Feature Selection

Feature Reduction in ML

- The information about the target class is **inherent in the variables**.
- Naïve view:

  More features

  => More information

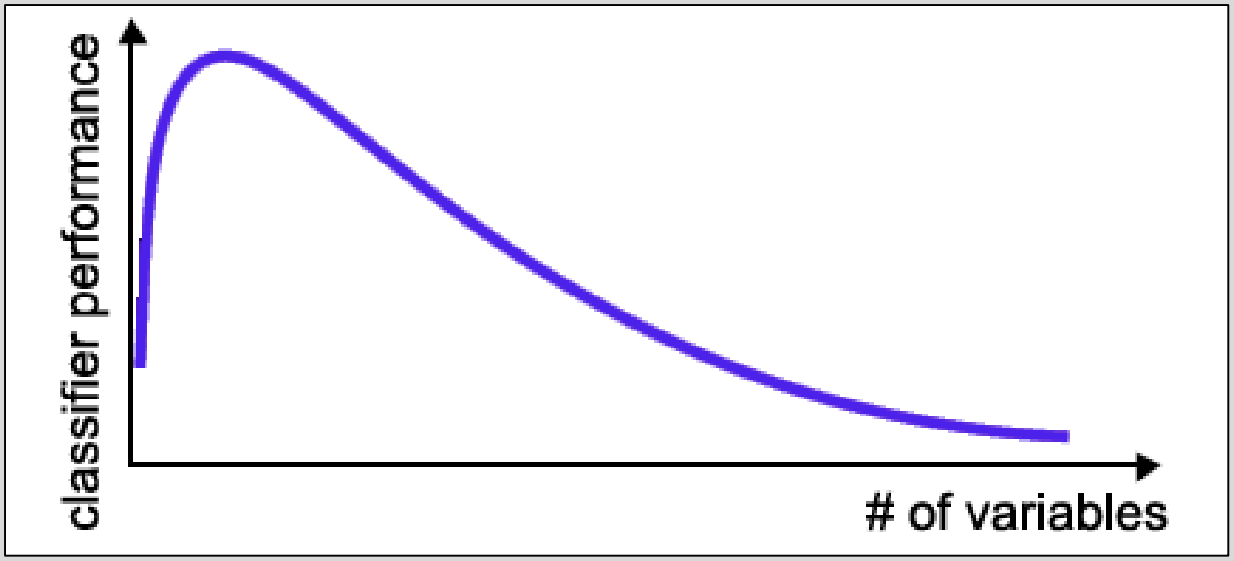
  => More discrimination power.
- In practice:

  many reasons why this is not the case!

Curse of Dimensionality

- number of training examples is fixed

  => the classifier's performance usually will degrade for a large number of features!



Feature Reduction in ML

- Irrelevant and
- redundant features
  - can confuse learners.

- Limited training data.
- Limited computational resources.
- **Curse of dimensionality**.

Feature Selection

Problem of selecting some subset of features, while ignoring the rest

Feature Extraction

- Project the original x_i , i =1,...,d dimensions to new $k < d$ dimensions, z_j , j =1,...,k

Criteria for selection/extraction:
either improve or maintain the classification accuracy, simplify classifier complexity.

