# Bil 470 / YAP 470

## Introduction to Machine Learning (Yapay Öğrenme)

Batuhan Bardak

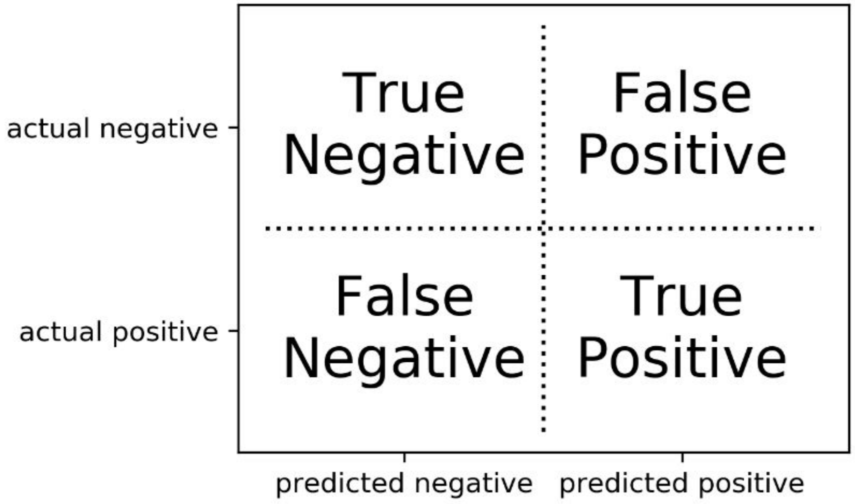
**Lecture 4**: Evaluation metrics, Feature Selection

**Date**: 27.09.2022

# Plan for today

- Evaluation metrics
- Feature Selection

# Evaluation metrics for binary classification




$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

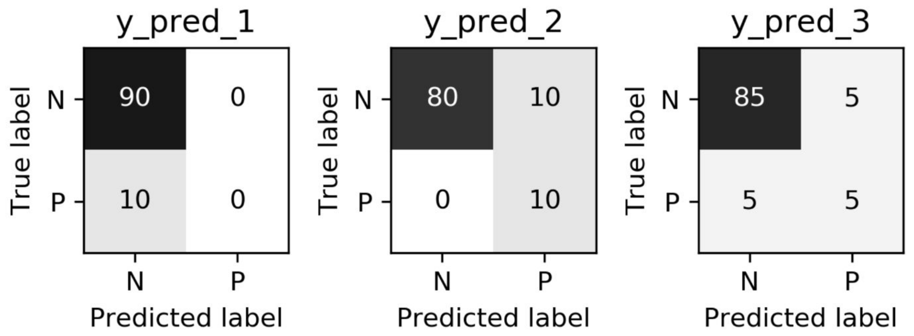
**Slide Credit:** Amuller, COMS4995

# Problems with Accuracy

Data with 90% negatives:

```
from sklearn.metrics import accuracy_score
for y_pred in [y_pred_1, y_pred_2, y_pred_3]:
    print(accuracy_score(y_true, y_pred))
```

```
0.9
0.9
0.9
```



# Evaluation Metrics for Classification

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Positive Predicted Value (PPV)

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Sensitivity, coverage, true positive rate.

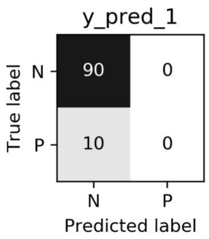
$$\text{F} = 2\,\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Harmonic mean of precision and recall

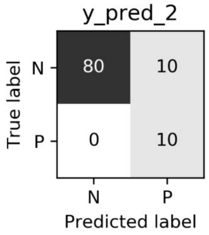
**Slide Credit:** Amuller, COMS4995

# Evaluation Metrics for Classification

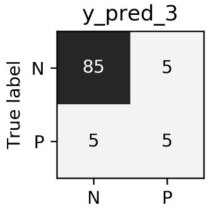
```
classification_report(y_true, y_pred)
```




```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95        90
           1       0.00      0.00      0.00        10

    accuracy                           0.90       100
   macro avg       0.45      0.50      0.47       100
weighted avg       0.81      0.90      0.85       100
```




```
              precision    recall  f1-score   support

           0       1.00      0.89      0.94        90
           1       0.50      1.00      0.67        10

    accuracy                           0.90       100
   macro avg       0.75      0.94      0.80       100
weighted avg       0.95      0.90      0.91       100
```




```
              precision    recall  f1-score   support

           0       0.94      0.94      0.94        90
           1       0.50      0.50      0.50        10

    accuracy                           0.90       100
   macro avg       0.72      0.72      0.72       100
weighted avg       0.90      0.90      0.90       100
```

