Feature Selection

Topics Covered

- Importance of Feature Selection
- Filter Methods
- Wrapper Methods
- Embedded Methods
- Difference between Filter and Wrapper methods
- Feature selection examples

What is Feature Selection?

- Feature selection means selecting and retaining only the most important features in the model.

- Feature selection is different from feature extraction.

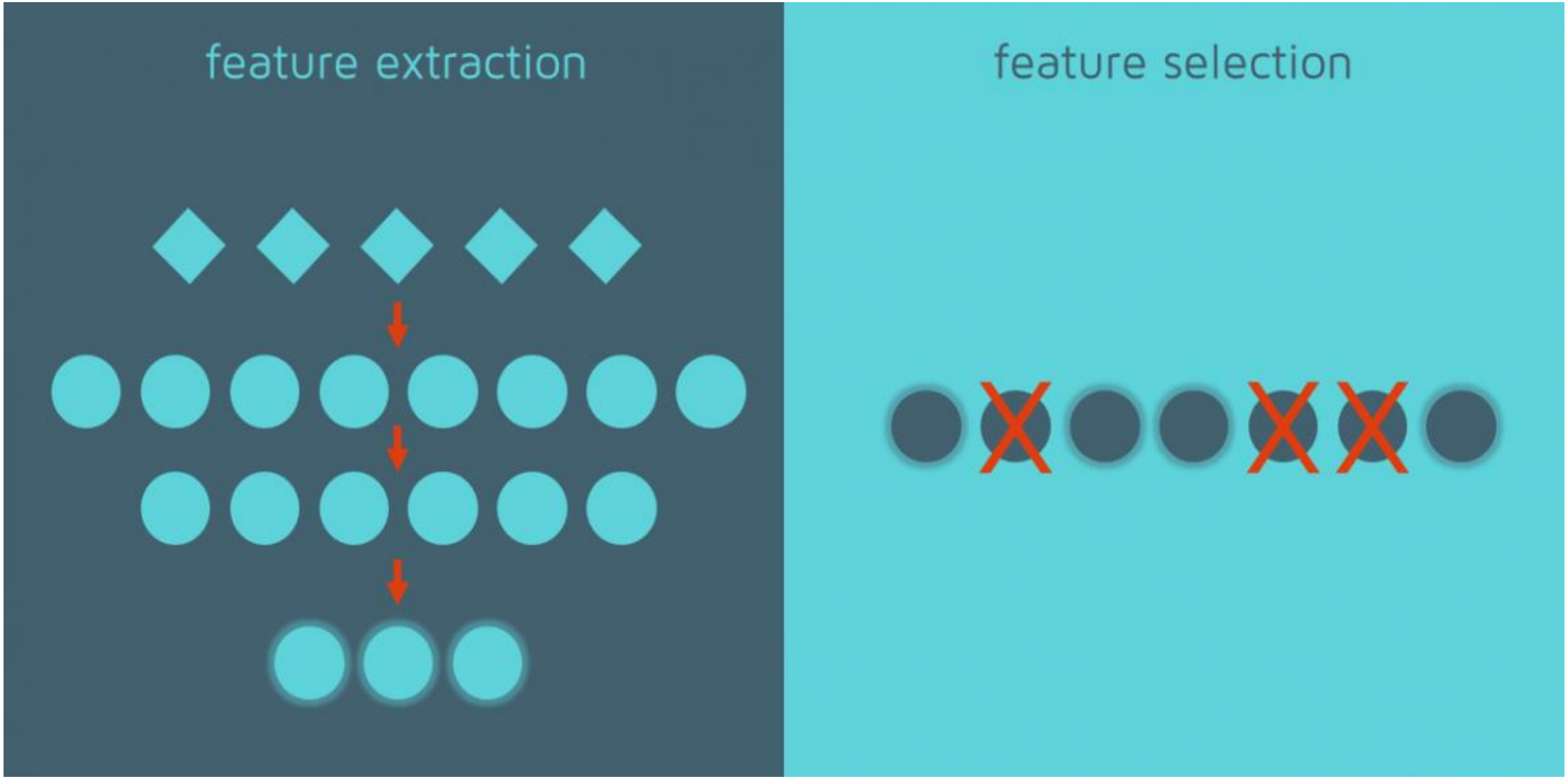
- **Extraction**: create a new feature from the existing features.
  - Example: **PCA** technique to reduce the dimensionality and eliminate redundancy

- **Selection**: choosing a subset of the original pool of features.

