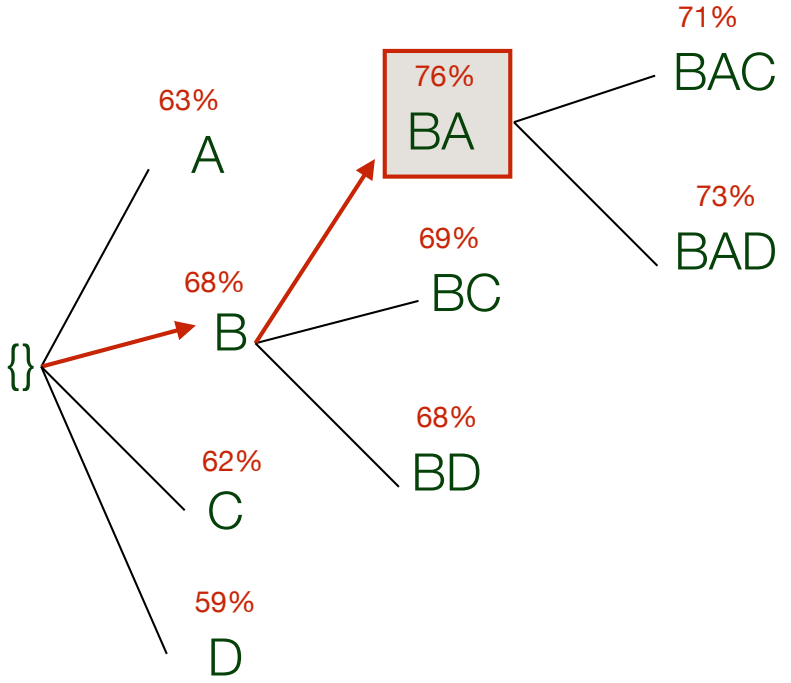
- In sequential search, performance estimation guides the subset search process (e.g. overall classification accuracy).
- The most basic sequential search methods use a heuristic stepwise approach, either:
  - Forward Sequential Selection
    - Start with an empty subset.
    - Find the most informative feature and add it to the subset.
    - Repeat until there is no improvement by adding features.
  - Backward Elimination
    - Start with the complete set of features.
    - Remove the least informative feature.
    - Repeat until there is no improvement by dropping features.

Sequential Search



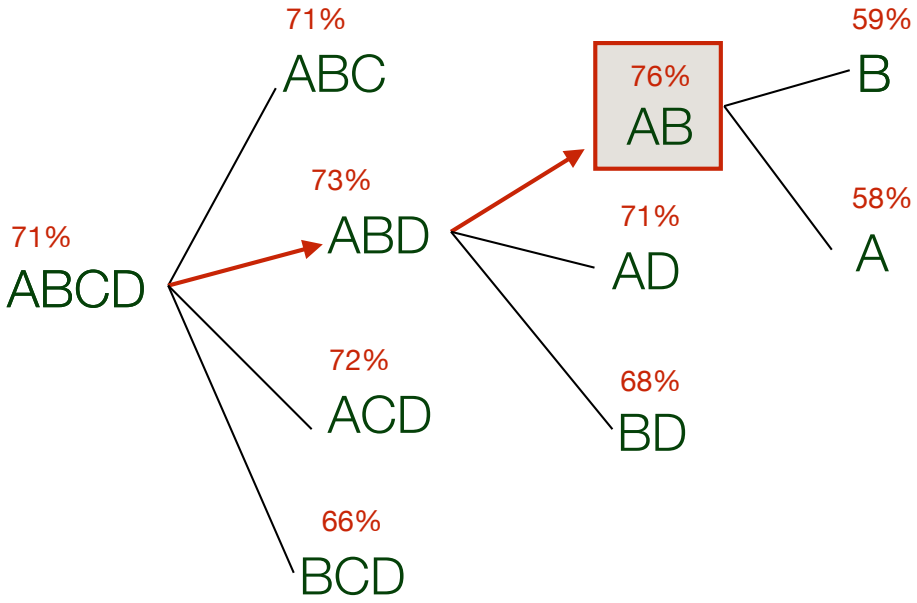
Example: Backward elimination from 4 features (ABCD) to 2 features (AB).

Sequential Search

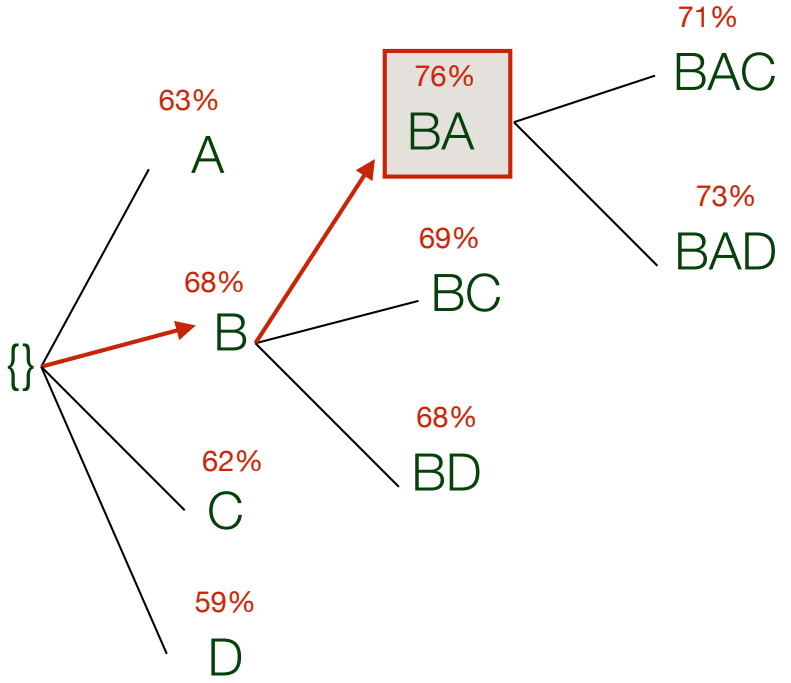


Example: Forward selection from no features to 2 features (BA).

Sequential Search



Example: Backward elimination from 4 features (ABCD) to 2 features (AB).



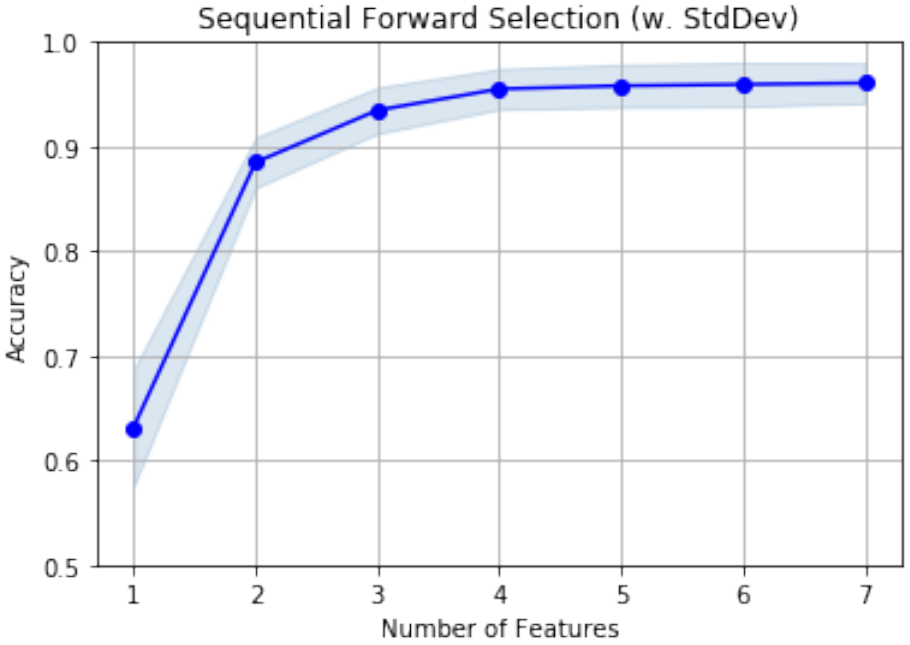
Example: Forward selection from no features to 2 features (BA).