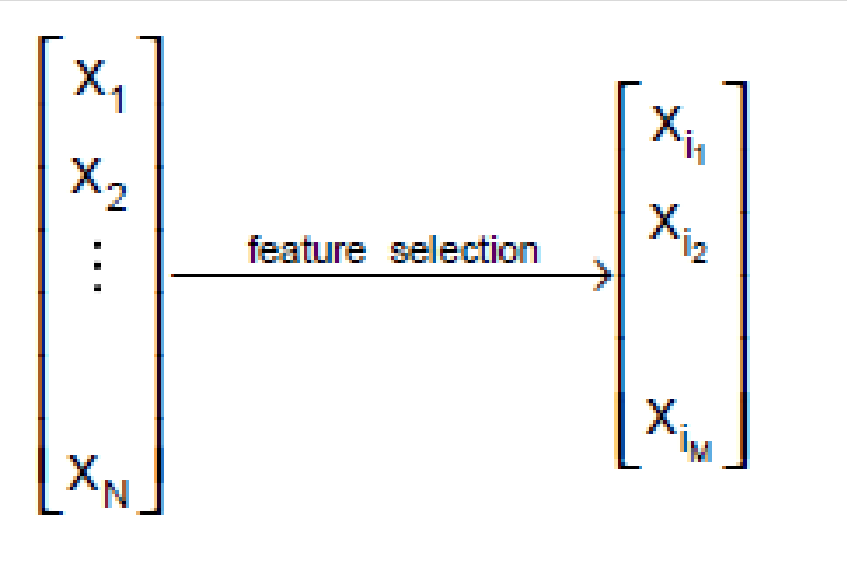
Feature Selection - Definition

- Given a set of features $F = \{x_1, \ldots, x_n\}$

  the Feature Selection problem is

  to find a subset $F' \subseteq F$ that maximizes the learners ability to classify patterns.

- Formally F' should maximize some scoring function



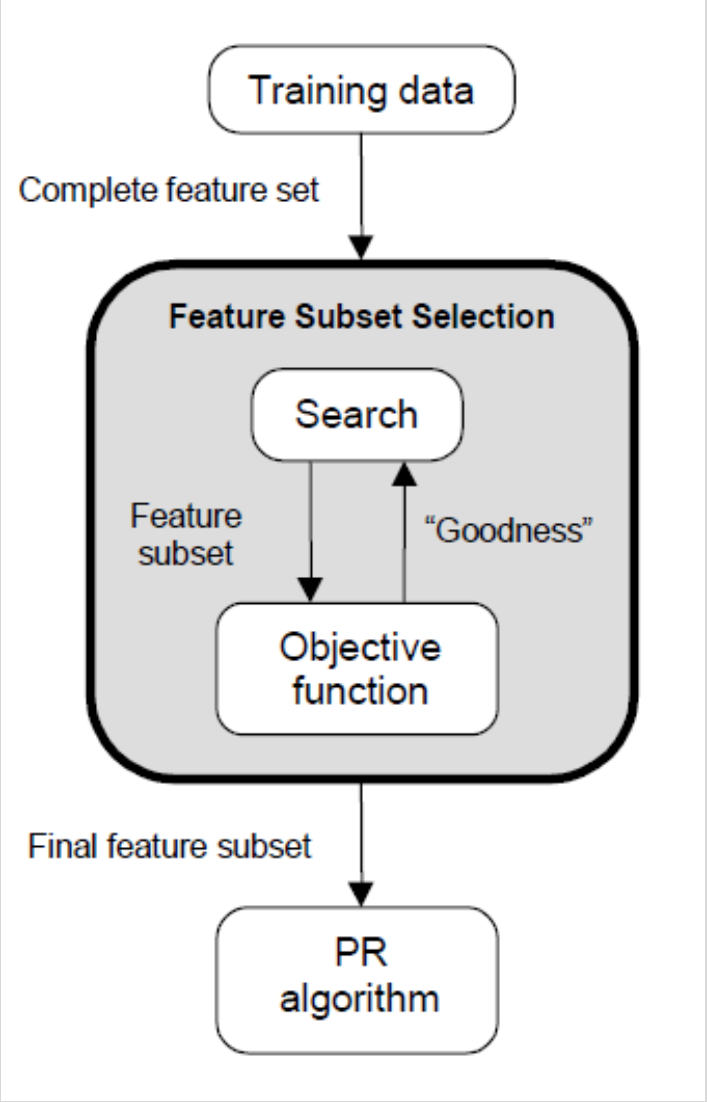
Subset selection

- d initial features
- There are 2^d possible subsets
- Criteria to decide which subset is the best:
  - classifier based on these m features has the lowest probability of error of all such classifiers
- Can't go over all 2^d possibilities
- Need some heuristics

Feature Selection Steps

Feature selection is an **optimization** problem.

