Ex-07-Feature-Selection

AIM

To Perform the various feature selection techniques on a dataset and save the data to a file.

Explanation

Feature selection is to find the best set of features that allows one to build useful models. Selecting the best features helps the model to perform well.

ALGORITHM

STEP 1

Read the given Data

STEP 2

Clean the Data Set using Data Cleaning Process

STEP 3

Apply Feature selection techniques to all the features of the data set

STEP 4

Save the data to the file

CODE

```
from sklearn.datasets import load_boston
boston_data=load_boston()
import pandas as pd
boston = pd.DataFrame(boston_data.data, columns=boston_data.feature_names)
boston['MEDV'] = boston_data.target
dummies = pd.get_dummies(boston.RAD)
boston = boston.drop(columns='RAD').merge(dummies,left_index=True,right_index=True)
X = boston.drop(columns='MEDV')
y = boston.MEDV
boston.head(10)

from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import KFold
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import cross_val_predict
from sklearn.linear_model import LinearRegression
from math import sqrt

cv = KFold(n_splits=10, random_state=None, shuffle=False)
classifier_pipeline = make_pipeline(StandardScaler(), KNeighborsRegressor(n_neighbors=10))
y_pred = cross_val_predict(classifier_pipeline, X, y, cv=cv)
print("RMSE: " + str(round(sqrt(mean_squared_error(y,y_pred)),2)))
print("R_squared: " + str(round(r2_score(y,y_pred),2)))

boston.var()

X = X.drop(columns = ['NOX','CHAS'])
y_pred = cross_val_predict(classifier_pipeline, X, y, cv=cv)
print("RMSE: " + str(round(sqrt(mean_squared_error(y,y_pred)),2)))
print("R_squared: " + str(round(r2_score(y,y_pred),2)))
```

```
# Filter Features by Correlation
import seaborn as sn
import matplotlib.pyplot as plt
fig_dims = (12, 8)
fig, ax = plt.subplots(figsize=fig_dims)
sn.heatmap(boston.corr(), ax=ax)
plt.show()
abs(boston.corr()["MEDV"])
abs(boston.corr()["MEDV"][abs(boston.corr()["MEDV"])>0.5].drop('MEDV')).index.tolist()
vals = [0.1,0.2,0.3,0.4,0.5,0.6,0.7]
for val in vals:
    features = abs(boston.corr()["MEDV"][abs(boston.corr()["MEDV"])>val].drop('MEDV')).index.tolist()

    X = boston.drop(columns='MEDV')
    X=X[features]

    print(features)

    y_pred = cross_val_predict(classifier_pipeline, X, y, cv=cv)
    print("RMSE: " + str(round(sqrt(mean_squared_error(y,y_pred)),2)))
    print("R_squared: " + str(round(r2_score(y,y_pred),2)))

# Feature Selection Using a Wrapper

boston = pd.DataFrame(boston_data.data, columns=boston_data.feature_names)
boston['MEDV'] = boston_data.target
boston['RAD'] = boston['RAD'].astype('category')
dummies = pd.get_dummies(boston.RAD)
boston = boston.drop(columns='RAD').merge(dummies,left_index=True,right_index=True)
X = boston.drop(columns='MEDV')
y = boston.MEDV

from mlxtend.feature_selection import SequentialFeatureSelector as SFS

sfs1 = SFS(classifier_pipeline,
```

```
            k_features=1,
            forward=False,
            scoring='neg_mean_squared_error',
            cv=cv)

X = boston.drop(columns='MEDV')
sfs1.fit(X,y)
sfs1.subsets_

X = boston.drop(columns='MEDV')[['CRIM','RM','PTRATIO','LSTAT']]
y = boston['MEDV']
y_pred = cross_val_predict(classifier_pipeline, X, y, cv=cv)
print("RMSE: " + str(round(sqrt(mean_squared_error(y,y_pred)),3)))
print("R_squared: " + str(round(r2_score(y,y_pred),3)))

boston[['CRIM','RM','PTRATIO','LSTAT','MEDV']].corr()

boston['RM*LSTAT']=boston['RM']*boston['LSTAT']

X = boston.drop(columns='MEDV')[['CRIM','RM','PTRATIO','LSTAT']]
y = boston['MEDV']
y_pred = cross_val_predict(classifier_pipeline, X, y, cv=cv)
print("RMSE: " + str(round(sqrt(mean_squared_error(y,y_pred)),3)))
print("R_squared: " + str(round(r2_score(y,y_pred),3)))

sn.pairplot(boston[['CRIM','RM','PTRATIO','LSTAT','MEDV']])

boston = boston.drop(boston[boston['MEDV']==boston['MEDV'].max()].index.tolist())

X = boston.drop(columns='MEDV')[['CRIM','RM','PTRATIO','LSTAT','RM*LSTAT']]
y = boston['MEDV']
y_pred = cross_val_predict(classifier_pipeline, X, y, cv=cv)
print("RMSE: " + str(round(sqrt(mean_squared_error(y,y_pred)),3)))
print("R_squared: " + str(round(r2_score(y,y_pred),3)))

boston['LSTAT_2']=boston['LSTAT']**2
```

```
X = boston.drop(columns='MEDV')[['CRIM','RM','PTRATIO','LSTAT']]
y_pred = cross_val_predict(classifier_pipeline, X, y, cv=cv)
print("RMSE: " + str(round(sqrt(mean_squared_error(y,y_pred)),3)))
print("R_squared: " + str(round(r2_score(y,y_pred),3)))
```