- Backward elimination tends to find better models, can find subsets with interacting features. But tends to be slower.
- Forward selection starts with small subsets, so requires less running time if stopped early.

Wrapper Feature Selection with MLxtend

- Forward Sequential Search

```
from sklearn.neighbors import KNeighborsClassifier
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
sfs_forward = SFS(knn,
                  k_features=7,
                  forward=True, floating=False,
                  verbose=1, scoring='accuracy',
                  cv=10, n_jobs = -1)

sfs_forward = sfs_forward.fit(X, y,
                              custom_feature_names=feature_names)
```




```
'REGION-CENTROID-ROW',
‘VEDGE-MEAN',
'HEDGE-MEAN',
'RAWRED-MEAN',
'EXRED-MEAN',
'EXGREEN-MEAN',
'SATURATION-MEAN'
```

Code Notebook:
09 Feature Selection

Wrappers - Disadvantages

There are two main disadvantages with wrapper methods:

1. Computational Cost

- Significant running time, particularly when the number of features is large. Far slower than filters.
- Computational bottleneck is the requirement to retrain the model many different times on different feature subsets.

2. Overfitting

- Risk of overfitting, particularly when the number of training examples is insufficient.
- We can end up finding the best feature subset for the training data, which does not work well for new unseen data.

Embedded Methods

- Embedded approaches apply the feature selection process as an integral part of the learning algorithm.
- A typical embedded method is Decision Tree learning.
  - features are selected at each step in the process

- Other embedded methods:
  - Lasso Regression
  - Random Forests
  - Gradient Boost

Aside: Regularisation

- Regularisation is used to avoid overfitting by modifying the learning algorithm to create models that are stable with respect to changes in the input, effectively simplifying the model
  - pruning decision trees by early stopping
  - adding a penalty term to the loss function to constrain the value of the weights (neural networks & regression)
  - hard/soft optimisation in SVM, depending on the extent of misclassification permitted, resulting in a wider or narrower margin separating the hyperplane (C parameter)

LASSO Regression

- **Lasso** (Least Absolute Shrinkage and Selection Operator) is Regression with L1 regularisation
- L1 regularisation adds a penalty term to the loss function which uses the *absolute* values of the weights (called shrinkage methods)
- Minimise the error/loss function, SSE or L_2

$$L_2(\mathbb{M}_{\mathbf{w}}, \mathcal{D}) = \frac{1}{2}\sum_{i=1}^{n}\left(t_i - \sum_{j=1}^{k}\mathbf{w}_j \times \mathbf{d}_{ij}\right)^2 + \lambda\sum_{j=1}^{k}|\mathbf{w}_j|$$

regularisation parameter

error

penalty term

- Penalty term drives the weights to zero
- Higher values of λ, fewer features have non-zero values

Lasso in scikit-learn

Code Notebook:
09 Feature Selection

```
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LogisticRegression

lr_selector = SelectFromModel(LogisticRegression(penalty="l1",
                        C=.001, solver="liblinear"), max_features=X.shape[1])
lr_selector.fit(X, y)
```

```
11 selected features
Selected features:

[0, 1, 2, 6, 8, 10, 12, 13, 14, 15, 16]
```

REGION-CENTROID-COL
REGION-CENTROID-ROW
REGION-PIXEL-COUNT
VEDGE-SD
HEDGE-SD
RAWRED-MEAN
RAWGREEN-MEAN
EXRED-MEAN
EXBLUE-MEAN
EXGREEN-MEAN
VALUE-MEAN

Dimension Reduction Strategies

Feature Selection

- Tries to find a minimum subset of the original features that optimises one or more criteria, rather than producing an entirely new set of dimensions for the data.

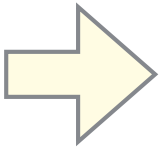
Feature Transformation

- A popular alternative strategy in data pre-processing is to transform the data into an entirely different format.
- Examples represented by one set of features are transformed to another new set of features.
- Resulting features can be more compact and less noisy, resulting in more accurate predictions.
- Typically involve a linear transformation of the original data.

Feature Selection v Transformation

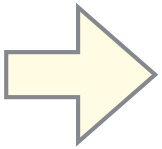
	region-centroid-col	region-centroid-row	region-pixel-count	vedge-mean	vedge-sd	hedge-mean	hedge-sd	intensity-mean
x1	218.0	178.0	9.0	0.8	0.5	1.1	0.5	59.6
x2	113.0	130.0	9.0	0.3	0.3	0.3	0.4	0.9
x3	202.0	41.0	9.0	0.9	0.8	1.1	1.0	123.0
x4	32.0	173.0	9.0	1.7	1.8	9.0	6.7	43.6
x5	61.0	197.0	9.0	1.4	1.5	2.6	1.9	49.6
x6	149.0	185.0	9.0	1.6	1.1	3.1	1.9	49.3
x7	197.0	229.0	9.0	1.4	1.6	1.2	0.6	17.7
x8	29.0	111.0	9.0	0.4	0.2	0.6	0.2	5.4
...	...	...	...	...	...	...	...	...

Feature Selection:
Select subset of the original features, use them to represent the data.



	region-centroid-row	hedge-mean	hedge-sd	intensity-mean
x1	178.0	1.1	0.5	59.6
x2	130.0	0.3	0.4	0.9
x3	41.0	1.1	1.0	123.0
x4	173.0	9.0	6.7	43.6
x5	197.0	2.6	1.9	49.6
x6	185.0	3.1	1.9	49.3
x7	229.0	1.2	0.6	17.7
x8	111.0	0.6	0.2	5.4
...	...	...	...	...

Feature Transformation:
Create a new set of dimensions, use them to represent the data.