**Slide Credit:** Amuller, COMS4995

# Averaging Strategies

$$\text{macro} \; \frac{1}{|L|} \sum_{l \in L} R(y_l, \hat{y}_l)$$

$$\text{weighted} \; \frac{1}{n} \sum_{l \in L} n_l R(y_l, \hat{y}_l)$$

```
print("Weighted average: ", recall_score(y_test, y_pred_1, average="weighted"))
print("Macro average: ", recall_score(y_test, y_pred_1, average="macro"))
```

```
Weighted average: 0.90
Macro average: 0.50
```

**Slide Credit:** Amuller, COMS4995

# Balanced Accuracy

```
balanced_accuracy_score(y_t, y_p) == recall_score(y_t, y_p, average='macro')
```

$$\text{balanced\_accuracy} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right)$$

- Always 0.5 for chance predictions

- Equal to accuracy for balanced datasets

**Slide Credit:** Amuller, COMS4995

# Changing Thresholds

```
y_pred = rf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       False       0.99      1.00      0.99      2732
        True       0.90      0.56      0.69        64

    accuracy                           0.99      2796
   macro avg       0.94      0.78      0.84      2796
weighted avg       0.99      0.99      0.99      2796
```

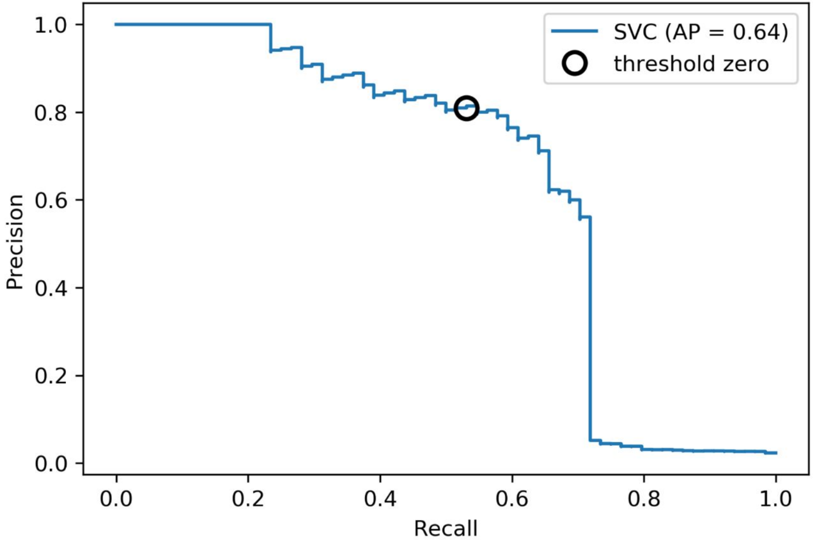
```
y_pred = rf.predict_proba(X_test)[:, 1] > .30


print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       False       0.99      0.99      0.99      2732
        True       0.71      0.64      0.67        64

    accuracy                           0.99      2796
   macro avg       0.85      0.82      0.83      2796
weighted avg       0.99      0.99      0.99      2796
```

**Slide Credit:** Amuller, COMS4995

# Precision-Recall Curve

```
svc = make_pipeline(StandardScaler(), SVC(C=100, gamma=0.1))
svc.fit(X_train, y_train)
plot_precision_recall_curve(svc, X_test, y_test, name='SVC')
```



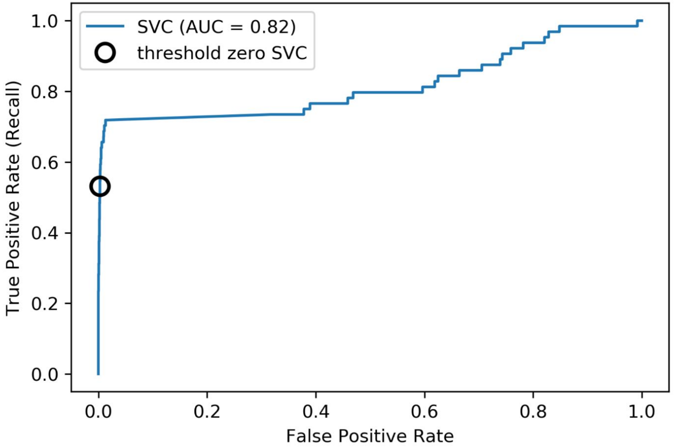



Slide Credit: Amuller, COMS4995

# ROC Curve

(Receiver Operating Characteristic)

