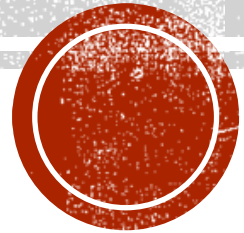
LECTURE # 07:

SUPPORT VECTOR MACHINE



INTRODUCTION:

- Support Vector Machine (SVM) is a powerful and widely used machine learning algorithm for both classification and regression tasks. It is a supervised learning model that finds the optimal hyperplane (decision boundary) that separates different classes in a high-dimensional feature space.

FUNDAMENTAL IDEA BEHIND SVM:

- The fundamental idea behind SVM is to maximize the margin between the decision boundary and the closest data points (called support vectors) from each class. By maximizing the margin, SVM aims to find the most robust and generalized decision boundary, leading to better classification or regression performance.

REAL WORLD APPLICATION:

- SVM has been successfully applied in various domains due to its versatility and effectiveness. Some real-world applications of SVM include:
- Text classification and sentiment analysis
- Image recognition and object detection
- Bioinformatics and genomic pattern detection
- Intrusion detection and network security
- Financial data analysis and credit risk assessment
- Speech and handwriting recognition
- Protein structure prediction

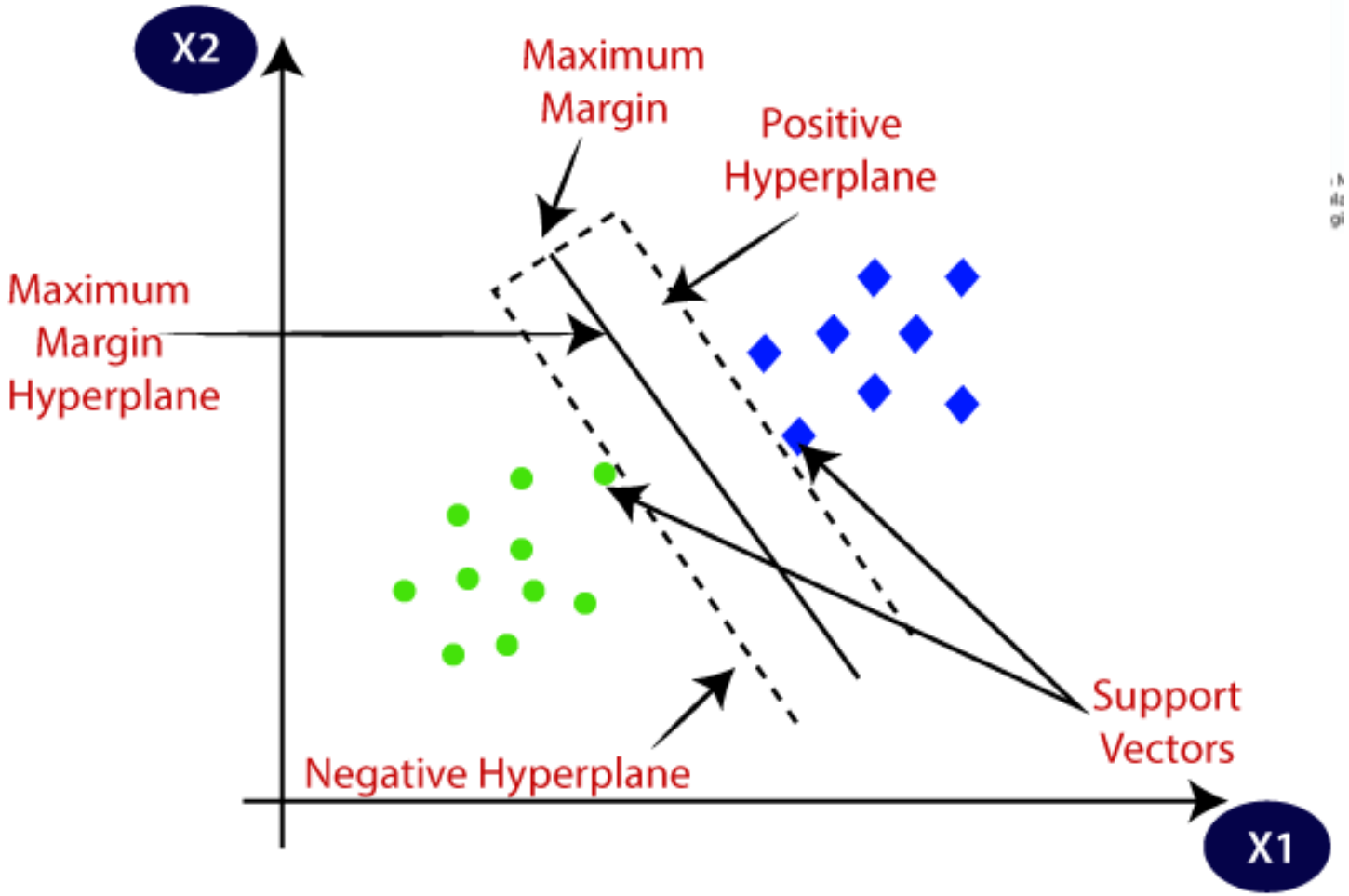
SVM REGRESSION ADVANTAGES:

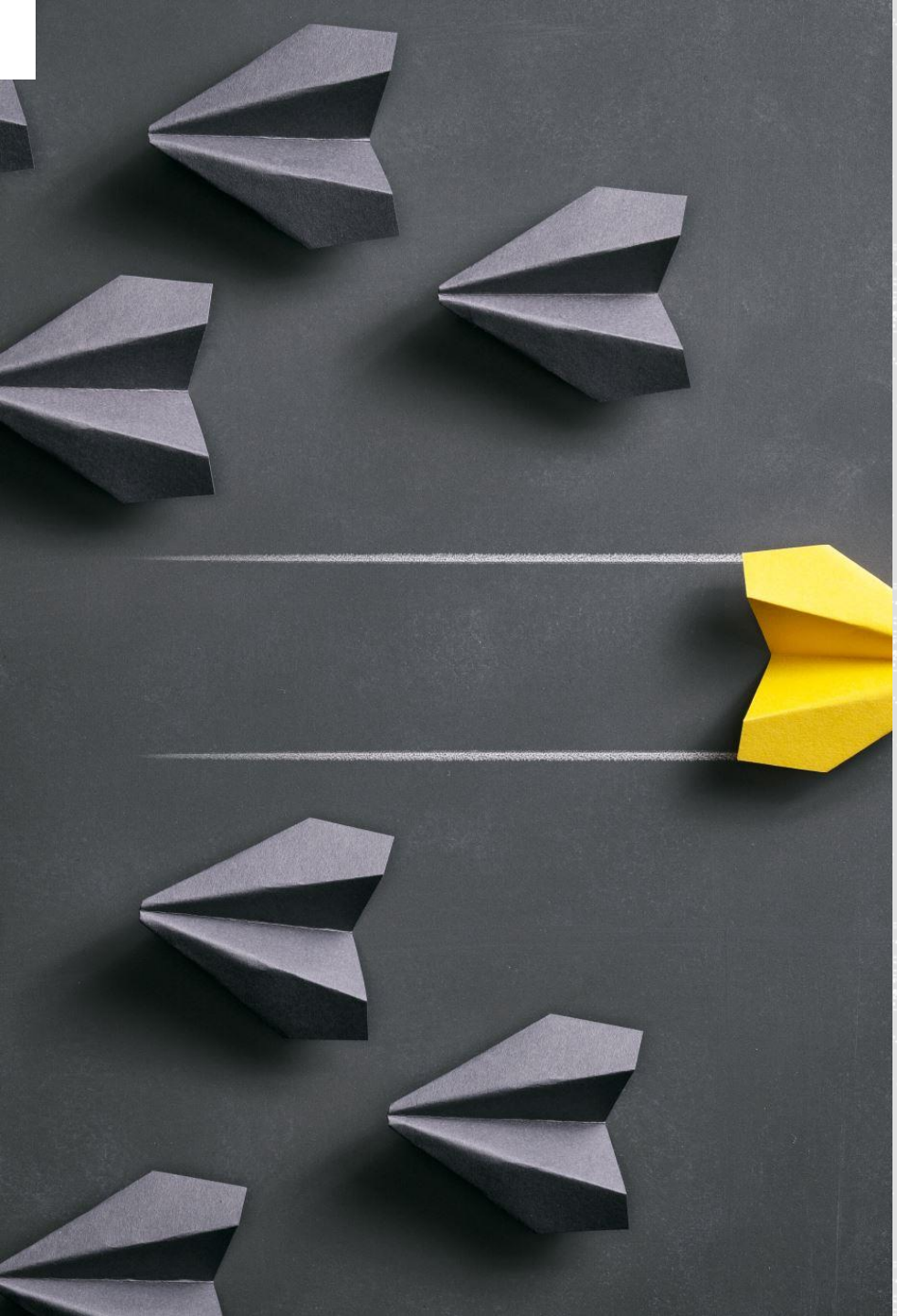
1. **Predicting the Future**:
    1. Regression analysis helps you make predictions about the future.
    2. For example, if you know how much it rained in the past and how much crop yield was produced, regression can predict how much crop yield you might get with a certain amount of rainfall in the future.
    3. This is useful for planning and decision-making.
2. **Understanding Relationships**:
    1. Regression analysis helps you understand how different factors are related to each other.
    2. For instance, you can see how a person's height, weight, and age are related to their blood pressure.
    3. Understanding these relationships can help you make better decisions or take preventive measures.
3. **Identifying Important Factors**:
    1. Regression analysis can identify which factors are most important in influencing an outcome.
    2. For example, in a study of house prices, regression can tell you which factors like location, size, age of the house, etc., have the biggest impact on the price.
    3. This information can be valuable for decision-makers, policymakers, or businesses.