- Step 1: Search the space of possible feature subsets.
- Step 2: Pick the subset that is optimal or near-optimal with respect to some objective function.



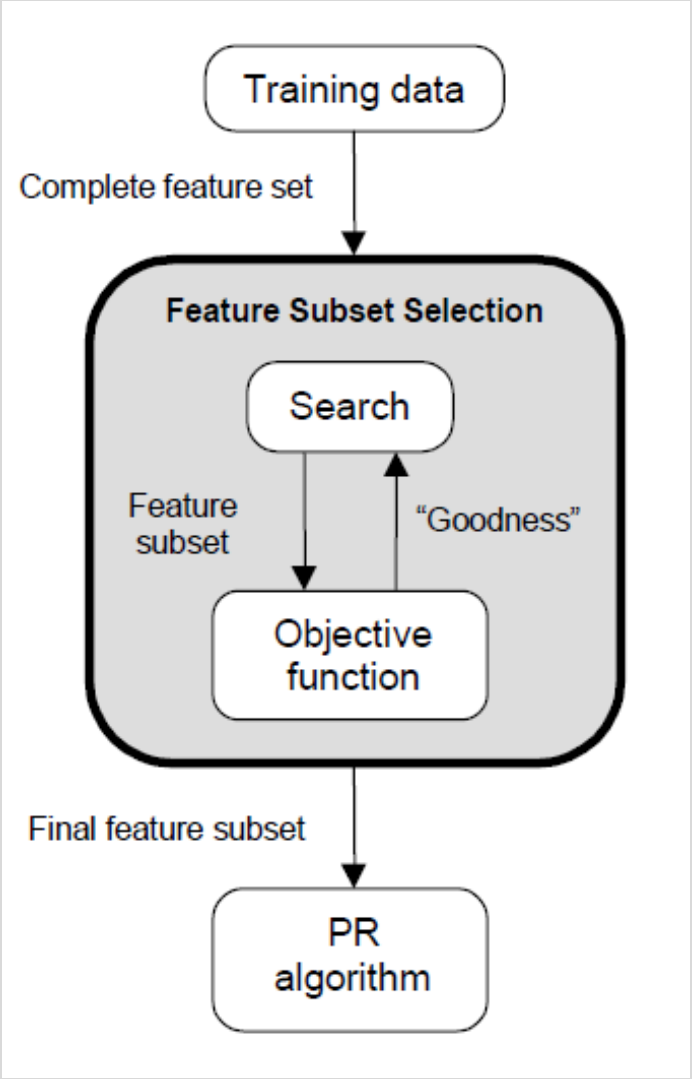
Feature Selection Steps (cont'd)