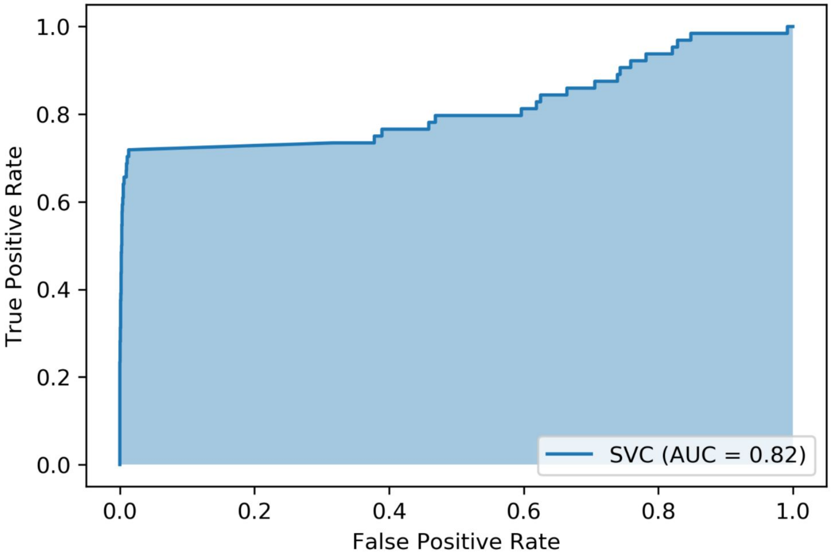
```
plot_roc_curve(svc, X_test, y_test, name='SVC')
```




- True positive rate (recall)
- False Positive Rate (FPR)
    - Negative instances that are incorrectly classified as positive.
    - 1 - True negative rate (specificity)

**Slide Credit:** Amuller, COMS4995

# Area Under ROC Curve (AUC)




- Always .5 for random/constant prediction

# Summary of metrics for binary classification

- Threshold-based
  - (balanced) accuracy
  - precision , recall, f1

- Ranking
  - Average precision
  - ROC AUC

**Slide Credit:** Amuller, COMS4995

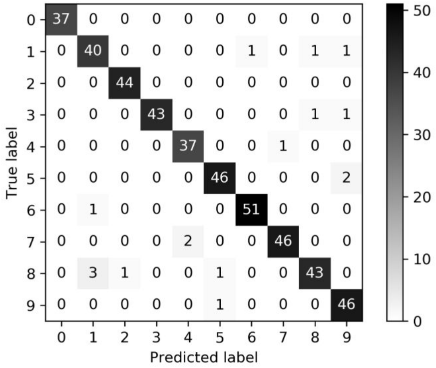
# Picking metrics

- Accuracy rarely what you want
- Problems are rarely balanced
- Find the right criterion for the task
- OR pick a substitude, but at least think about it
- Emphasis on recall or precision?
- Which classes are the important ones?

# Multi-class classification

```python
from sklearn.datasets import load_digits
from sklearn.metrics import accuracy_score

digits = load_digits()
 # data is between 0 and 16
X_train, X_test, y_train, y_test = train_test_split(
    digits.data / 16., digits.target, random_state=0)
lr = LogisticRegression().fit(X_train, y_train)
pred = lr.predict(X_test)
print("Accuracy: {:.3f}".format(accuracy_score(y_test, pred)))
plot_confusion_matrix(lr, X_test, y_test, cmap='gray_r')
```

```
Accuracy: 0.964
```