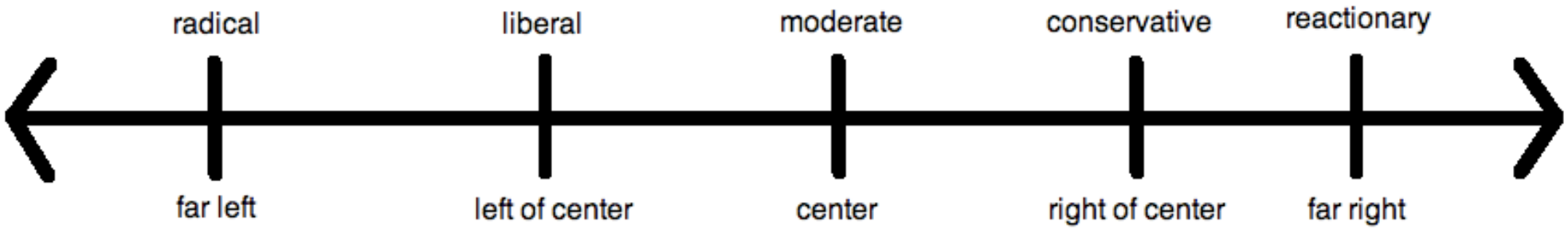


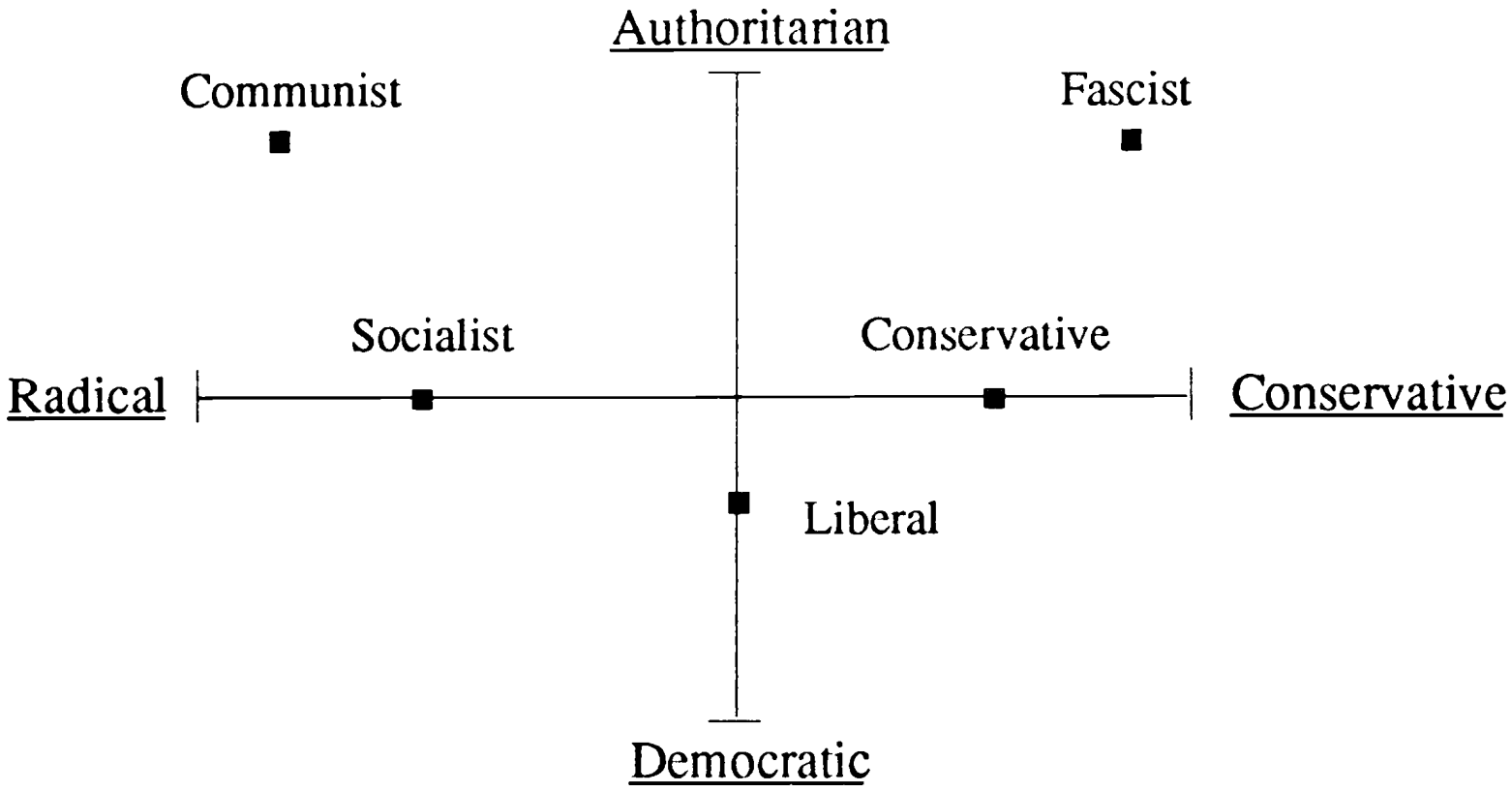
	PC1	PC2	PC3	PC4
x1	95.04	19.51	36.68	37.97
x2	-13.34	11.55	19.53	-29.57
x3	76.50	-2.37	-112.33	41.45
x4	-90.95	5.04	44.07	29.96
x5	-61.17	13.49	61.95	43.10
x6	26.18	16.23	49.07	34.47
x7	74.49	26.46	100.29	21.20
x8	-97.65	5.20	2.54	-29.51
...	...	...	...	...

- 2D projection of a 3D object

But

- *Intrinsic* dimension of surface of the Earth is 2D.
- Earth surface occupies a 2D *manifold* in a 3D space.


radical
liberal
moderate
conservative
reactionary
far left
left of center
center
right of center
far right


Authoritarian
Communist
Fascist
Socialist
Conservative
Radical
Conservative
Liberal
Democratic

Linear Transformations

- In a linear transformation, we map data to new variables, which are linear functions of the original variables.
- **Example:** Bill owes Mary £5 and CHF15 after a holiday. Current exchange rates:

  €:£ → 1.25:1, €:CHF → 0.8:1

- Based on rates, Bill owes €(1.25 x 5 + 0.8 x 15) = €18.25
- We can view this as a linear transformation…

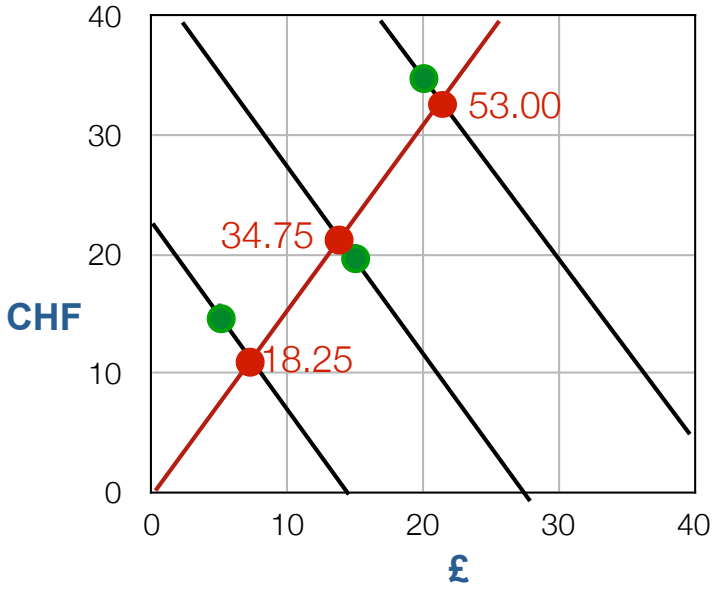
£	CHF
5	15
15	20
20	35

x

1.25
0.8

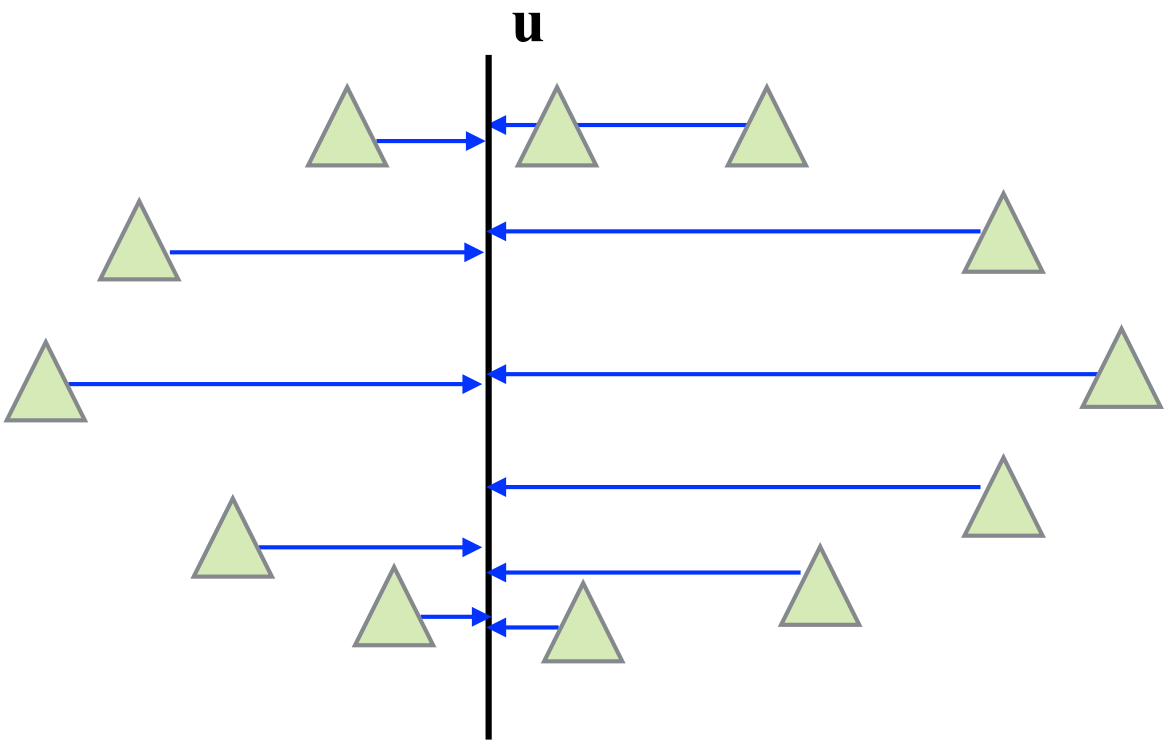
=

€
18.25
34.75
53.00



Projection Methods

- Projection methods map the original *d*-dimensional space to a new *(k < d)*-dimensional space, with the minimum loss of information.
- "Good" spaces for projections are characterised by preserving most of the useful information in the data - the variation in the data.