Search strategies

- Optimum
- Heuristic
- Randomized

Evaluation strategies

- Filter methods
- Wrapper methods

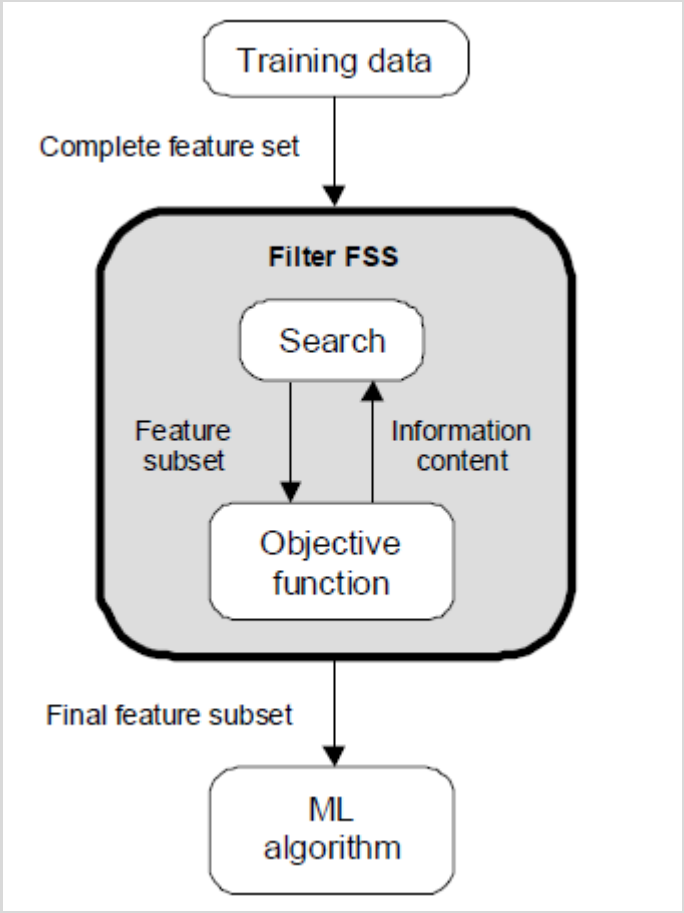


Evaluating feature subset

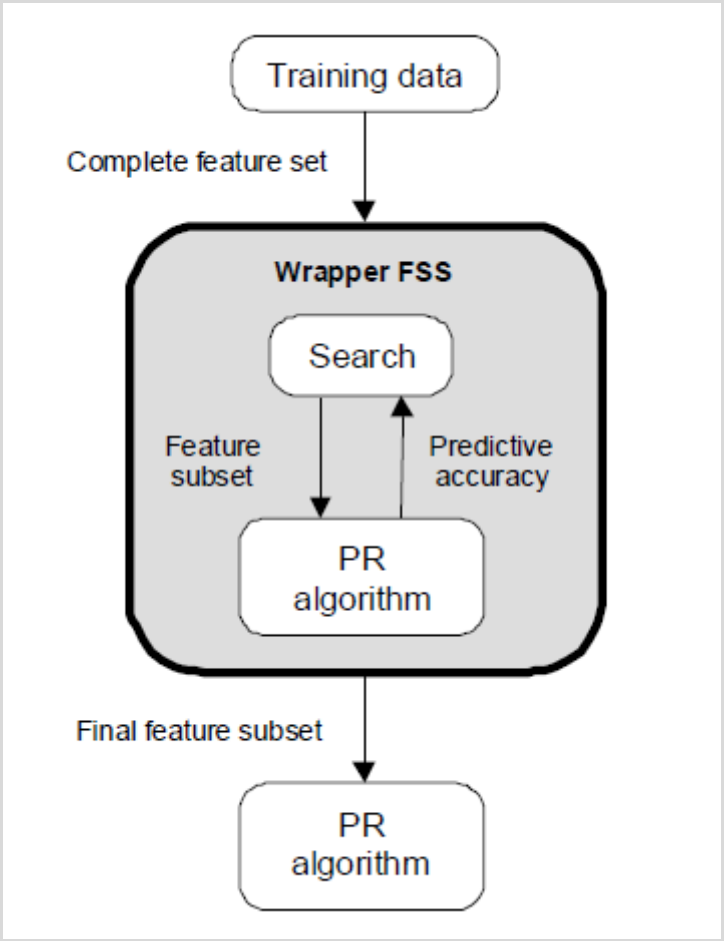
- Supervised (wrapper method)
  - Train using selected subset
  - Estimate error on validation dataset
- Unsupervised (filter method)
  - Look at input only
  - Select the subset that has the most information

Evaluation Strategies

Filter Methods



Wrapper Methods



Filter Methods

- while doing the pre-processing step
- very fast and inexpensive and are very good for removing duplicated, correlated, redundant features

Techniques used:

- ❑ **Information Gain**
- ❑ **Chi-square test**
- ❑ **Fisher's Score**
- ❑ **Correlation Coefficient**
- ❑ **ANOVA (Analysis of Variance)**
- ❑ **Mean Absolute Difference (MAD)**
- ❑ **Dispersion Ratio**
- ❑ **Mutual Dependence**
- ❑ **Relief**