Feature selection

- Given a set of potential features, select some of them and discard the rest.
- Feature selection is applied either **to prevent redundancy and/or irrelevancy existing in the features** or just to get a limited number of features to prevent from overfitting.

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	1
15603246	Female	27	57000	0
15804002	Male	19	76000	1


feature extraction
feature selection

When should we apply feature extraction and selection?

Feature extraction:

- Feature extraction is always needed for ML models.
- We wouldn't need any feature extraction in deep learning neural networks (our algorithm can perform feature extraction by itself).

Feature selection:

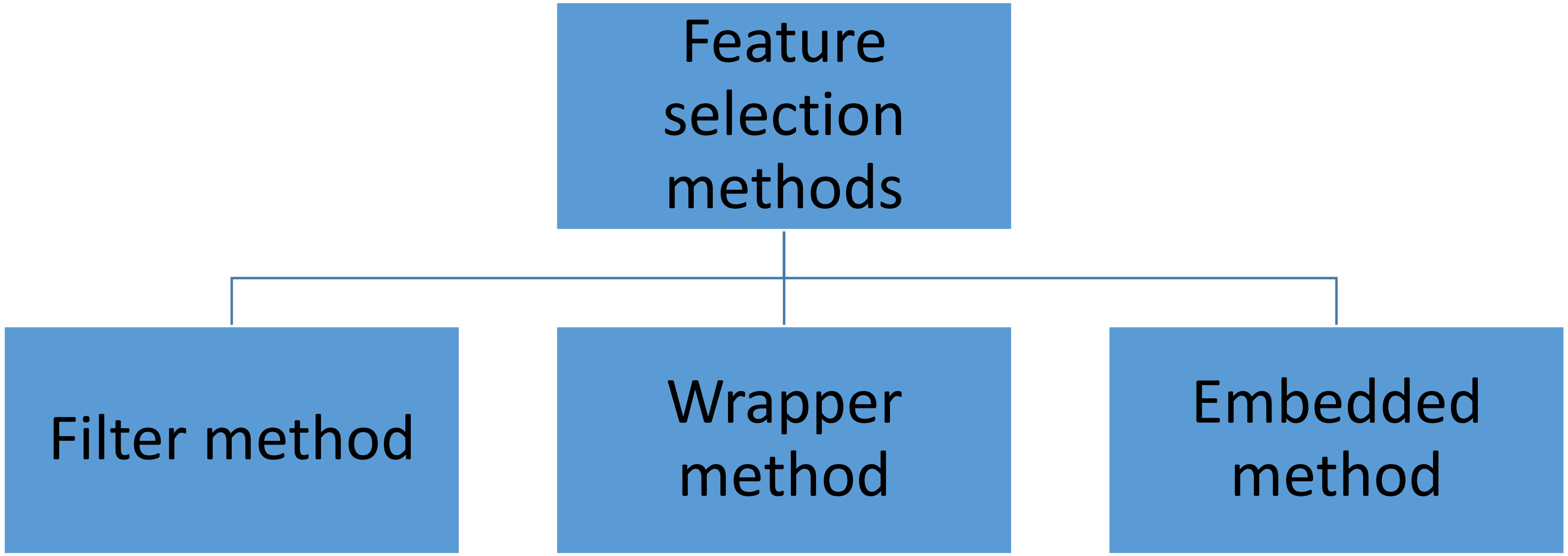
- Apply feature selection when there is a **redundancy or irrelevancy in dataset**, since these affect the model accuracy or simply add noise.
- Feature selection may be performed only to reduce the number of features, in order **to favor computing feasibility**.