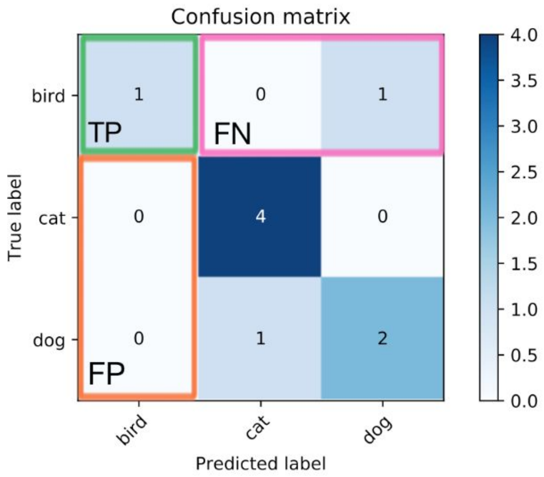

```python
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        37
           1       0.91      0.93      0.92        43
           2       0.98      1.00      0.99        44
           3       1.00      0.96      0.98        45
           4       0.95      0.97      0.96        38
           5       0.96      0.96      0.96        48
           6       0.98      0.98      0.98        52
           7       0.98      0.96      0.97        48
           8       0.96      0.90      0.92        48
           9       0.92      0.98      0.95        47

    accuracy                           0.96       450
   macro avg       0.96      0.96      0.96       450
weighted avg       0.96      0.96      0.96       450
```

**Slide Credit:** Amuller, COMS4995

# Multiclass classification

| Label | Predicted |
|---|---|
| cat | cat |
| cat | cat |
| cat | cat |
| cat | cat |
| dog | dog |
| dog | dog |
| dog | cat |
| bird | dog |
| bird | bird |




| | TP | FP | FN |
|---|---|---|---|
| bird | 1 | 0 | 1 |
| cat | 4 | 1 | 0 |
| dog | 2 | 1 | 1 |
| TOTAL | 7 | 2 | 2 |

$$Precision_{birds} = \frac{TP_{birds}}{TP_{birds} + FP_{birds}} = \frac{1}{1+0} = 1$$

$$Precision_{cats} = \frac{TP_{cats}}{TP_{cats} + FP_{cats}} = \frac{4}{4+1} = 0.8$$

$$Precision_{dogs} = \frac{TP_{dogs}}{TP_{dogs} + FP_{dogs}} = \frac{2}{2+1} = 0.667$$

**Slide Credit:** MARIA KHALUSOVA

# Multiclass classification

- **Micro-averaged**: all samples equally contribute to the final averaged metric
- **Macro-averaged:** all classes equally contribute to the final averaged metric
- **Weighted-averaged**: each classes contribution to the average is weighted by its size

| | TP | FP | FN | Precision | Number of samples |
|---|---|---|---|---|---|
| bird | 1 | 0 | 1 | 1 | 2 |
| cat | 4 | 1 | 0 | 0.8 | 4 |
| dog | 2 | 1 | 1 | 0.667 | 3 |
| TOTAL | 7 | 2 | 2 | | |

$$\text{Micro-averaged Precision} = \frac{TP_{total}}{TP_{total} + FP_{total}} = \frac{7}{7+2} = 0.7777$$

$$\text{Macro-averaged Precision} = \frac{1}{3} Precision_{birds} + Precision_{cats} + Precision_{dogs} = \frac{1}{3}(1 + 0.8 + 0.6666) = 0.8222$$

$$\text{Weighted-averaged Precision} = \frac{Precision_{birds} * N_{birds} + Precision_{cats} * N_{birds} + Precision_{dogs} * N_{birds}}{\text{Total number of samples}} = \frac{1 * 2 + 0.8 * 4 + 0.6666 * 3}{2 + 4 + 3} = 0.8$$

**Slide Credit:** MARIA KHALUSOVA

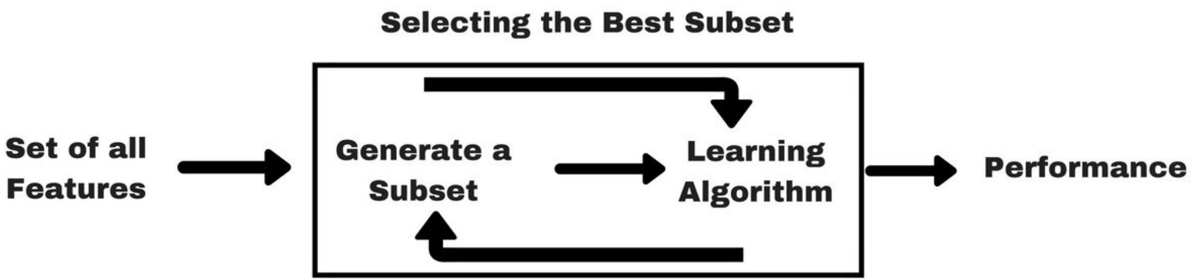
# Feature Selection

- Filtering-based feature selection
- Wrapper-based feature selection
- Embedded feature selection
  - Regularization
- Dimensionality Reduction
  - Do not select features, instead construct new features that are effectively represent combinations original features
- Motivation
  - Avoid overfitting
  - Faster prediction and training
  - Less storage for model
  - More interpretable model