Wrapper methods

- greedy algorithms train the algorithm by using a subset of features in an iterative manner
- Based on the conclusions made from training in prior to the model, addition and removal of features takes place
- provide an optimal set of features for training the model
- better accuracy than the filter methods but are computationally more expensive

Techniques used:

- ❑ **Forward selection**
- ❑ **Backward elimination**
- ❑ **Bi-directional elimination**
- ❑ **Exhaustive selection**
- ❑ **Recursive elimination**

Filter Methods

- **Correlation-based Feature Selection**: This method evaluates the correlation between each feature and the target variable. Features with high correlation coefficients are considered more relevant and retained, while those with low correlation are discarded.
- **Mutual Information**: Mutual information measures the amount of information shared between two variables. In feature selection, it assesses the dependency between each feature and the target variable. Features with high mutual information scores are selected.
- **Chi-square Test**: Chi-square test evaluates the independence between categorical features and the target variable. It computes the statistical significance of the association between each feature and the target. Features with high chi-square scores are retained.
- **Information Gain**: Information gain, commonly used in decision trees, quantifies the reduction in uncertainty about the target variable provided by a given feature. Features with high information gain are considered more informative and selected for further analysis.

Filter Methods

- **ANOVA (Analysis of Variance)**: ANOVA assesses the variation in the target variable explained by each feature in a regression or classification task. It computes the F-statistic to determine the significance of feature contributions. Features with high F-statistic values are retained.
- **Univariate Feature Selection**: Univariate feature selection methods evaluate each feature individually with respect to the target variable. They apply statistical tests or scoring functions to rank features and select the top-ranked ones based on predefined criteria.