Why Feature Selection is important?

1. Reduces the complexity of a model	6. Better visualization
2. Makes it easier to interpret	7. Reduces training time: It enables the machine learning algorithm to train faster
3. Data reduction	8. Reduces over-fitting
4. Less storage	9. Improves accuracy of the model, if the right subset is chosen
5. Fewest possible assumptions (Occam's razor)	10. Reduce the dimension

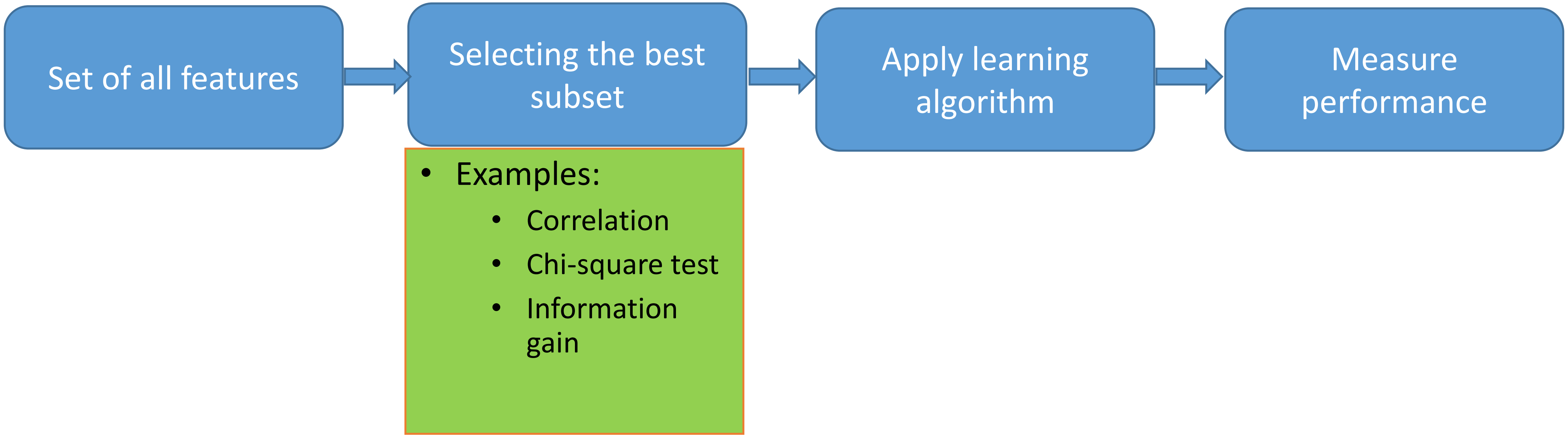
Feature Selection Methods

- Feature selection methods can be grouped into three categories:



Filter Methods

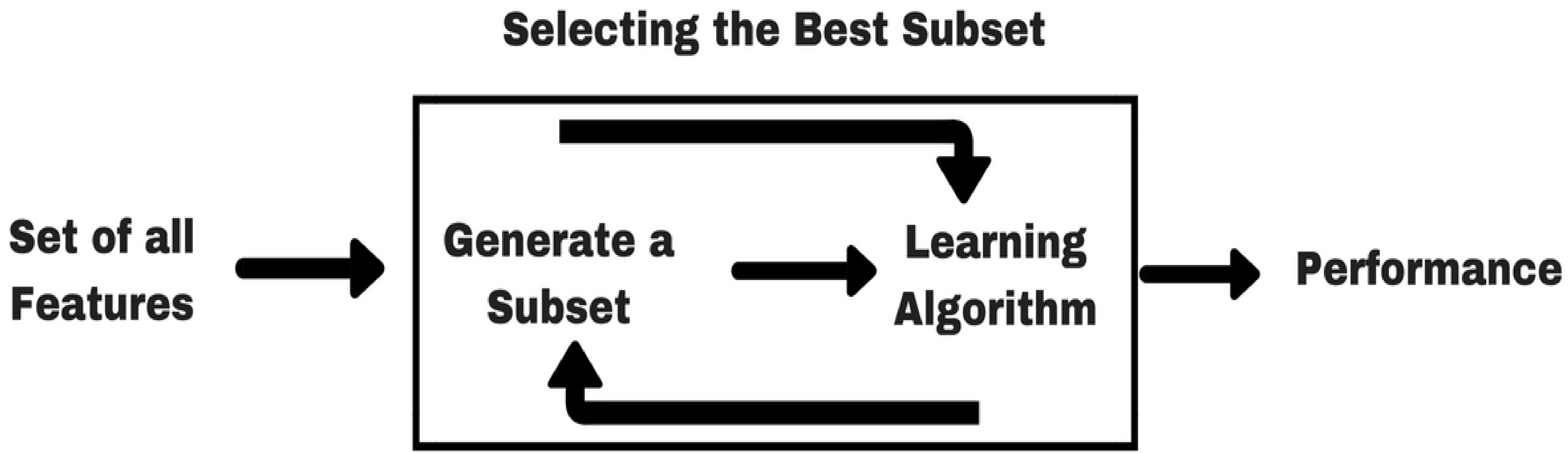
- Used as a preprocessing step.
- Selection of features is independent of any machine learning algorithms.
- Features are selected based on their scores in various statistical tests for their correlation with the outcome variable.



Wrapper Methods

- A wrapper evaluates a specific model sequentially using different subsets of features to get the best subset.

- They are **highly costly** and have a **high chance of overfitting**, but also a **high chance of success**, on the other hand.



Wrapper Methods

- These methods are usually computationally very expensive.

- Examples of wrapper methods are

  - Forward feature selection

  - Backward feature elimination

  - Recursive feature elimination etc.

Forward Selection

- Forward selection
  - Starts with no feature in the model.
  - In each iteration, we keep adding the feature which best improves our model till an addition of a new variable does not improve the performance of the model.

Backward Elimination

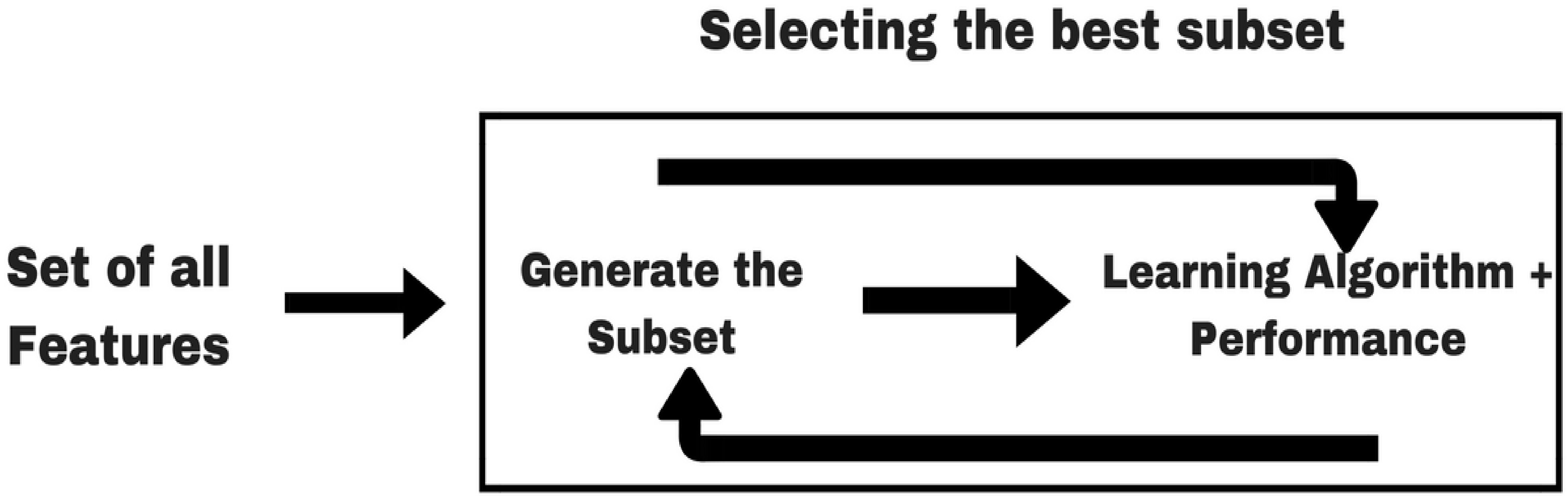
- Backward elimination
  - We start with all the features
  - Removes the least significant feature at each iteration which improves the performance of the model.
  - We repeat this until no improvement is observed on removal of features.