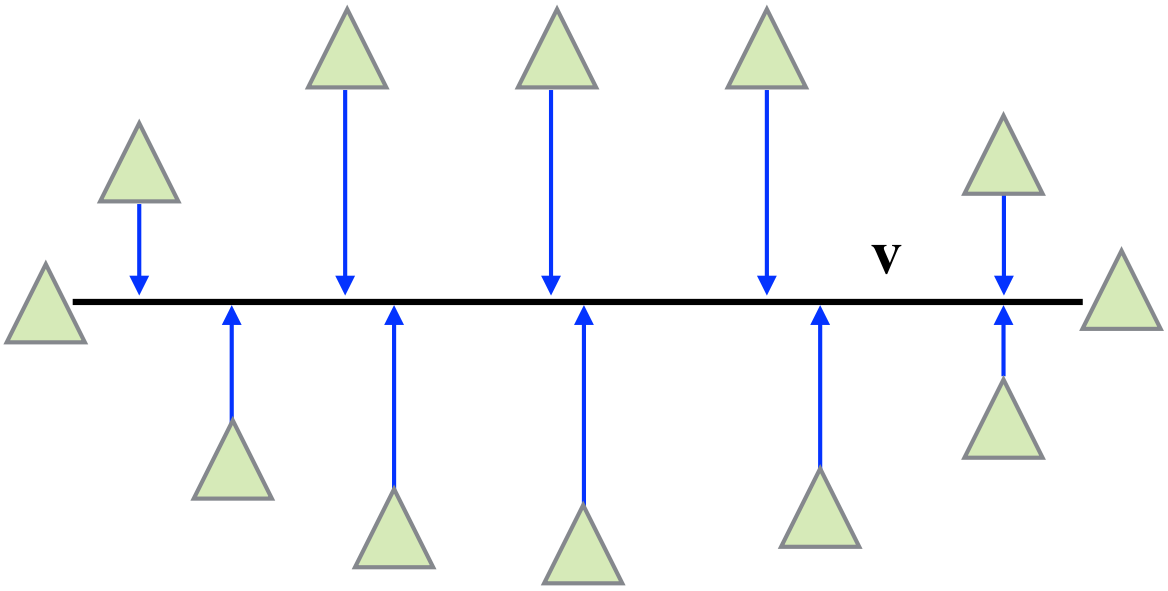
georgemdallas.wordpress.com

If we project the original data to this new dimension ***u***, the data is not very spread out.

The dimension does not have high variance.

Projection Methods

- Projection methods map the original *d*-dimensional space to a new *(k < d)*-dimensional space, with the minimum loss of information.
- "Good" spaces for projections are characterised by preserving most of the useful information in the data - the variation in the data.

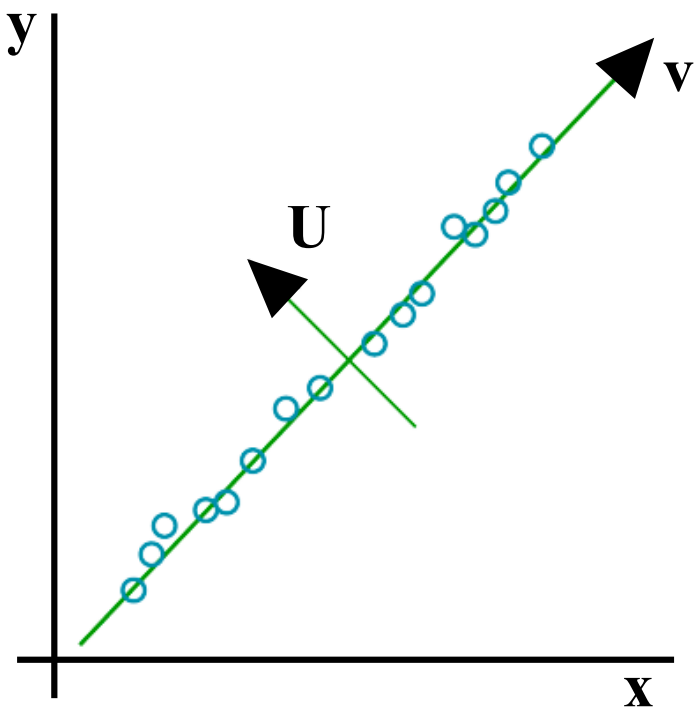


If we project to the alternative new dimension v, the data is far more spread out.

This new dimension has more variation - i.e. more information has been preserved.

Principal Component Analysis (PCA)

- To minimise the loss of information, we need to find new dimension(s) which maximise the variation.



Given data in terms of dimensions *(x,y)*, the principal direction in which the data varies is along the ***v*** axis.

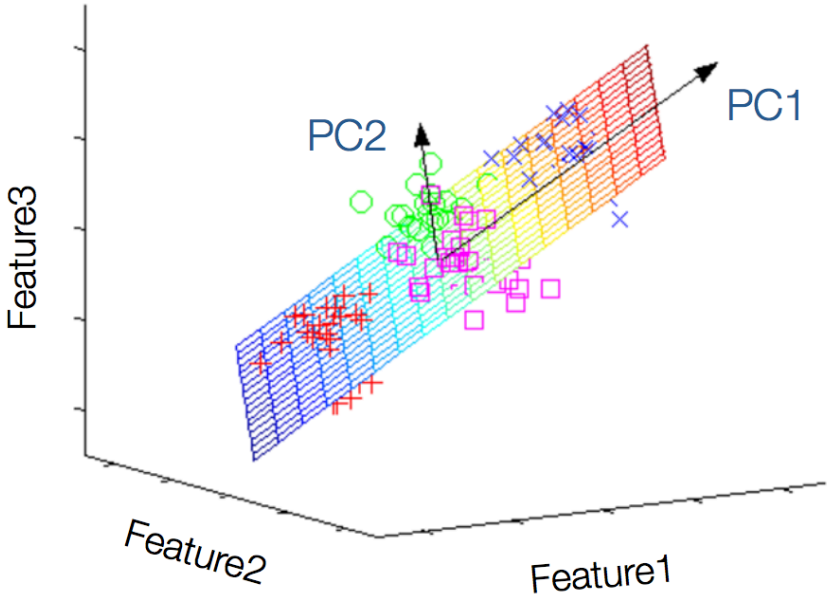
Very little variation in the data in the direction of ***u***.