output:

Read the given dataset:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	TAX	PTRATIO	...	MEDV	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	24.0
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	296.0	15.3	...	24.0	1	0	0	0	0	0	0	0	0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	242.0	17.8	...	21.6	0	1	0	0	0	0	0	0	0
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	242.0	17.8	...	34.7	0	1	0	0	0	0	0	0	0
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	222.0	18.7	...	33.4	0	0	1	0	0	0	0	0	0
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	222.0	18.7	...	36.2	0	0	1	0	0	0	0	0	0
5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	222.0	18.7	...	28.7	0	0	1	0	0	0	0	0	0
6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	311.0	15.2	...	22.9	0	0	0	0	1	0	0	0	0
7	0.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	311.0	15.2	...	27.1	0	0	0	0	1	0	0	0	0
8	0.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	311.0	15.2	...	16.5	0	0	0	0	1	0	0	0	0
9	0.17004	12.5	7.87	0.0	0.524	6.004	85.9	6.5921	311.0	15.2	...	18.9	0	0	0	0	1	0	0	0	0

10 rows × 22 columns

Finding of Errors:

```
cv = KFold(n_splits=10, random_state=None, shuffle=False)
classifier_pipeline = make_pipeline(StandardScaler(), KNeighborsRegressor(n_neighbors=10))
y_pred = cross_val_predict(classifier_pipeline, X, y, cv=cv)
print("RMSE: " + str(round(sqrt(mean_squared_error(y,y_pred)),2)))
print("R_squared: " + str(round(r2_score(y,y_pred),2)))
```

```
RMSE: 6.51
R_squared: 0.5
```