Recursive Feature elimination

- It is a greedy optimization algorithm which aims to find the best performing feature subset.
- It repeatedly creates models and keeps aside the best or the worst performing feature at each iteration.
- It constructs the next model with the left features until all the features are exhausted.
- It then ranks the features based on the order of their elimination.

Embedded Methods

- Embedded methods combine the qualities of filter and wrapper methods.
- It’s implemented by algorithms that have their own built-in feature selection methods.



Feature selection using scikit-learn

1. Dropping features which have low variance

2. Univariate feature selection

3. Model based feature selection

4. Feature selection using pipeline

Feature selection using scikit-learn

1. Dropping features which have low variance

1. Dropping features with zero variance
2. Dropping features with variance below the threshold variance

1. Dropping features with zero variance

- Variance threshold option drops two features with zero variance.

- Default variance threshold is zero.

```
VarianceThreshold()
```

```
VarianceThreshold(threshold=0.0)
```

offer	Age	online payment	items
1	35	0	5
1	26	0	4
1	41	0	9
1	34	0	10
1	38	0	3

Two features have zero variance

Dropping features with Zero variance

```
In [55]: from sklearn.feature_selection import VarianceThreshold
         selector = VarianceThreshold(threshold = 0.0)
         selector.fit_transform(dataset)

Out[55]: array([[35,  5],
                [26,  4],
                [41,  9],
                [34, 10],
                [38,  3]], dtype=int64)
```

offer	Age	online payment	items
1	35	0	5
1	26	0	4
1	41	0	9
1	34	0	10
1	38	0	3

Dropping features with low variance (less than threshold)

In [56]:
```
# Importing the dataset
dataset = pd.read_csv('data-2.csv')
dataset
```

Out[56]:

	referred	repeat	Age	Promoted	items
0	1	0	35	1	5
1	0	0	26	0	4
2	1	0	41	1	9
3	1	1	34	0	10
4	1	0	38	1	3
5	1	0	40	0	7

referred and repeat have low variance

Dropping features with low variance (less than threshold)

- If we want to drop a feature which contains only 0’s 80% of the time or only 1s 80% of the time.

- Then the variance of the feature would be 0.8 * (1-0.8) = 0.16

```
VarianceThreshold(threshold=0.16)
```

```
VarianceThreshold(threshold=(0.8 * (1-0.8)))
```

In [57]:
```python
from sklearn.feature_selection import VarianceThreshold
selector = VarianceThreshold(threshold = 0.16)
selector.fit_transform(dataset)
```

Out[57]:
```
array([[35,  1,  5],
       [26,  0,  4],
       [41,  1,  9],
       [34,  0, 10],
       [38,  1,  3],
       [40,  0,  7]], dtype=int64)
```

Two features referred and repeat have low variance.
Either only 1s or 0s for 80% of the time is dropped

2. Univariate feature selection

- The Iris Dataset contains four features
  - (length and width of sepals and petals) of 50 samples of three species of Iris (Iris setosa, Iris virginica and Iris versicolor).
  - These measures were used to create a linear discriminant model to classify the species.