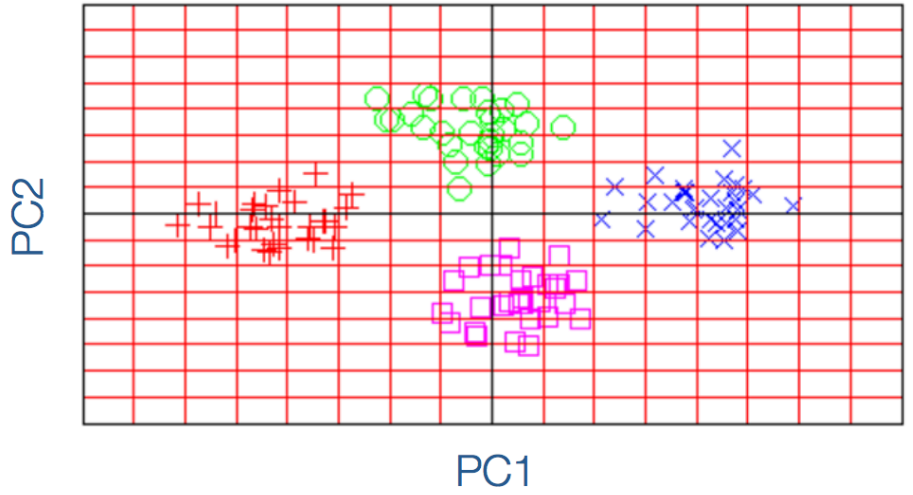
➡ Select the ***v*** dimension to represent the data.

- Use Principal Component Analysis (PCA) - an unsupervised projection method which performs dimensionality reduction, while trying to keep as much of the variance in the data as possible.

Principal Component Analysis (PCA)

- **Example:** Transformation of original data (3 dimensions) to a lower dimensional space (2 dimensions) via PCA.
- Using PCA, we can identify the two-dimensional plane that optimally describes the highest variance of the data. Each resulting dimension is a linear combination of the original 3 dimensions.





Eigenvectors and Eigenvalues

- Given an input matrix $\mathbf{X}$, an eigenvector of the matrix is a non-zero vector $\mathbf{v}$ that satisfies the equation:

$$\mathbf{X}v = \lambda v$$

  where the corresponding number λ is called an eigenvalue.

- Eigendecomposition is the factorisation of a matrix into its eigenvalues and eigenvectors.

$$\mathbf{X} = \begin{pmatrix} 1.0000 & 0.5000 & 0.3330 & 0.2500 \\ 0.5000 & 1.0000 & 0.6667 & 0.5000 \\ 0.3333 & 0.6667 & 1.0000 & 0.7500 \\ 0.2500 & 0.5000 & 0.7500 & 1.0000 \end{pmatrix}$$

4 x 4 symmetric matrix

$$\mathbf{\Lambda} = \begin{pmatrix} 2.5361 & 0 & 0 & 0 \\ 0 & 0.8483 & 0 & 0 \\ 0 & 0 & 0.4078 & 0 \\ 0 & 0 & 0 & 0.2077 \end{pmatrix}$$

4 eigenvalues

Eigenvectors and values exist in pairs: every eigenvector has a corresponding eigenvalue.

$$\mathbf{V} = \begin{pmatrix} -0.37775 & -0.81052 & -0.44217 & -0.06988 \\ -0.53223 & -0.18762 & 0.74199 & 0.36221 \\ -0.56139 & 0.30099 & 0.04872 & -0.76926 \\ -0.50881 & 0.46611 & -0.50155 & 0.52169 \end{pmatrix}$$

4 eigenvectors

Eigendecomposition in *sklearn*

```
X = np.array([[1, 0.5, 0.333, 0.25],
              [0.5, 1, 0.667, 0.5],
              [0.333, 0.667,1, 0.75],
              [0.25, 0.5, 0.75, 1]
             ])
In [24]:
eVal, eVec = np.linalg.eig(X)
In [25]:
vec1 = eVec[:,0]
val1 = eVal[0]
```

$$\mathbf{X}v = \lambda v$$

```
X.dot(vec1)
Out[37]:
array([-0.95799966, -1.34996974, -1.42377554, -1.29038555])

val1 * vec1
Out[38]:
array([-0.95799966, -1.34996974, -1.42377554, -1.29038555])
```

Notebook 09 Eigen

Eigenvectors in PCA

- Each eigenvector has a direction - i.e. it is a dimension.
- Eigenvectors of symmetric matrices are orthogonal to each other - i.e. they point in completely different directions.
- Each eigenvalue is a number indicating how much variance there is in the data in that direction.
- The eigenvector with the largest eigenvalue has the most variance, making it a useful dimension for projection.

- Principal Components (PCs): New dimensions constructed as linear combinations of the original features, which are uncorrelated with one another. Constructed from eigenvectors.
- The first PC accounts for the most variability in the data. The next PC has the highest variance possible under the constraint that it is uncorrelated with the first PC, and so on…

Covariance Matrix

- To assess variability in the data, we can measure the covariance between the original features - i.e. the tendency for two features x and y to vary in the same direction:
  - Do features x and y tend to increase together?
  - Or does feature y decrease as feature x increases?

$$cov_{x,y} = \frac{\sum_1^n (x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

- We estimate the covariances based on all *n* examples.
- By measuring the covariance between all pairs of features, we can create a symmetric covariance matrix.

Original
n x d
matrix

$$\begin{bmatrix} c_{11} & c_{12} & .. & c_{1d} \\ c_{21} & c_{22} & .. & c_{2d} \\ .. & .. & .. & .. \\ c_{d1} & c_{d2} & .. & c_{dd} \end{bmatrix}$$

Applying PCA

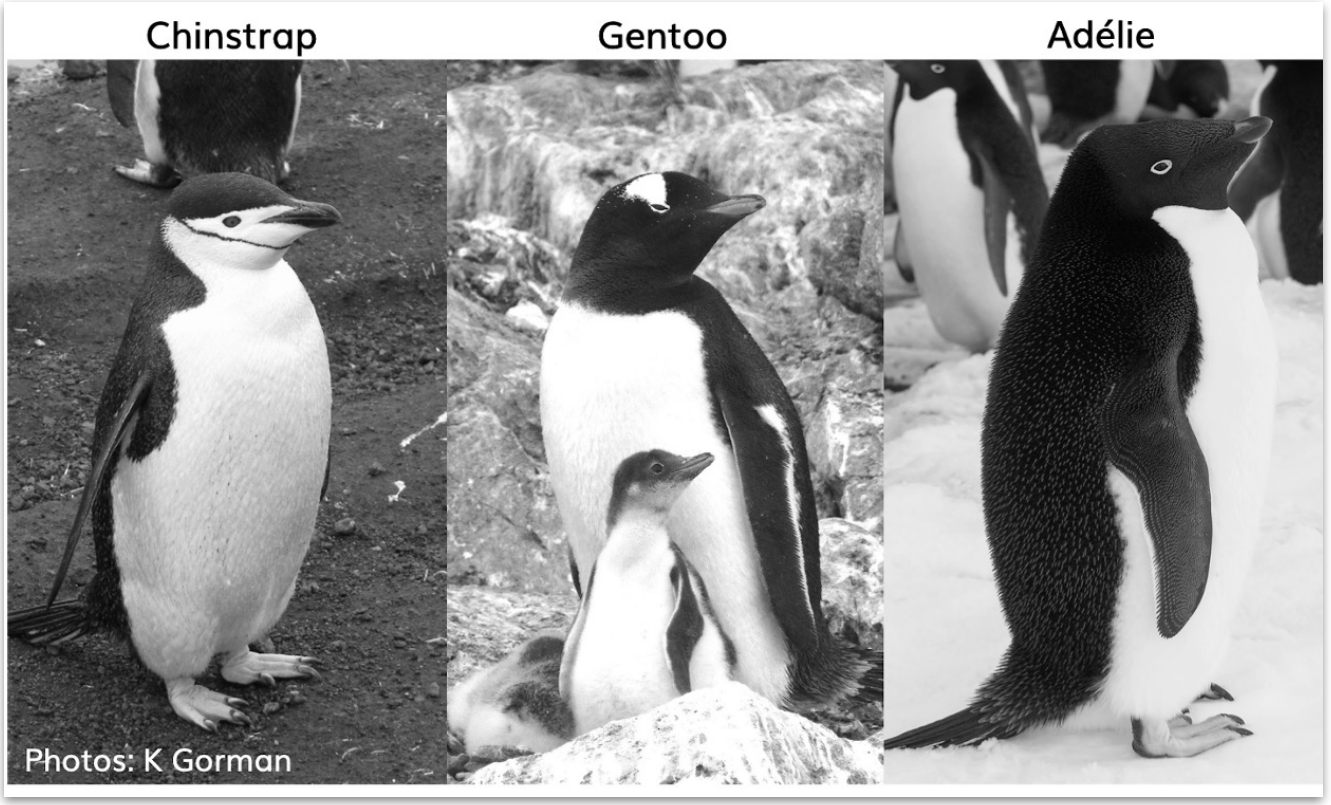
- **Input:** A dataset matrix $\mathbf{X}$ with n examples (i.e. rows). The features of this matrix (i.e columns) might potentially be correlated.
- **PCA Process:**
  1. Calculate the mean of the columns of $\mathbf{X}$.
  2. Subtract the column means from each row of $\mathbf{X}$, to create the centred matrix $\mathbf{Y}$.
  3. Calculate the covariance matrix $\mathbf{C} = \mathbf{Y}^{\mathsf{T}}\mathbf{Y}/(n-1)$
  4. Calculate the eigenvectors of the covariance matrix $\mathbf{C}$.
  5. The Principal Components (PCs) are given by the eigenvectors of $\mathbf{C}$. The *i*-th PC is given by the eigenvector corresponding to the *i*-th largest eigenvalue of $\mathbf{C}$.
  6. Select an appropriate number of PCs k and use them as a new reduced n x k representation of the dataset.