HOW SVM WORKS:

- SVM works by finding the optimal hyperplane that separates different classes in a high-dimensional feature space. The hyperplane is chosen in such a way that the margin (distance between the hyperplane and the nearest data points from each class) is maximized.

HOW SVM WORKS





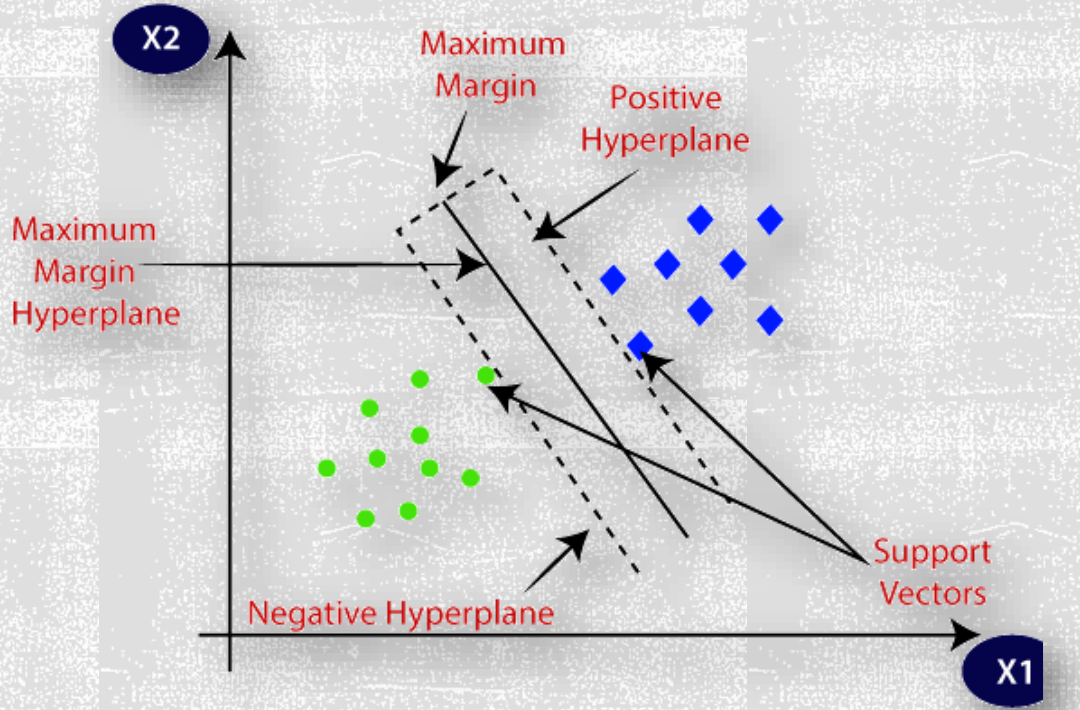
WHAT IS HYPER PLANE:

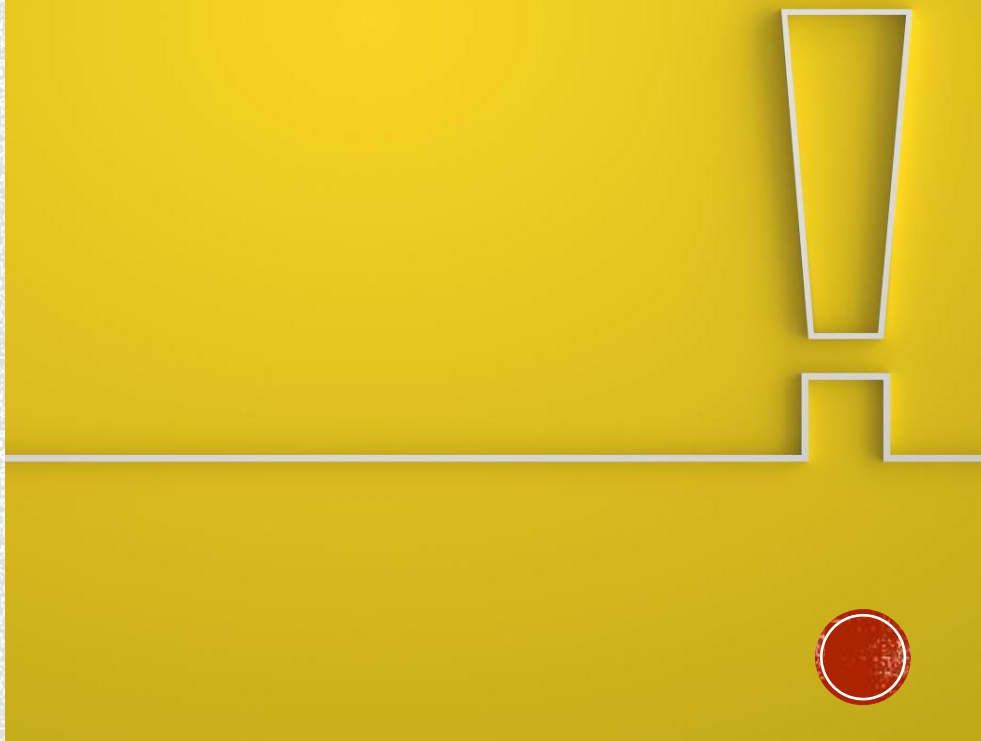
imagine you're playing a game with your toys. You have different toys, like balls, blocks, and cars, and you want to separate them into two groups based on their colors. But here's the twist: you can only use a single piece of cardboard to divide them.
Now, think of this cardboard as our hyperplane. Your goal is to position this cardboard in such a way that it splits your toys into two groups as best as possible. For example, all the red toys go on one side, and all the blue toys go on the other.

Now, think of the cardboard as the hyperplane in Support Vector Machines (SVM). In SVM, we use this hyperplane to separate different kinds of data, just like you separated your toys based on their colors. The goal is to find the best position for the hyperplane so that it maximally separates the different groups of data, just like you tried to position the cardboard to best separate your toys.