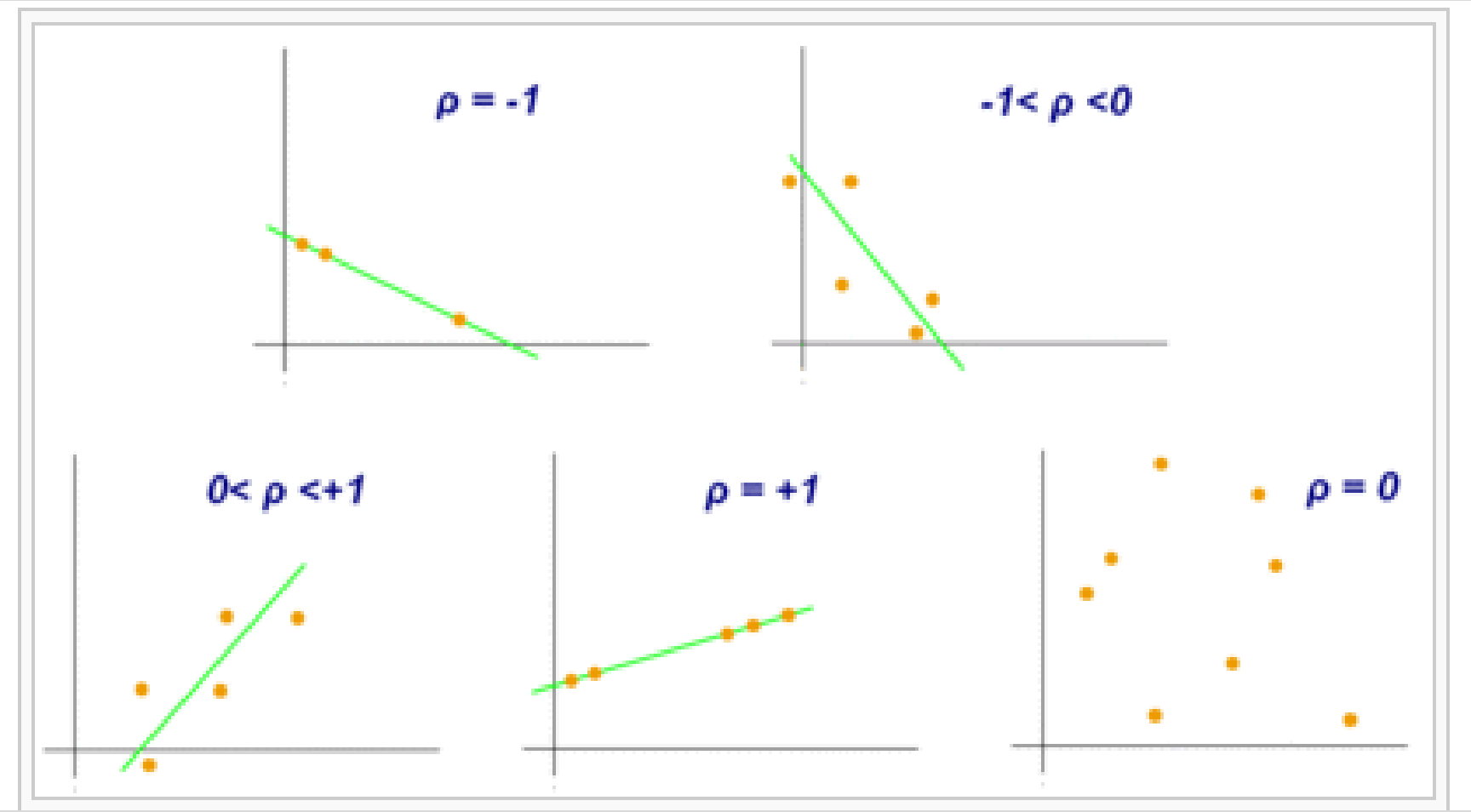
Pearson correlation coefficient

- Measures the correlation between two variables
- Formula for Pearson correlation =

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

- The correlation r is between $+1$ and -1.
  - $+1$ means perfect positive correlation
  - -1 in the other direction

Pearson correlation coefficient



From Wikipedia

Example

Student	Hours of Study	Exam Score
1	2	75
2	3	80
3	4	85
4	5	90
5	6	95

1. Compute Pearson's Correlation Coefficients:

Answer

1. **Calculate Means**: Calculate the means) for Hours of Study and Exam Score.
2. **Calculate Deviations from Mean**: Calculate the deviations from the mean for both features.
3. **Calculate Sum of Squares of Deviations**: Calculate the sum of squares of deviations for both features.
4. **Calculate Pearson's Correlation Coefficient**: Use the formula to calculate Pearson's correlation coefficient.

Calculate Means:

$$\bar{x} = \frac{2+3+4+5+6}{5} = \frac{20}{5} = 4$$

$$\bar{y} = \frac{75+80+85+90+95}{5} = \frac{425}{5} = 85$$

Calculate Deviations from Mean:

Hours of Study	Exam Score	$(x_i - \bar{x})$	$(y_i - \bar{y})$
2	75	-2	-10
3	80	-1	-5
4	85	0	0
5	90	1	5
6	95	2	10

Calculate Sum of Squares of Deviations:

$$\sum (x_i - \bar{x})^2 = (-2)^2 + (-1)^2 + 0^2 + 1^2 + 2^2 = 10$$
$$\sum (y_i - \bar{y})^2 = (-10)^2 + (-5)^2 + 0^2 + 5^2 + 10^2 = 250$$

Calculate Pearson's Correlation Coefficient:

$$\text{Pearson's Correlation Coefficient} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \cdot \sum (y_i - \bar{y})^2}}$$

$$\text{Pearson's Correlation Coefficient} = \frac{(-2 \times -10) + (-1 \times -5) + (0 \times 0) + (1 \times 5) + (2 \times 10)}{\sqrt{10 \times 250}} = \frac{50}{\sqrt{2500}} \approx \frac{50}{50} = 1$$

So, the Pearson's correlation coefficient between Hours of Study and Exam Score is 1, indicating a perfect positive linear relationship. This suggests that as the number of hours of study increases, the exam score also increases linearly. It can be considered as a relevant feature for predicting the Exam Score.