Penguin dataset

Kristen Gorman

Allison Horst

https://github.com/allisonhorst



3 classes, four features

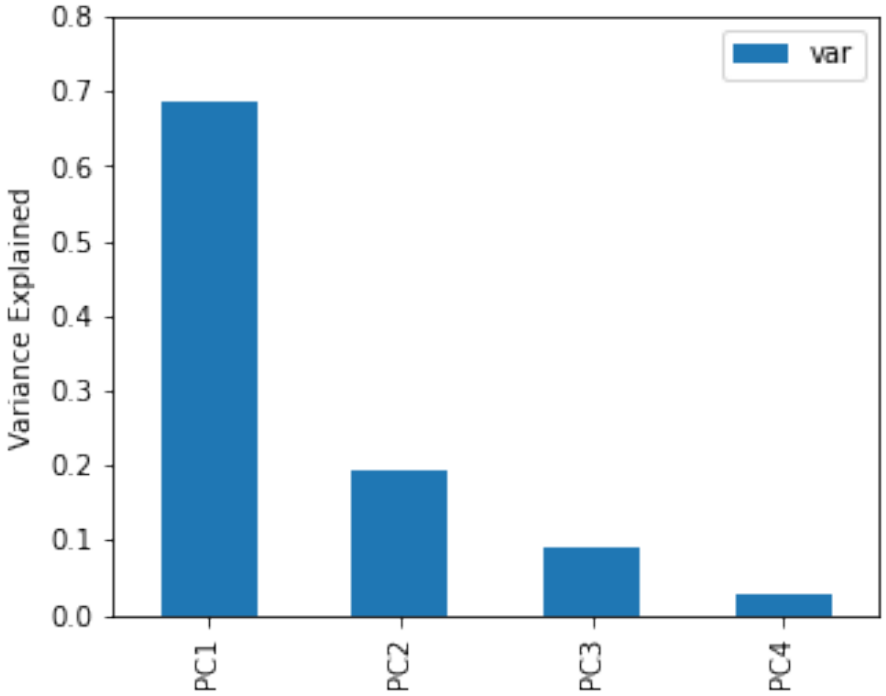
- bill length
- bill depth
- flipper length
- body mass

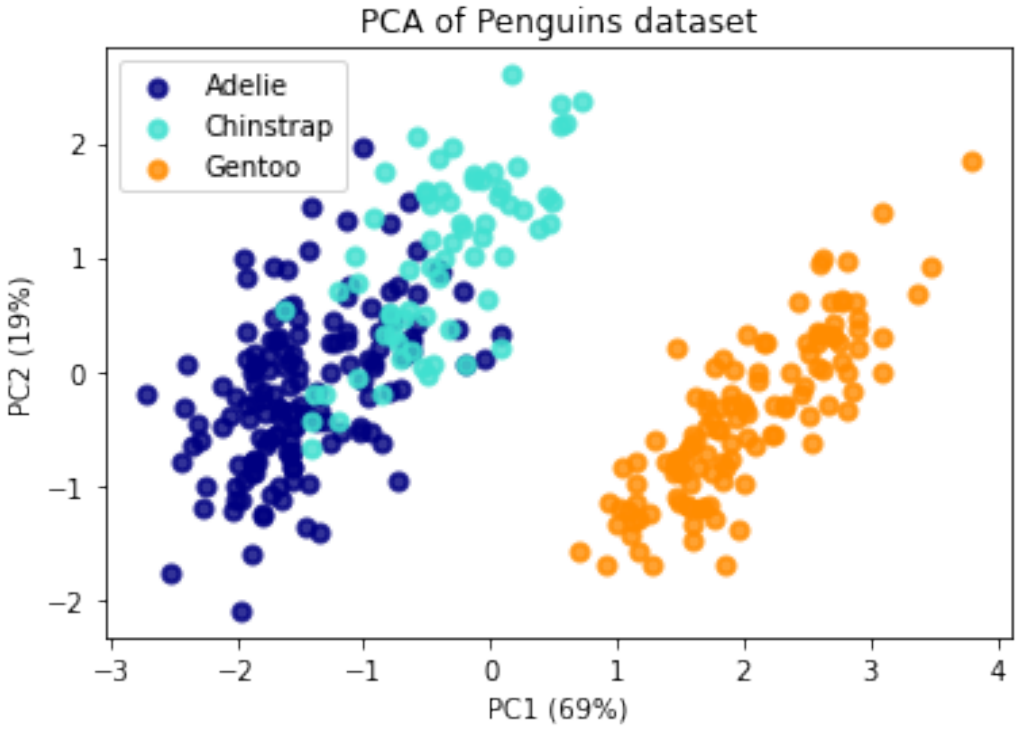
Applying PCA

We generally select the k PCs with the highest variance.

Example: Apply PCA to *Penguins* data set, and examine amount of variance in each PC.

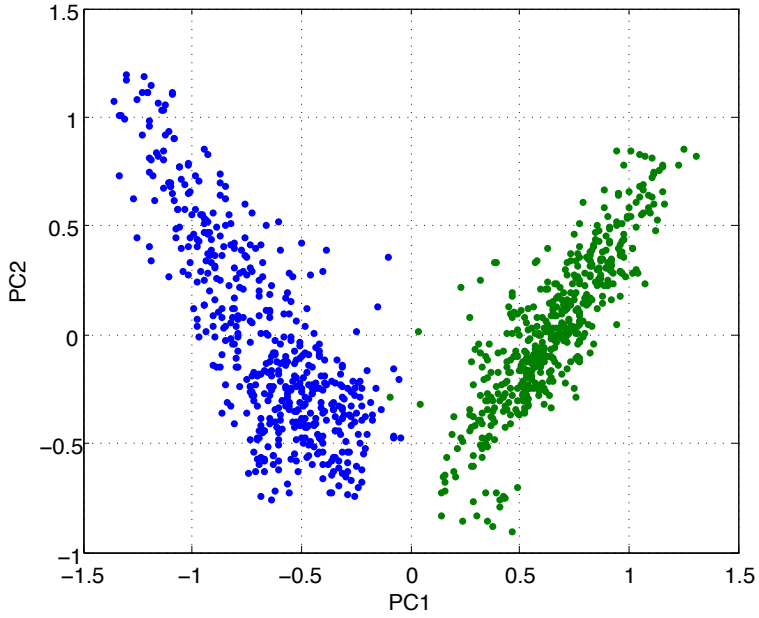
The first two PCs account for ~88% of the variance in the data.