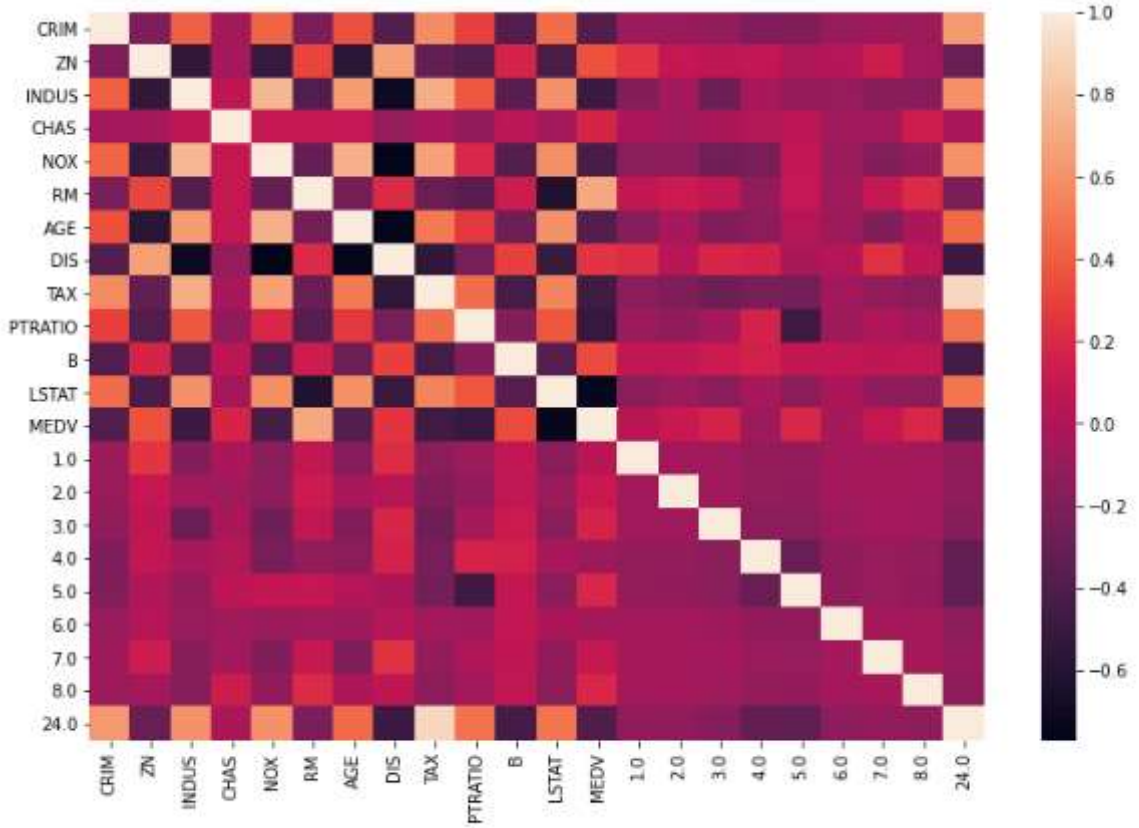
```
X = X.drop(columns = ['NOX','CHAS'])
y_pred = cross_val_predict(classifier_pipeline, X, y, cv=cv)
print("RMSE: " + str(round(sqrt(mean_squared_error(y,y_pred)),2)))
print("R_squared: " + str(round(r2_score(y,y_pred),2)))
```

```
RMSE: 6.28
R_squared: 0.53
```

```
boston.var()
```

```
CRIM          73.986578
ZN           543.936814
INDUS         47.064442
CHAS           0.064513
NOX            0.013428
RM             0.493671
AGE          792.358399
DIS            4.434015
TAX        28404.759488
PTRATIO        4.686989
B           8334.752263
LSTAT         50.994760
MEDV          84.586724
1.0            0.038039
2.0            0.045271
3.0            0.069597
4.0            0.170469
5.0            0.175968
6.0            0.048840
7.0            0.032532
8.0            0.045271
24.0           0.193198
dtype: float64
```

Filter Features by Correlation :2




```
['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'TAX', 'PTRATIO', 'B', 'LSTAT', 2.0, 3.0, 5.0, 8.0, 24.0]
RMSE: 6.47
R_squared: 0.5
['CRIM', 'ZN', 'INDUS', 'NOX', 'RM', 'AGE', 'DIS', 'TAX', 'PTRATIO', 'B', 'LSTAT', 24.0]
RMSE: 5.27
R_squared: 0.67
['CRIM', 'ZN', 'INDUS', 'NOX', 'RM', 'AGE', 'TAX', 'PTRATIO', 'B', 'LSTAT', 24.0]
RMSE: 5.42
R_squared: 0.65
['INDUS', 'NOX', 'RM', 'TAX', 'PTRATIO', 'LSTAT']
RMSE: 4.89
R_squared: 0.72
['RM', 'PTRATIO', 'LSTAT']
RMSE: 4.73
R_squared: 0.74
['RM', 'LSTAT']
RMSE: 4.8
R_squared: 0.73
['LSTAT']
RMSE: 5.7
```

```
R_squared: 0.61
```