# Filter Based Feature Selection


- Variance-based: 0 variance or mostly constant
- Covariance-based: remove correlated features (or pearson corr.)
- Chi-Square test: a statistical test that compares the frequencies of a term between different classes

# Wrapper Based Feature Selection




- Forward Selection
- Backward Elimination
- Recursive Feature Elimination

**Slide Credit:** MARIA KHALUSOVA

# Forward Feature Selection

- **Forward Feature Selection**: is an iterative method in which we start having 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.

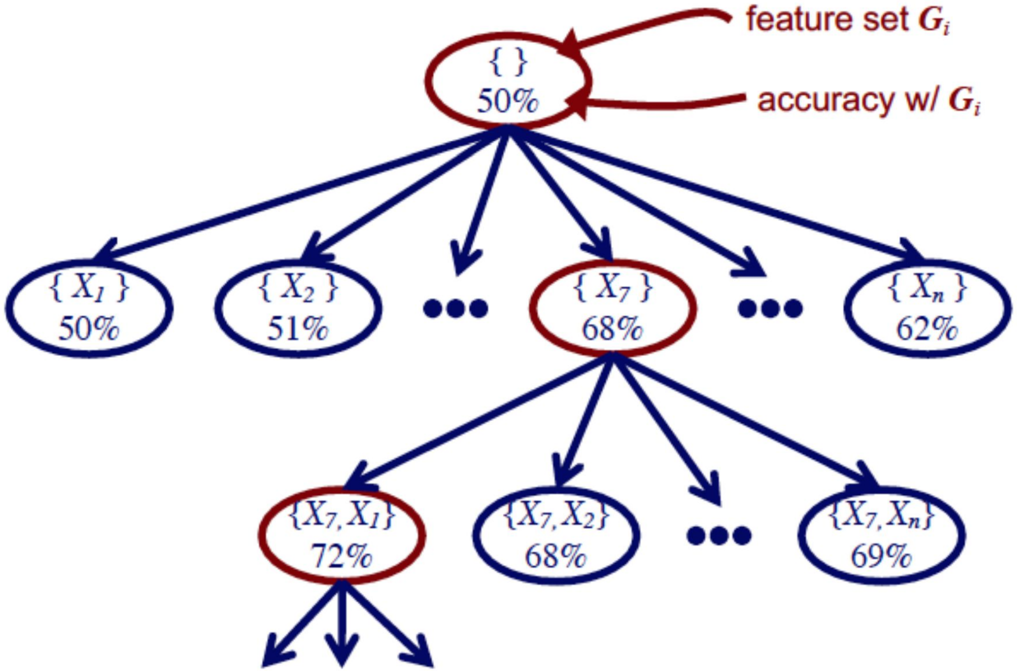
Given: feature set $\{X_i ,\ldots, X_n\}$, training set $D$, learning method $L$

$\boldsymbol{F} \leftarrow \{\ \}$
while score of $\boldsymbol{F}$ is improving
  for $i \leftarrow 1$ to $n$ do
    if $X_i \notin \boldsymbol{F}$
      $\boldsymbol{G}_i \leftarrow \boldsymbol{F} \cup \{X_i\}$
      $Score_i = \text{Evaluate}(\boldsymbol{G}_i, L, D)$
  $\boldsymbol{F} \leftarrow \boldsymbol{G}_b$ with best $Score_b$
return feature set $\boldsymbol{F}$

scores feature set $G$ by learning model(s) with $L$ and assessing its (their) accuracy