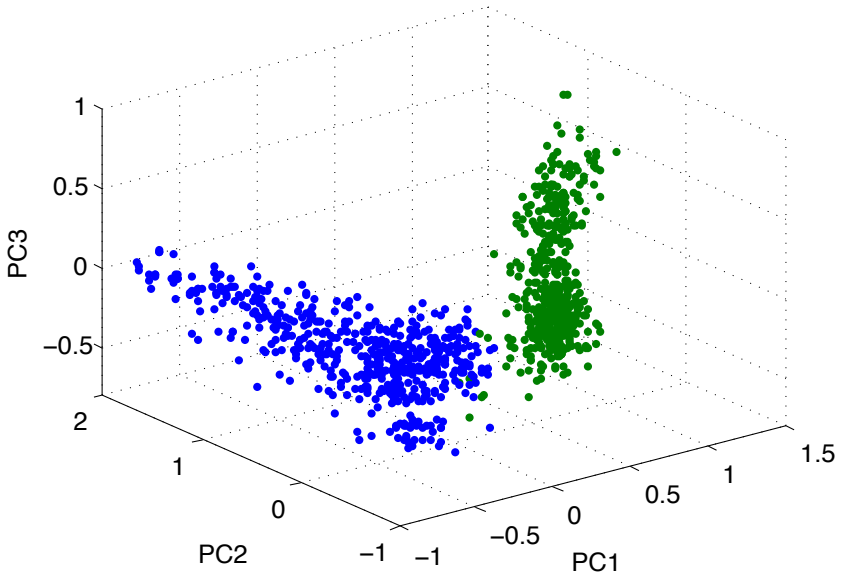


Example: PCA

- Collection of 1,021 BBC news articles on business + sport, represented by high-dimensional space with 5,570 features (words).
- Applying PCA allows us to visualise the data in a low dimensional space using a small number of PCs.



2 leading principal components (PCs)



3 leading principal components (PCs)

Python code notebook 10 PCA

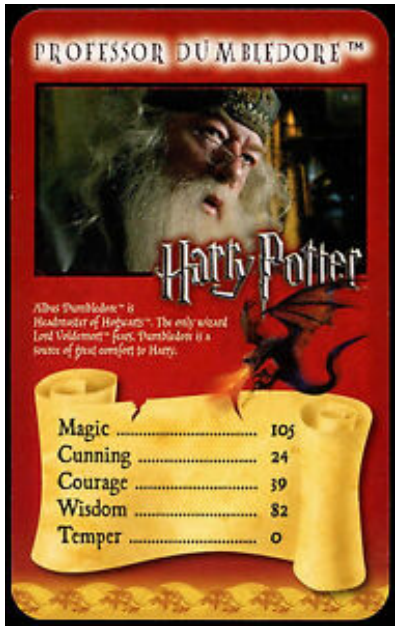
Scale the features to N(0,1)

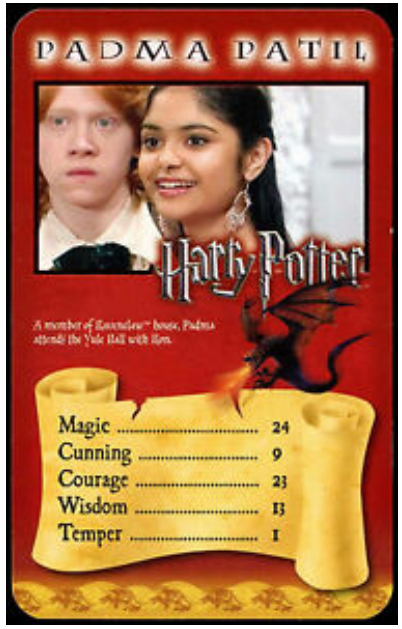
Fit the PCA and transform

pca object contains info on explained variance

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

X = penguins_all[['bill_length_mm', 'bill_depth_mm',
                  'flipper_length_mm', 'body_mass_g']]
y = penguins_all['species']

X_scal = StandardScaler().fit_transform(X)

pca = PCA(n_components=4)
X_r = pca.fit(X_scal).transform(X_scal)

# Proportion of variance explained for each components
pca.explained_variance_ratio_
Out[28]:
array([0.68633893, 0.19452929, 0.09216063, 0.02697115])
```

Harry Potter Top Trumps data

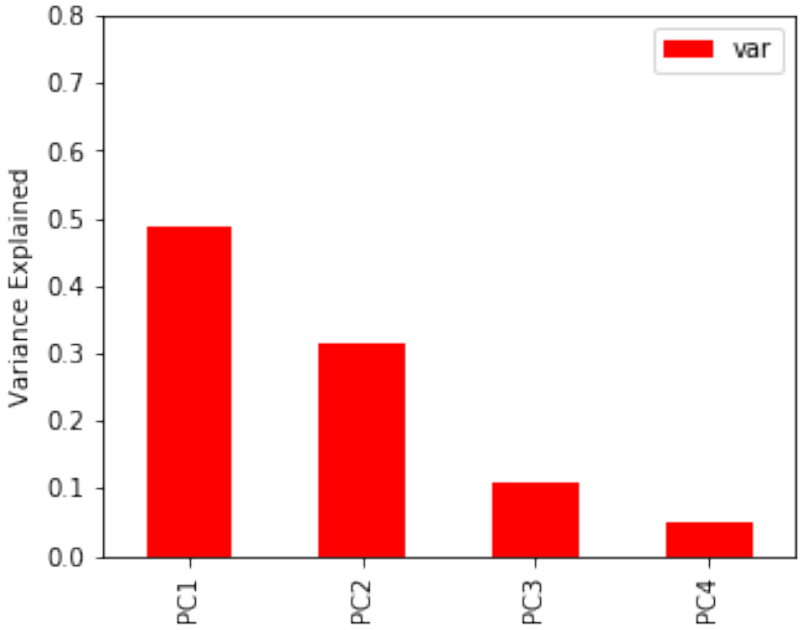




	Name	Magic	Cunning	Courage	Wisdom	Temper
0	Harry Potter'	62	21	42	26	7
1	Hermione Granger'	60	16	40	73	2
2	Ron Weasley'	45	14	40	22	4
3	Prof. Dumbledore'	105	24	39	82	0
4	Prof. Snape'	85	24	19	71	7
	...	...	...	...	...	...

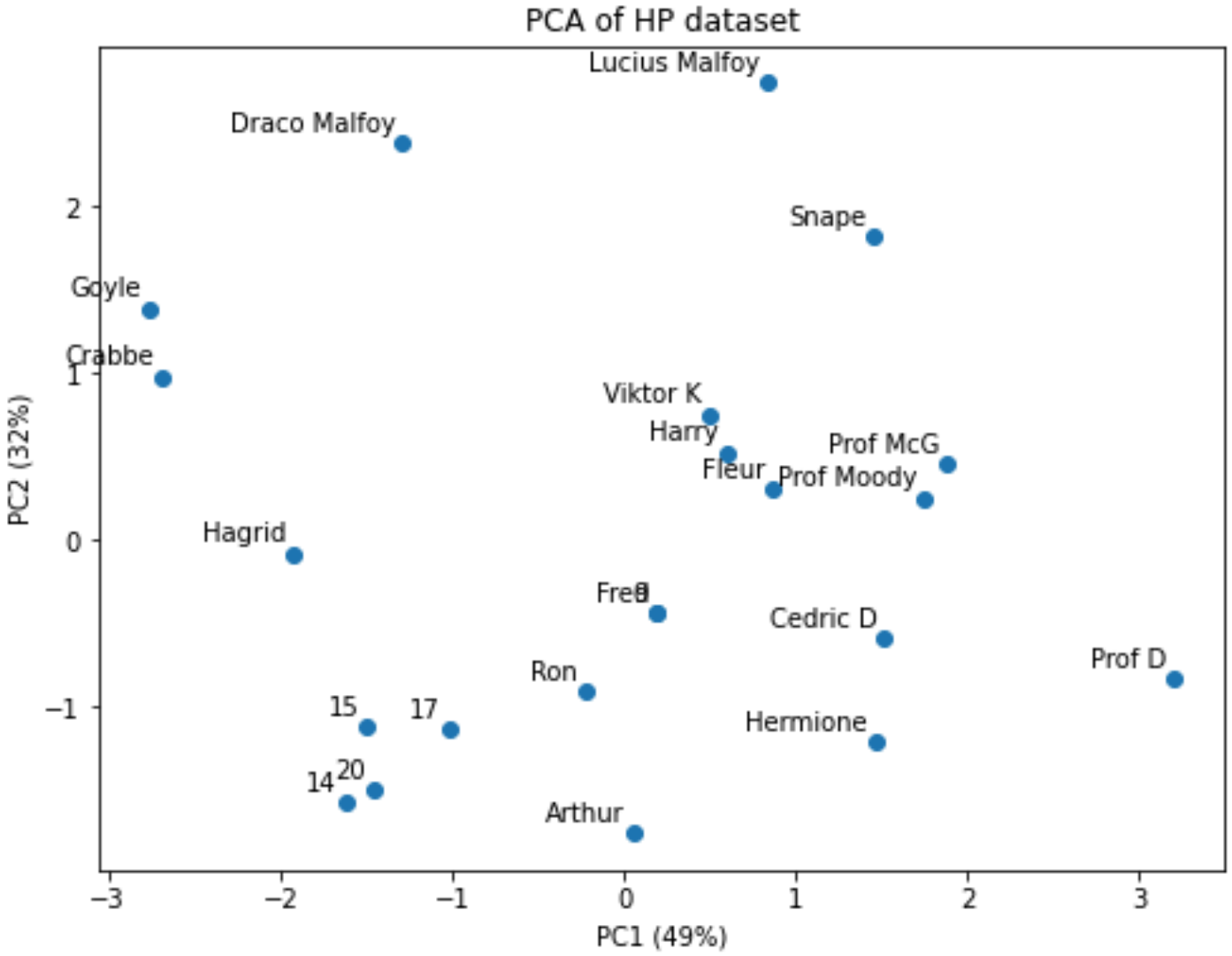
PCA on the Harry Potter data

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

TT_df = pd.read_csv('HarryPotterTT.csv')

y = TT_df.pop('Name').values
X = TT_df.values

X_scal = StandardScaler().fit_transform(X)

pcaHP = PCA(n_components=4)
X_r = pcaHP.fit(X_scal).transform(X)
pcaHP.explained_variance_ratio_
```