Feature Selection Using a Wrapper:4

```
X = boston.drop(columns='MEDV')
sfs1.fit(X,y)
sfs1.subsets_
```

```
{21: {'feature_idx': (0,
   1,
   2,
   3,
   4,
   5,
   6,
   7,
   8,
   9,
   10,
   11,
   12,
   13,
   14,
   15,
   16,
   17,
   18,
```

```
X = boston.drop(columns='MEDV')[['CRIM','RM','PTRATIO','LSTAT']]
y = boston['MEDV']
y_pred = cross_val_predict(classifier_pipeline, X, y, cv=cv)
print("RMSE: " + str(round(sqrt(mean_squared_error(y,y_pred)),3)))
print("R_squared: " + str(round(r2_score(y,y_pred),3)))
```

```
RMSE: 4.458
R_squared: 0.765
```

```
boston[['CRIM','RM','PTRATIO','LSTAT','MEDV']].corr()
```

	CRIM	RM	PTRATIO	LSTAT	MEDV
CRIM	1.000000	-0.219247	0.289946	0.455621	-0.388305
RM	-0.219247	1.000000	-0.355501	-0.613808	0.695360
PTRATIO	0.289946	-0.355501	1.000000	0.374044	-0.507787
LSTAT	0.455621	-0.613808	0.374044	1.000000	-0.737663
MEDV	-0.388305	0.695360	-0.507787	-0.737663	1.000000

```
X = boston.drop(columns='MEDV')[['CRIM','RM','PTRATIO','LSTAT']]
y = boston['MEDV']
y_pred = cross_val_predict(classifier_pipeline, X, y, cv=cv)
print("RMSE: " + str(round(sqrt(mean_squared_error(y,y_pred)),3)))
print("R_squared: " + str(round(r2_score(y,y_pred),3)))
```

```
RMSE: 4.458
R_squared: 0.765
```

```
X = boston.drop(columns='MEDV')[['CRIM','RM','PTRATIO','LSTAT','RM*LSTAT']]
y = boston['MEDV']
y_pred = cross_val_predict(classifier_pipeline, X, y, cv=cv)
print("RMSE: " + str(round(sqrt(mean_squared_error(y,y_pred)),3)))
print("R_squared: " + str(round(r2_score(y,y_pred),3)))
```

```
RMSE: 3.509
R_squared: 0.801
```