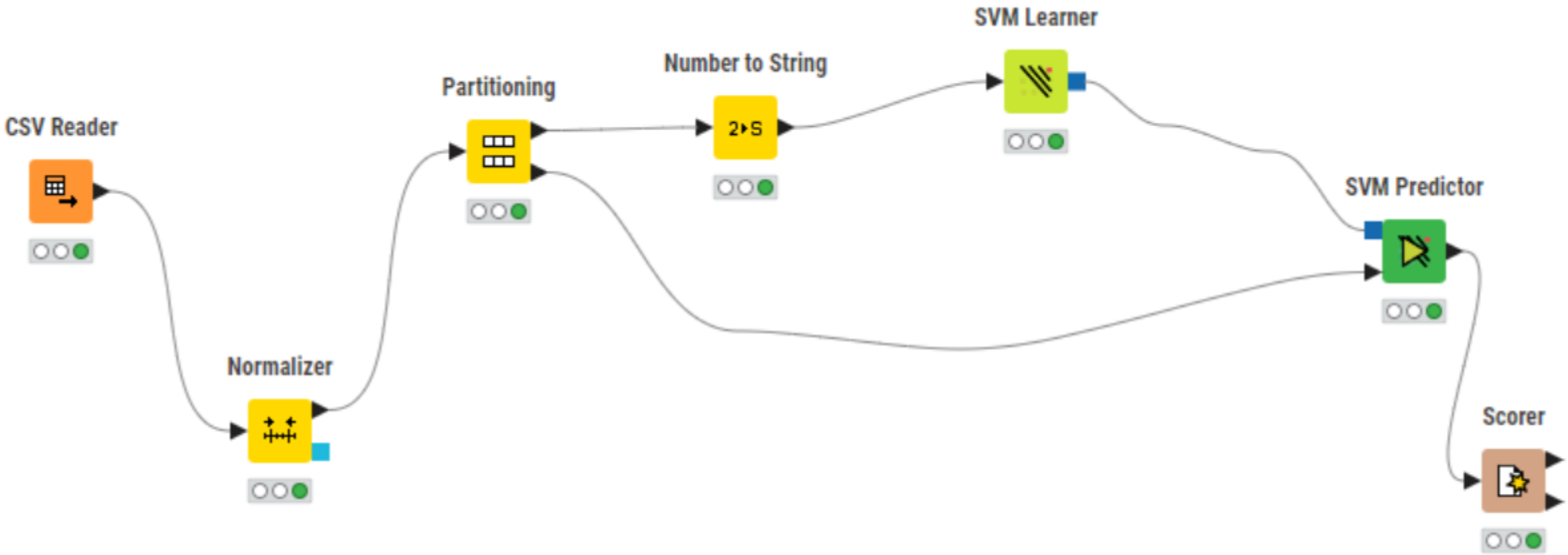
WHAT IS MARGIN?

- In SVM, the margin is like a safety zone around the decision boundary (the line separating different classes of data). It helps prevent overfitting by ensuring that the decision boundary is not too close to the training data points the margin in SVM ensures a clearer distinction between different classes of data.