Slide Credit: Bilkent, CS 464

# Forward Feature Selection



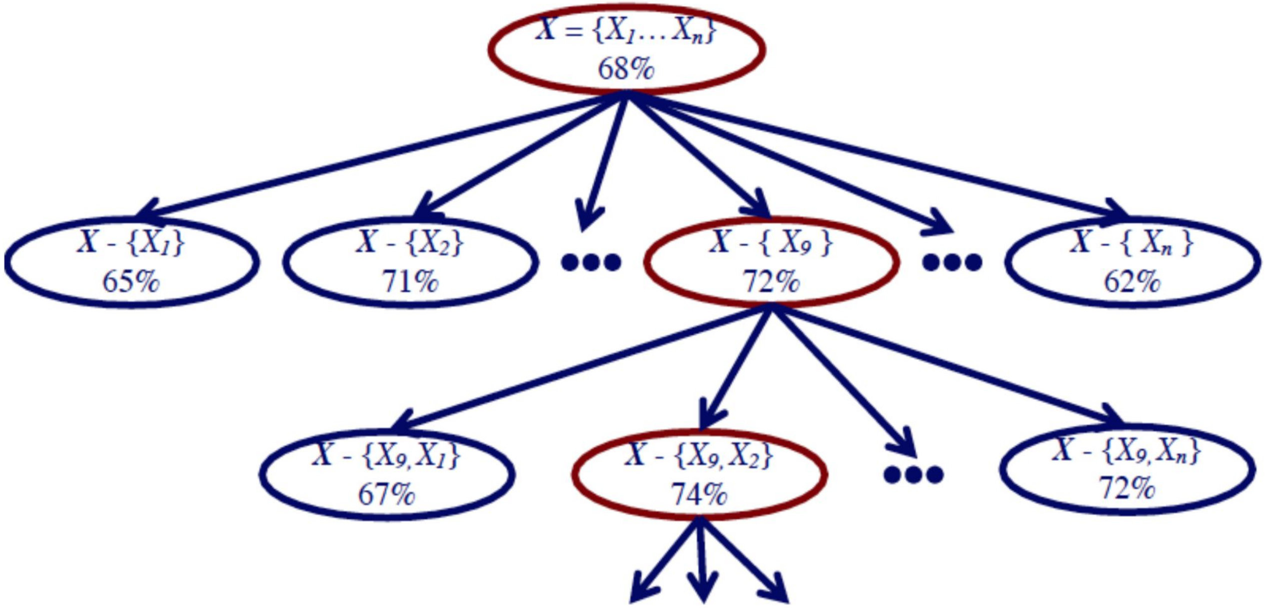

**Slide Credit:** Bilkent, CS 464

# Backward Feature Elimination

- **Backward Feature Elimination:** In backward elimination, we start with all the features and 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.

**Slide Credit:** Bilkent, CS 464

# Backward Feature Elimination

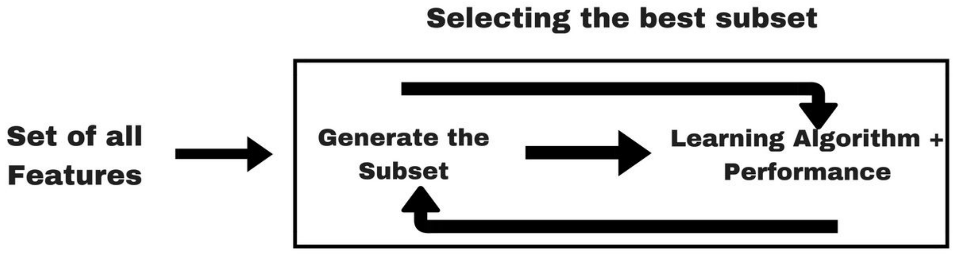



**Slide Credit:** Bilkent, CS 464

# Forward Selection vs Backward Elimination

- Forward Selection
  - Efficient for choosing a small subset of the features
  - Misses features whose usefulness requires other features (feature synergy)
- Backward elimination
  - Efficient for discarding a small subset of the features
  - Preserves features whose usefulness requires other features

# Embedding Based Feature Selection




- **Regularization**
  - Lasso Regression
  - Ridge Regression
- **Model-Based Selection**
  - Feature importance ()

# Difference between Filter and Wrapper Methods

- Filter methods measure the relevance of features by their correlation with dependent variable, while wrapper methods measure the usefulness of a subset of feature by actually training a model on it.
- Filter methods are much faster compared to wrapper methods.
- Filter methods use statistical methods for evaluation of a subset of features while wrapper methods use cross validation.
- Filter methods might fail to find the best subset of features in many occasions.
- Using the subset of features from the wrapper methods make the model more prone to overfitting as compared to using subset of features from the filter methods.

**Slide Credit:** analyticsvidhya

# Next Class:

Logistic Regression, Neural Network