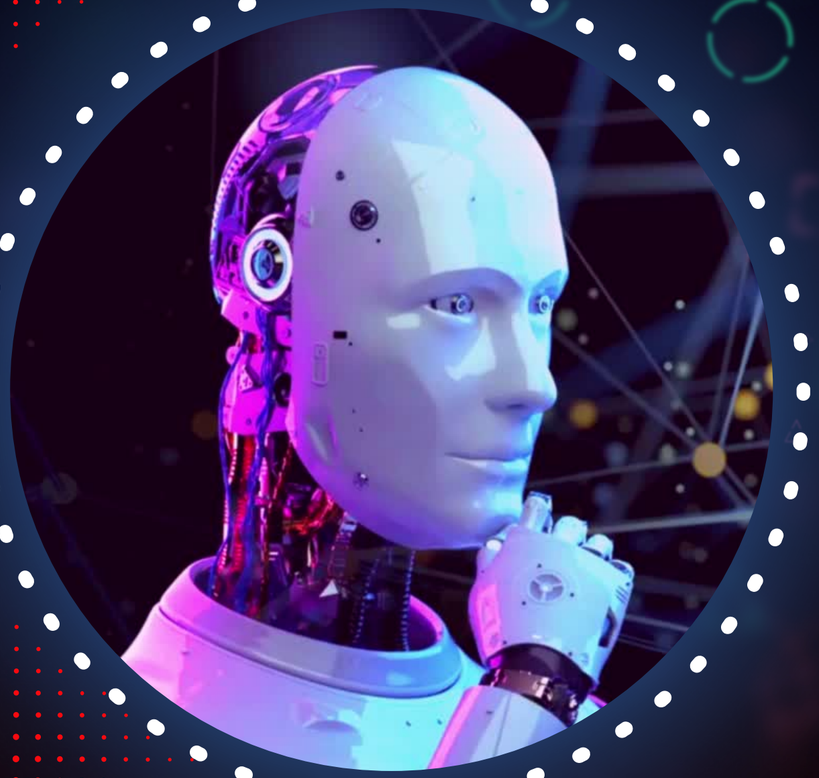
Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

LIVE



MACHINE LEARNING

DAY – 12

Steps in Machine Learning

- Data collection
- Preprocessing
- train, test, split
- Feature selection
- Model - selection, training
- Predictions
- Evaluate and improve

TRAIN TEST SPLIT

TRAIN_TEST_SPLIT

- In machine learning, the train-test split is a technique used to evaluate the performance of a predictive model.

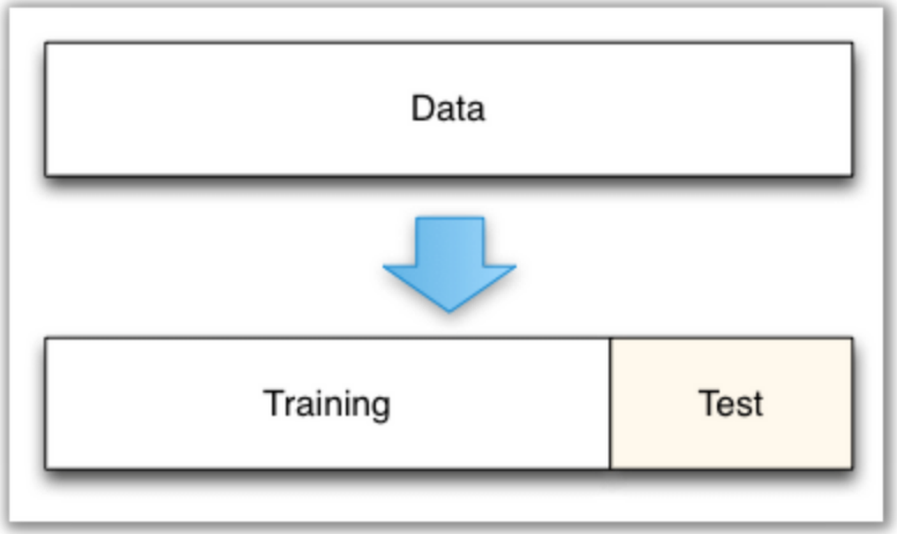
The basic idea is to split the available data into two parts:

- one part is used to train the model, and the other part is used to test its performance.