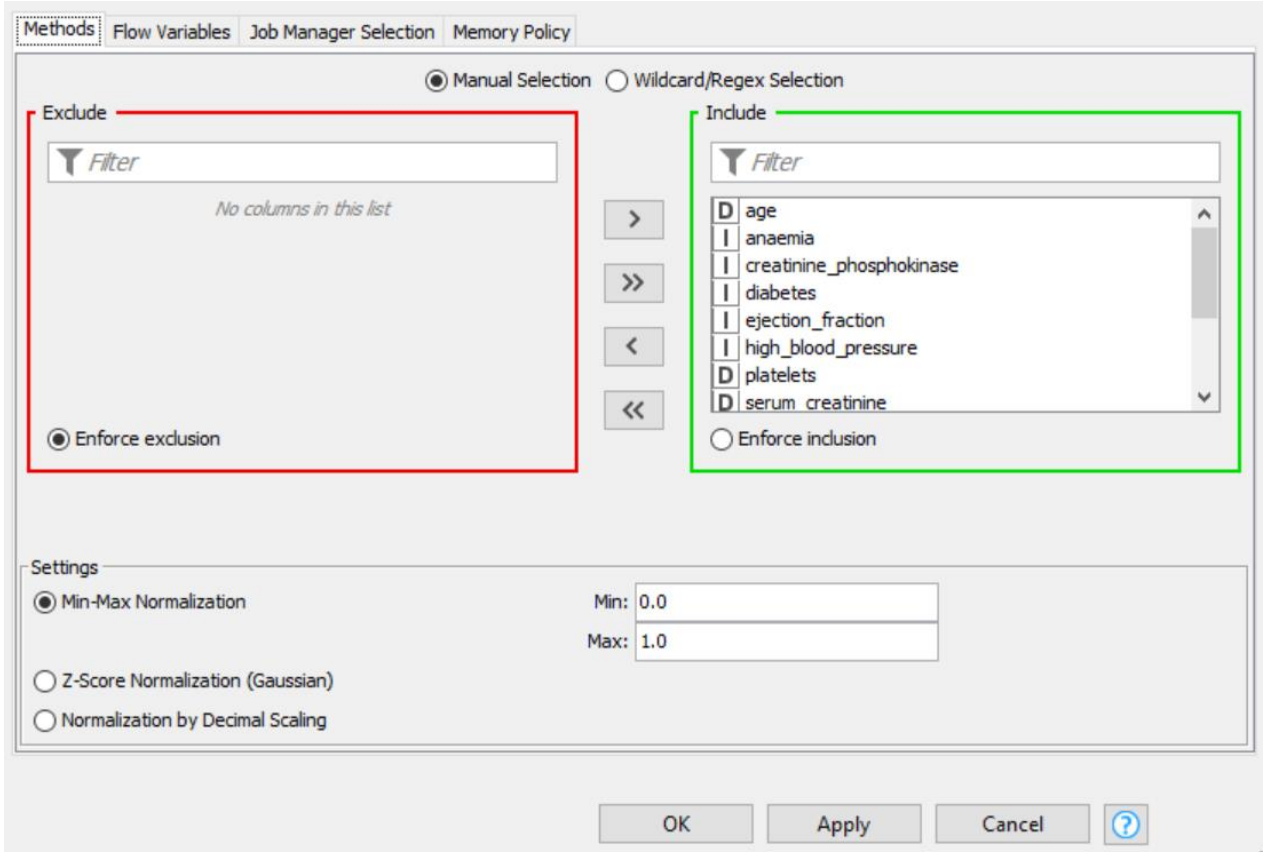




SVM IMPLEMENTATION IN KNIME:



NORMALIZATION CONFIGURATION:



DEATH EVENT PREDICTION BY SVM

ejection_f... Number (dou...	high_bloo... Number (dou...	platelets Number (dou...	serum_cr... Number (dou...	serum_so... Number (dou...	sex Number (dou...	smoking Number (dou...	time Number (dou...	Predictio... String
0.091	0	0.224	0.157	0.686	1	0	0.011	1.0
0.015	0	0.124	0.079	0.686	1	0	0.021	1.0
0.167	1	0.276	0.045	0.771	1	1	0.021	1.0
0.621	0	0.075	0.157	0.229	0	0	0.039	1.0
0.167	1	0.304	0.09	0.686	0	0	0.043	1.0
0.545	1	0.197	0.056	0.771	1	0	0.089	0.0
0.091	1	0.289	0.149	0.6	1	1	0.096	1.0
0.47	1	0.194	0.079	0.743	1	1	0.103	0.0
0.167	1	0.298	0.045	0.486	1	0	0.121	1.0
0.091	1	0.356	0.067	0.657	1	1	0.181	0.0
0.394	1	0.231	0.202	0.514	0	0	0.199	0.0
1	0	0.289	0.076	0.686	0	0	0.21	0.0
0.697	1	0.354	0.034	0.771	0	1	0.249	0.0
0.47	1	0.362	0.045	0.771	0	0	0.249	0.0
0.545	0	0.289	0.076	0.686	1	1	0.256	0.0
0.318	1	0.179	0.067	0.686	1	0	0.267	0.0
0.697	1	0.338	0.034	0.771	1	0	0.267	0.0
0.167	1	0.189	0.067	0.886	1	0	0.267	0.0
0.47	0	0.33	0.056	0.571	1	0	0.27	0.0
0.697	0	0.279	0.067	0.657	0	0	0.288	0.0
0.394	0	0.237	0.067	0.771	0	0	0.292	0.0

