Feature Selection:(Formal definition) Feature selection is a process that choose a subset of features from the original features so that the feature space is optimally reduced according to a certain criterion.

The goal of the feature selection in machine learning:
1) To reduce the dimensionality of feature space
2) To speed up a learning algorithm
3) To improve the predictive accuracy of the algorithm

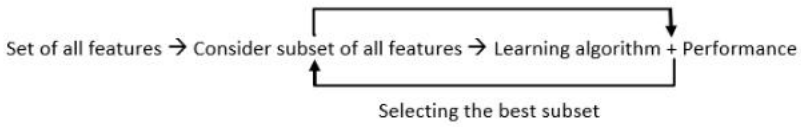
The are three general classes of feature selection algorithms:
1) Filter Methods--The filter method asses the relevance of features based on statistical tests like correlation coefficient ,chi-square test, or ,mutual information to evaluate the relationship between each feature and the target variable.

2) Wrapper Methods:This is an iterative method. First certain features are selected then the machine learning model is evaluated with its accuracy ,then different features are selected again and checks for the accuracy, based on the best accuracy features are selected.

-->The method gives the best feature selection process this is one of the pros
Cons:This method is computationally very expensive and results in over-fitting the training dataset.

-->uses techniques such as forward selection,backward elimination and recursive feature elimination

3) Embedded methods: The methods methods are selecting the features during the training of the model and testing the accuracy.The embedded methods have the merged or the combined adavantage of the both filter methods and the wrapper methods.



Embedded Methods Implementation

These are the some techniques followed by the embedded methods:

Regularization:This method adds a penalty to different parameters of the machine learning model to avoid over-fitting of the model.This approach of feature selection uses lasso(l1 regularization) and elastic nets(L1 and L2 regularization).The penalty is applied over the coefficients, thus bringing down some coefficients to zero. The features having zero coeffcient can be removed from the dataset.

Key differences between the feature engineering and feature selection:

Feature Selection:Helps in selecting only the relevant features for the ML model.so that it is possible to improve the model accuracy and efficiency.

Example:Consider having the dataset of 50 features, but in that particular dataset only 10 features are very useful and influential. So choosing 10 features and discarding all the other features is termed to be feature selection.

-->To perform this feature selection we use different algorithms like random forests etc.

Feature Engineering:Instead of concentrating on the feature selection, the feature engineering concentrates on the transformation of the features. All normalization and standardization ,applying the all mathematical aggregations and encoding of the data, dimensionality reduction comes under this feature engineering.

RMSE(Root mean square error):This helps in showing the possible error that can be made by the ML model. The error would be the exact difference between the actual value and the predicted value.

Consider this example: if the rmse I 20000, it means on average the models predictions differ form actual sale prices by 20000.

RMSE Formula:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Where:

- n is the number of observations (or data points).
- y_i is the actual value of the target variable for the i-th observation.
- $\hat{y}_i$ is the predicted value of the target variable for the i-th observation.
- $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ calculates the sum of squared differences between actual and predicted values.
- $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ computes the mean squared error (MSE).
- $\sqrt{\cdot}$ takes the square root of the MSE to obtain the RMSE.

Always the lower RMSE indicates the higher accuracy of the model predictions. Higher rmse indicates higher prediction errors and lower accuracy if the model.

Rmse calculation using python sk learn library:

```
Import numpy as np
from sklearn.metrics import mean_squared_error

#considering the sample actual and predicted values
```

```
actual=np.array({10,20,30,40,50})
Predicted =np.array({12,18,28,41,49})

Rmse=np.sqrt(mean_squared_error(actual,predicted)

Print(f'RMSE :{rmse:.2f}')
```

Applying scaling to Machine learning algorithms:

```
#calculating the value of the RMSE using the KNN:

from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error

knn = KNeighborsRegressor(n_neighbors=7)

rmse = []

trainX = [X_train, X_train_norm, X_train_stand]
testX = [X_test, X_test_norm, X_test_stand]

for i in range(len(trainX)):

      knn.fit(trainX[i],y_train)
      pred = knn.predict(testX[i])
    rmse.append(np.sqrt(mean_squared_error(y_test,pred)))

# visualizing the result
df_knn = pd.DataFrame({'RMSE':rmse},index=['Original','Normalized','Standardized'])
df_knn
```

Calculating the rsme using DecisionTree:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
dt = DecisionTreeRegressor(max_depth=10,random_state=27)

rmse = []

trainX = [X_train,X_train_norm,X_train_stand]
testX = [X_test,X_test_norm,X_test_stand]

for i in range(len(trainX)):
   dt.fit(trainX[i],y_train)
   pred = dt.predict(testX[i])
```

```
    rmse.append(np.sqrt(mean_squared_error(y_test,pred)))

# visualizing the result
df_dt = pd.DataFrame({'RMSE':rmse},index=['Original','Normalized','Standardized'])
df_dt
```