	Name	Magic	Cunning	Courage	Wisdom	Temper
0	Harry Potter'	62	21	42	26	7
1	Hermione Granger'	60	16	40	73	2
2	Ron Weasley'	45	14	40	22	4
3	Prof. Dumbledore'	105	24	39	82	0
4	Prof. Snape'	85	24	19	71	7
	...	...	...	...	...	...

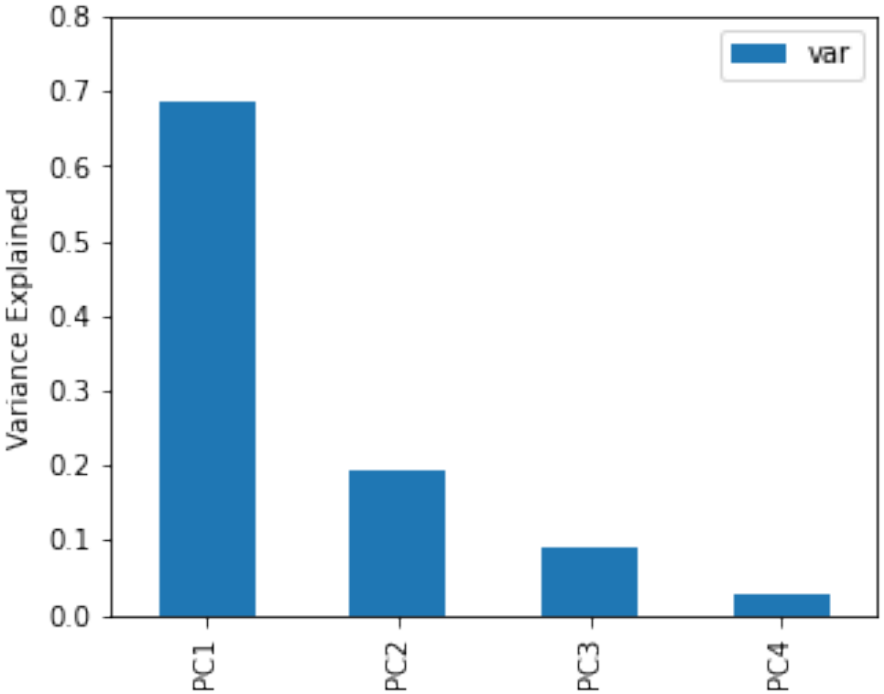
Harry Potter data - 1st two PCs



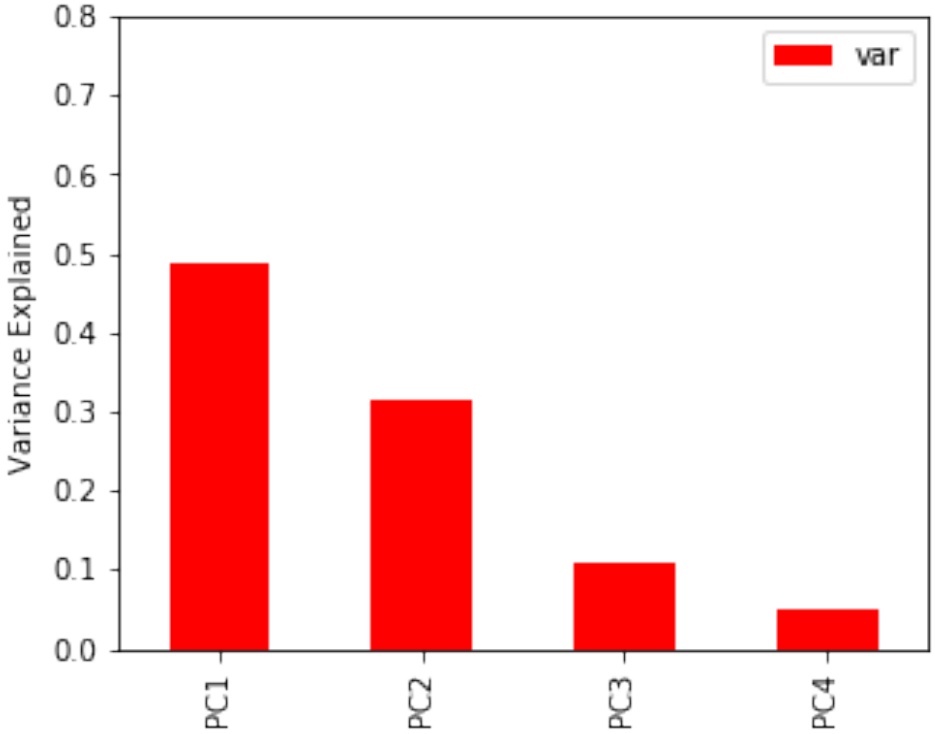
Variance Explained

- Compare inherent variance in both datasets

Penguin



Harry Potter



Summary

- The Curse of Dimensionality
- Dimension Reduction
- Feature Transformation v Selection
- Feature Selection in Supervised Learning
  - Filter Methods
  - Wrapper Methods
  - Embedded Methods
- Feature Transformation
  - Linear Transformations
  - Projection Methods
  - Principal Component Analysis (PCA)