TRAIN_TEST_SPLIT

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=21)
```



TRAIN_TEST_SPLIT PARAMETERS

Parameters:

arrays : *sequence of indexables with same length / shape[0]
Allowed inputs are lists, numpy arrays, scipy-sparse matrices or pandas dataframes.

test_size : *float or int, default=None*
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples. If None, the value is set to the complement of the train size. If `train_size` is also None, it will be set to 0.25.

train_size : *float or int, default=None*
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the train split. If int, represents the absolute number of train samples. If None, the value is automatically set to the complement of the test size.

random_state : *int, RandomState instance or None, default=None*
Controls the shuffling applied to the data before applying the split. Pass an int for reproducible output across multiple function calls. See Glossary.

shuffle : *bool, default=True*
Whether or not to shuffle the data before splitting. If shuffle=False then stratify must be None.

stratify : *array-like, default=None*
If not None, data is split in a stratified fashion, using this as the class labels. Read more in the User Guide.

FEATURE SELECTION

FEATURE SELECTION

- It is the process of reducing the number of input variables when developing a predictive model.

- It is desirable to reduce the number of input variables to both reduce the computational cost of modelling and also in some case to improve the performance of the model and get rid of noise.

FEATURE SELECTION METHODS

- Dropping constant features
- Correlation method
- Mutual information
- Chi-square test
- ANOVA F-value

DROPPING CONSTANT FEATURE

- It is the process of dropping constant features which are actually not important for solving the problem
- It can be done with the help of variance threshold from sklearn
- Variance threshold identifies features based on threshold value

VARIANCE THRESHOLD

```
from sklearn.feature_selection import VarianceThreshold

variance = VarianceThreshold(threshold=0)

variance.fit(data)

print(variance.get_support())
```

CORRELATION METHOD

- The idea behind this method is to remove all the columns which are highly correlated
- In feature selection using correlation, we typically set a threshold for the correlation coefficient and select only those features that have a correlation coefficient greater than the threshold with the target variable.

CORRELATION METHOD

```
def correlation_method(data, threshold):
  corr_column = set()
  corr = data.corr()
  for i in range(len(corr.columns)):
    for j in range(i):
      if abs(corr.iloc[i,j]) > threshold:
        column_name = corr.columns[i]
        corr_column.add(column_name)
  return corr_column


correlation_method(x, 0.7)
```

MUTUAL INFORMATION

- It is a statistical measure that tells how much two random variable are related to each other
- The value of Mutual Information will be zero if the two variable are independent
- The value of Mutual information will be high with higher dependency
- Mutual information for classifier and regressor is available in sklearn

MUTUAL INFORMATION CLASSIFIER

```
from sklearn.feature_selection import mutual_info_classif

mi = mutual_info_classif(x_train, y_train)

mi = pd.Series(mi)

mi.index = x_train.columns

mi.sort_values(ascending=False)
```

SELECT USING K BEST

```
from sklearn.feature_selection import mutual_info_classif

from sklearn.feature_selection import SelectKBest

select = SelectKBest(mutual_info_classif, k=5)

select.fit(x_train,y_train)

x_train.columns[select.get_support()]
```

SELECT USING PERCENTILE

```
from sklearn.feature_selection import mutual_info_classif

from sklearn.feature_selection import SelectPercentile

select = SelectPercentile(mutual_info_classif, Percentile=20)

select.fit(x_train,y_train)

x_train.columns[select.get_support()]
```

MUTUAL INFORMATION REGRESSION

```
from sklearn.feature_selection import mutual_info_regression

from sklearn.feature_selection import SelectKBest

select = SelectKBest(mutual_info_regression, k=5)

select.fit(x_train,y_train)

x_train.columns[select.get_support()]
```

CHI-SQUARED TEST

- The chi-squared test is a statistical test that can be used for feature selection in machine learning.

- It is commonly used to determine the statistical significance of a feature's relationship with a target variable in a classification problem.

CHI-SQUARED TEST

```
from sklearn.feature_selection import SelectKBest, chi2

selector = SelectKBest(chi2, k=2)

selector.fit(x,y)

x.columns[selector.get_support()]
```

ANOVA - F TEST

- ANOVA (Analysis of Variance) is a statistical method used to test the difference in means between two or more groups.

- It works by comparing the variance between two or more groups, identifying the features with significant differences, and ranking them based on the F-score.

ANOVA - F TEST

```
from sklearn.feature_selection import SelectKBest, f_classif

selector = SelectKBest(f_classif, k=2)

selector.fit(x,y)

x.columns[selector.get_support()]
```

THANK YOU