CONFUSION MATRIX

Prediction ...	1.0	0.0
1.0	14	0
0.0	0	46

Correct classified: 60

Wrong classified: 0

Accuracy: 100%

Error: 0%

Cohen's kappa (κ): 1%

SVM IN PYTHON: IMPORT LIBRARIES:

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

LOAD DATASET:

```
# Load the iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

SPLIT DATASET:

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

DEFINE THE KERNEL:

```
# Define the kernels
kernels = ['linear', 'poly', 'rbf', 'sigmoid']

for kernel in kernels:
    # Create an SVM classifier with the specified kernel
    svm = SVC(kernel=kernel)
```

TRAIN SVM CLASSIFIER:

```
# Train the classifier
svm.fit(X_train, y_train)
```

PREDICTION ON TEST DATASET:

```
# Make predictions on the test set
y_pred = svm.predict(X_test)
```

CALCULATE ACCURACY OOF SVM:

```
# Calculate the accuracy score
accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy for {kernel} kernel: {accuracy:.2f}")
```

SVM ON TENNIS DATASET:

```python
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler

# Load the dataset
data = pd.read_csv('tennis dataset.csv')

# Convert categorical features to numerical values
encoder = LabelEncoder()
data['Outlook'] = encoder.fit_transform(data['Outlook'])
data['Temperature'] = encoder.fit_transform(data['Temperature'])
data['Humidity'] = encoder.fit_transform(data['Humidity'])
data['Wind'] = encoder.fit_transform(data['Wind'])

# Split the data into features (X) and target variable (y)
X = data.drop('Play _Tennis', axis=1)
y = data['Play _Tennis']

# Convert target variable to numerical values
y = encoder.fit_transform(y)

# Scale the features
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create an SVM model instance
svm_model = SVC(kernel='linear', C=1.0, random_state=42)

# Train the SVM model
svm_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = svm_model.predict(X_test)

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 1.0
```

SVM ON HEART FAILURE DATASET:

```
[ ]  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import StandardScaler
     from sklearn.svm import SVC
     from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

     # Load the dataset
     data = pd.read_csv('/content/heart_failure_clinical_records_dataset.csv')

     # Split the data into features (X) and target variable (y)
     X = data.drop('DEATH_EVENT', axis=1)
     y = data['DEATH_EVENT']

     # Split the data into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

     # Scale the features
     scaler = StandardScaler()
     X_train = scaler.fit_transform(X_train)
     X_test = scaler.transform(X_test)

     # Create an SVM model instance
     svm_model = SVC(kernel='linear', C=1.0, random_state=42)

     # Train the SVM model
     svm_model.fit(X_train, y_train)

     # Make predictions on the test set
     y_pred = svm_model.predict(X_test)

     # Evaluate the model's performance
     accuracy = accuracy_score(y_test, y_pred)
     print(f'Accuracy: {accuracy}')

     # Generate and print classification report and confusion matrix
     print("Classification Report:")
     print(classification_report(y_test, y_pred))
     print("Confusion Matrix:")
     print(confusion_matrix(y_test, y_pred))
```

OUTPUT:

```
Accuracy: 0.8
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.94      0.85        35
           1       0.88      0.60      0.71        25

    accuracy                           0.80        60
   macro avg       0.82      0.77      0.78        60
weighted avg       0.82      0.80      0.79        60

Confusion Matrix:
[[33  2]
 [10 15]]
```

TASK 1:

- write a python program to implement the support vector machine on Diabetes dataset. implement the following different kernels of SVM and compare the accuracy score and visualize the confusion matrix and hyperplane.

TASK 2:

- Design the workflow with the help of KNIME to implement the Support Vector Machine Algorithm on any classification dataset