Iris Setosa:
Length of sepals: 5.1
Width of sepals: 3.5

Length of petals: 1.4
Width of petals : 0.2

5.1,3.5,1.4,0.2,Iris-setosa

Univariate Feature Selection

In [7]:
```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.datasets import load_iris
```

Original dataset contains 4 predictors

In [8]:
```
iris = load_iris()
X, y = iris.data, iris.target
X.shape
```

Out[8]: (150, 4)

Iris features:
Length of sepals
Width of sepals

Length of petals
Width of petals

In [9]:
```
#Select 3 best features
X_select = SelectKBest(chi2, k=3).fit_transform(X, y)
X_select.shape
```

Out[9]: (150, 3)

Best 3 predictors are retained based on chi-square value

Model Based Feature Selection

In [62]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
from sklearn.feature_selection import SelectFromModel
iris = load_iris()
X, y = iris.data, iris.target
X.shape
```

Out[62]:
```
(150, 4)
```

Estimate feature importance using **RandomForest** model to select features.

In [64]:
```
clf = RandomForestClassifier(n_estimators=10)
clf = clf.fit(X, y)
clf.feature_importances_
```

Out[64]:
```
array([0.24504133, 0.04692402, 0.39381251, 0.31422214])
```

n_estimators – number of decision trees in the forest.

model = SelectFromModel(clf, prefit = True)

```
In [64]: clf = RandomForestClassifier(n_estimators=10)
         clf = clf.fit(X, y)
         clf.feature_importances_

Out[64]: array([0.24504133, 0.04692402, 0.39381251, 0.31422214])
```

```
model = SelectFromModel(clf, prefit = True)
```

SelectFromModel()

- Threshold value – mean of all the features. (0.2450+0.0469+0.3938+0.31422/4) = 0.25
- Features whose importance is greater or equal to 0.25 are kept while the others are discarded.

In [65]:
```python
model = SelectFromModel(clf, prefit = True)
X_new = model.transform(X)
X_new.shape
```

Out[65]:
```
(150, 2)
```

Out of 4 only 2 features have been retained

X
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
```

X_new
```
array([[1.4, 0.2],
       [1.4, 0.2],
       [1.3, 0.2],
       [1.5, 0.2],
```

```
In [17]: X

Out[17]: array([[5.1, 3.5, 1.4, 0.2],
                [4.9, 3. , 1.4, 0.2],
                [4.7, 3.2, 1.3, 0.2],
                [4.6, 3.1, 1.5, 0.2],
                [5. , 3.6, 1.4, 0.2],
                [5.4, 3.9, 1.7, 0.4],
                [4.6, 3.4, 1.4, 0.3],
                [5. , 3.4, 1.5, 0.2],
                [4.4, 2.9, 1.4, 0.2],
                [4.9, 3.1, 1.5, 0.1],
                [5.4, 3.7, 1.5, 0.2],
                [4.8, 3.4, 1.6, 0.2],
                [4.8, 3. , 1.4, 0.1],
                [4.3, 3. , 1.1, 0.1],
                [5.8, 4. , 1.2, 0.2],
                [5.7, 4.4, 1.5, 0.4],
                [5.4, 3.9, 1.3, 0.4],
                [5.1, 3.5, 1.4, 0.3],
```

```
In [18]: X_new

Out[18]: array([[1.4, 0.2],
                [1.4, 0.2],
                [1.3, 0.2],
                [1.5, 0.2],
                [1.4, 0.2],
                [1.7, 0.4],
                [1.4, 0.3],
                [1.5, 0.2],
                [1.4, 0.2],
                [1.5, 0.1],
                [1.5, 0.2],
                [1.6, 0.2],
                [1.4, 0.1],
                [1.1, 0.1],
                [1.2, 0.2],
                [1.5, 0.4],
                [1.3, 0.4],
```

Feature Selection Using Pipeline

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
clfs = Pipeline([('feature_selection',
                  SelectFromModel(RandomForestClassifier(n_estimators=10))),
                 ('classification', KNeighborsClassifier())])
model = clfs.fit(X, y)
```

Pipeline process:

1. First features are selected
2. Model is built using selected features

Thank you