$$\text{Age} = [20, 25, 30, 35, 40]$$

$$\bar{x} = \frac{20 + 25 + 30 + 35 + 40}{5} = \frac{150}{5} = 30$$

Calculate the deviations from the mean for Age:

Age	$(x_i - \bar{x})$
20	-10
25	-5
30	0
35	5
40	10

square these deviations:

$$\begin{aligned}\sum (x_i - \bar{x})^2 &= (-10)^2 + (-5)^2 + 0^2 + 5^2 + 10^2 \\ &= 100 + 25 + 0 + 25 + 100 \\ &= 250\end{aligned}$$

```
import numpy as np

# Define the data
hours_of_study = np.array([2, 3, 4, 5, 6])
exam_score = np.array([75, 80, 85, 90, 95])

# Calculate Pearson's correlation coefficient
correlation_coefficient = np.corrcoef(hours_of_study, exam_score)[0, 1]

print("Pearson's Correlation Coefficient:", correlation_coefficient)
```

Wrapper methods

- **Forward Selection**: Forward selection starts with an empty set of features and iteratively adds one feature at a time, selecting the feature that improves the model performance the most until a stopping criterion is met. This process continues until no further improvement is observed or a predefined number of features is reached.
- **Backward Elimination**: Backward elimination begins with all features included and removes one feature at a time, selecting the feature whose removal improves the model performance the most. This process continues until no further improvement is observed or a predefined number of features remains.
- **Recursive Feature Elimination (RFE)**: RFE is a variant of backward elimination. It recursively removes the least important features based on their weights or coefficients, typically using the feature ranking provided by a machine learning algorithm. RFE continues removing features until the desired number of features is reached or until the model performance deteriorates beyond a certain threshold.
- **Bidirectional Elimination (Bidirectional Search)**: Bidirectional elimination combines forward selection and backward elimination. It starts with an empty set of features and iteratively adds and removes features based on their individual contributions to the model performance. Bidirectional elimination can be more computationally expensive compared to forward selection or backward elimination but may result in a more optimal feature subset.

Forward Selection Example

Sample	Feature 1	Feature 2	Feature 3	Target
1	1	0	1	0
2	2	1	0	1
3	0	1	1	1
4	1	1	1	1
5	2	0	0	0

1. Start with an empty set of selected features.
2. Train a model using each individual feature and select the one that results in the best performance according to a chosen evaluation metric.
3. Add the selected feature to the set of selected features.
4. Repeat steps 2 and 3 until a stopping criterion is met (e.g., a maximum number of features or no further improvement in performance).

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Define the dataset
X = np.array([[1, 0, 1],
              [2, 1, 0],
              [0, 1, 1],
              [1, 1, 1],
              [2, 0, 0]])
y = np.array([0, 1, 1, 1, 0])