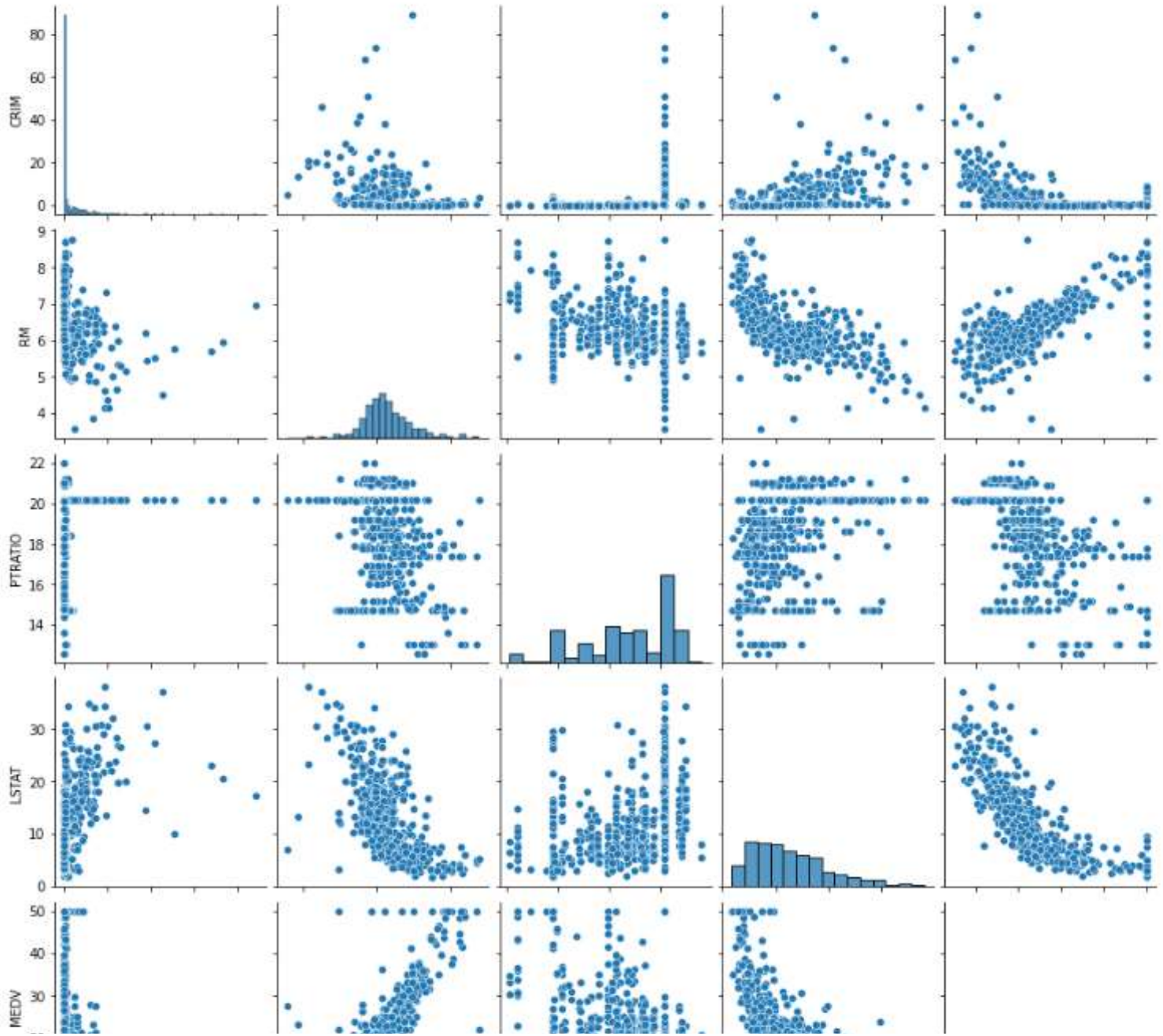
```
X = boston.drop(columns='MEDV')[['CRIM','RM','PTRATIO','LSTAT']]
y_pred = cross_val_predict(classifier_pipeline, X, y, cv=cv)
print("RMSE: " + str(round(sqrt(mean_squared_error(y,y_pred)),3)))
print("R_squared: " + str(round(r2_score(y,y_pred),3)))
```

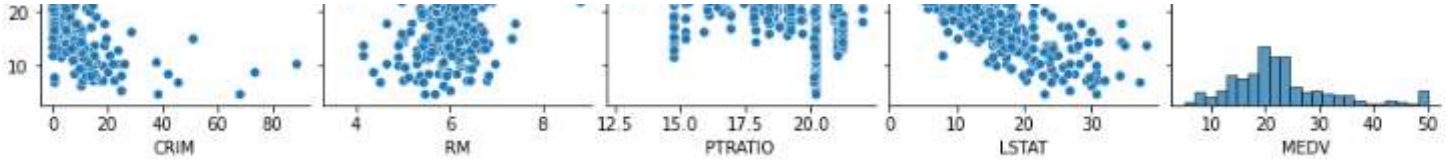
```
RMSE: 3.548
R_squared: 0.796
```

Pair ploat:

```
sn.pairplot(boston[['CRIM','RM','PTRATIO','LSTAT','MEDV']])
```

```
<seaborn.axisgrid.PairGrid at 0x14b3ffe2f10>
```





Result;

the various feature selection techniques has been performed on a dataset and saved the data to a file