# Initialize an empty set of selected features
selected_features = []

# Define the maximum number of features to select
max_features = 3

# Perform forward selection
for _ in range(max_features):
    best_feature_idx = -1
    best_accuracy = 0
```

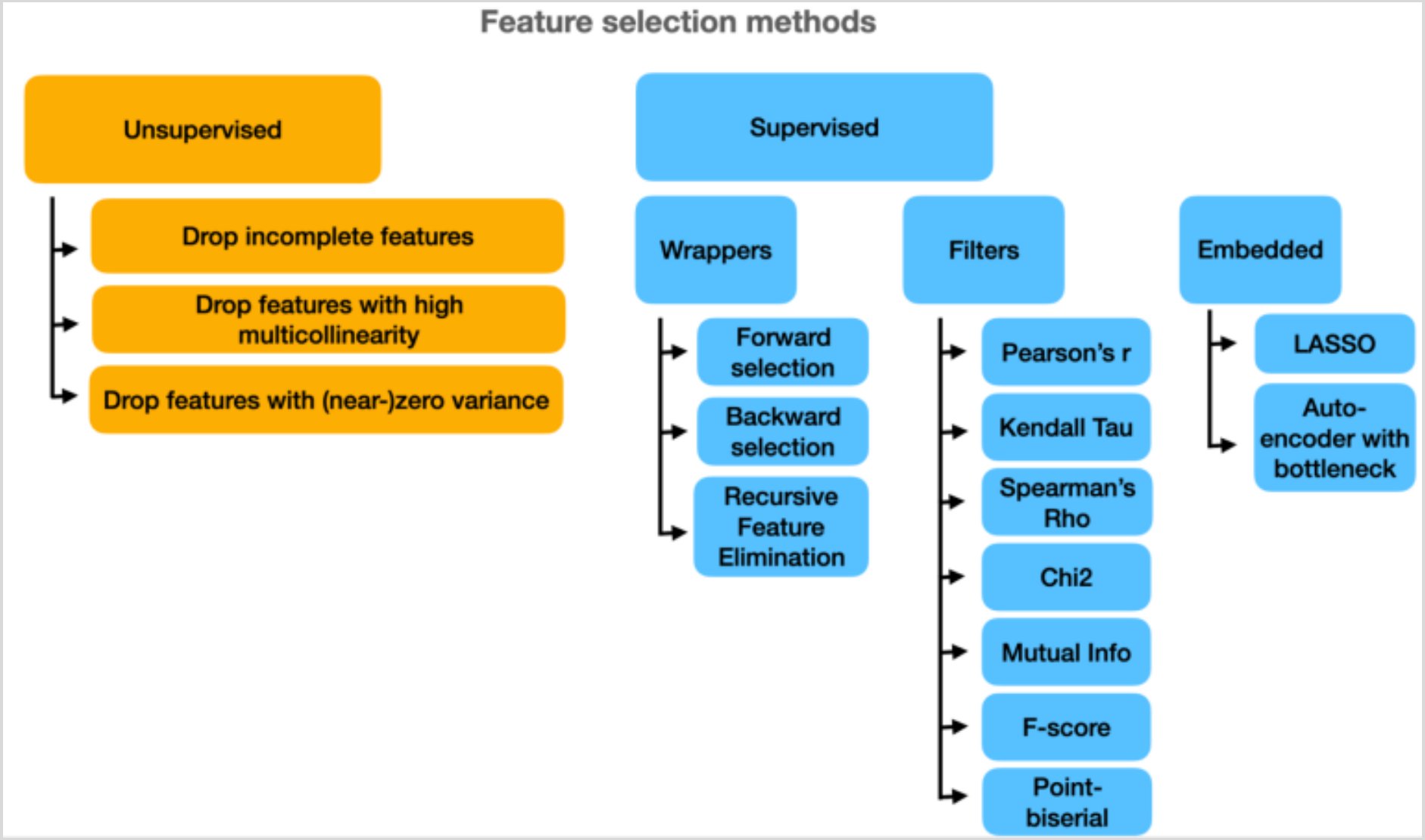
```
 # Train a model using each individual feature
   for feature_idx in range(X.shape[1]):
      if feature_idx not in selected_features:
         # Select the feature
         features = selected_features + [feature_idx]
         X_selected = X[:, features]

         # Train a logistic regression model
         model = LogisticRegression()
         model.fit(X_selected, y)

         # Make predictions
         y_pred = model.predict(X_selected)

         # Calculate accuracy
         accuracy = accuracy_score(y, y_pred)

         # Check if this feature yields the best performance so far
         if accuracy > best_accuracy:
            best_accuracy = accuracy
            best_feature_idx = feature_idx
      # Add the best feature to the set of selected features
   selected_features.append(best_feature_idx)
   print("Selected Feature:", best_feature_idx)
print("Selected Features:", selected_features)
```


Feature selection methods
Unsupervised
Drop incomplete features
Drop features with high multicollinearity
Drop features with (near-)zero variance
Supervised
Wrappers
Forward selection
Backward selection
Recursive Feature Elimination
Filters
Pearson's r
Kendall Tau
Spearman's Rho
Chi2
Mutual Info
F-score
Point-biserial
Embedded
LASSO
Auto-encoder with bottleneck