2000080110_ML Skill5

September 2, 2021

[40]:
```
#multiple linear regression-bangaluruhouse data
import sklearn
from sklearn.metrics import mean_squared_error as MSE
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
d=pd.read_csv(r'E:\M&L excel\Bengaluru_House_Data.csv')
d.dropna(inplace=True)
#d.drop(['society','balcony'],axis=1,inplace=True)
print(d.isna().sum())
d.info()
```

```
area_type       0
availability    0
location        0
size            0
society         0
total_sqft      0
bath            0
balcony         0
price           0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7496 entries, 0 to 13318
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   area_type     7496 non-null   object
 1   availability  7496 non-null   object
 2   location      7496 non-null   object
 3   size          7496 non-null   object
 4   society       7496 non-null   object
 5   total_sqft    7496 non-null   object
 6   bath          7496 non-null   float64
 7   balcony       7496 non-null   float64
 8   price         7496 non-null   float64
dtypes: float64(3), object(6)
```

```
memory usage: 585.6+ KB
```

[41]:
```
le=LabelEncoder()
d['society']=le.fit_transform(d['society'])
d['location']=le.fit_transform(d['location'])
d['size']=le.fit_transform(d['size'])
d['total_sqft']=le.fit_transform(d['total_sqft'])
d['availability']=le.fit_transform(d['availability'])
d['area_type']=le.fit_transform(d['area_type'])
d = d.astype({"area_type":'float64', "availability":'float64',"location":
 ↪'float64',"size":'float64',"society":'float64',"total_sqft":'float64'})
print(d.head())
#d.drop('bath',axis=1,inplace=True)
d.info()
```

```
    area_type  availability  location  size  society  total_sqft  bath  \
0         3.0          35.0     210.0   3.0    443.0        63.0   2.0
1         2.0          73.0     149.0   8.0   2353.0      1128.0   5.0
3         3.0          73.0     387.0   5.0   2109.0       551.0   3.0
5         3.0          73.0     625.0   3.0    585.0       192.0   2.0
11        2.0          73.0     625.0   8.0   1566.0      1163.0   5.0

    balcony   price
0       1.0   39.07
1       3.0  120.00
3       1.0   95.00
5       1.0   38.00
11      3.0  295.00
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7496 entries, 0 to 13318
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   area_type     7496 non-null   float64
 1   availability  7496 non-null   float64
 2   location      7496 non-null   float64
 3   size          7496 non-null   float64
 4   society       7496 non-null   float64
 5   total_sqft    7496 non-null   float64
 6   bath          7496 non-null   float64
 7   balcony       7496 non-null   float64
 8   price         7496 non-null   float64
dtypes: float64(9)
memory usage: 585.6 KB
```

[42]:
```
from sklearn.metrics import mean_squared_error as MSE
#from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

```
from sklearn.ensemble import RandomForestRegressor
X=d.drop(['price'],axis=1)
Y=d['price']
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.2)
x_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5996 entries, 6448 to 10210
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   area_type     5996 non-null   float64
 1   availability  5996 non-null   float64
 2   location      5996 non-null   float64
 3   size          5996 non-null   float64
 4   society       5996 non-null   float64
 5   total_sqft    5996 non-null   float64
 6   bath          5996 non-null   float64
 7   balcony       5996 non-null   float64
dtypes: float64(8)
memory usage: 421.6 KB
```

[43]:
```
from sklearn.linear_model import LinearRegression
# creating an object of LinearRegression class
LR = LinearRegression()
# fitting the training data
LR.fit(x_train,y_train)
y_prediction =  LR.predict(x_test)
```

[44]:
```
# importing r2_score module
from sklearn.metrics import r2_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_squared_error
# predicting the accuracy score
score=r2_score(y_test,y_prediction)
print('r2 socre is ',score)
print('mean_sqrd_error is==',mean_squared_error(y_test,y_prediction))
print('root_mean_squared error of is==',np.
 ↪sqrt(mean_squared_error(y_test,y_prediction)))
```

```
r2 socre is  0.4936803957135154
mean_sqrd_error is== 3968.673226167664
root_mean_squared error of is== 62.99740650350349
```

[45]:
```
#boston house data
d=pd.read_csv(r'E:\M&L excel\Boston.csv')
d.drop('Unnamed: 0',axis=1,inplace=True)
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64  
 4   nox      506 non-null    float64
 5   rm       506 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64  
 9   tax      506 non-null    int64  
 10  ptratio  506 non-null    float64
 11  black    506 non-null    float64
 12  lstat    506 non-null    float64
 13  medv     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

[46]:
```
y=d['medv']
x=d.drop('medv',axis=1)
x.head()
```

[46]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	\
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	

	black	lstat
0	396.90	4.98
1	396.90	9.14
2	392.83	4.03
3	394.63	2.94
4	396.90	5.33

[47]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

[48]:
```
from sklearn.linear_model import LinearRegression
# creating an object of LinearRegression class
LR = LinearRegression()
# fitting the training data
LR.fit(x_train,y_train)
test_prediction =  LR.predict(x_test)
```

```
train_pred=LR.predict(x_train)
score=r2_score(y_test,test_prediction)
score1=r2_score(y_train,train_pred)
print('testing r2 socre for LinearRegression is ',score)
print('training r2 socre for LinearRegression is ',score1)
```

```
testing r2 socre for LinearRegression is  0.7274691495085868
training r2 socre for LinearRegression is  0.7400581317602286
```

[49]:
```
from sklearn.ensemble import RandomForestRegressor
dt = RandomForestRegressor()
dt.fit(x_train,y_train)
y_predicted = dt.predict(x_test)
accuracy = dt.score(x_test,y_test)
print("Training Accuracy:",dt.score(x_train,y_train))
print("Testing Accuracy:",accuracy)
```

```
Training Accuracy: 0.9816949712664925
Testing Accuracy: 0.8721654863502171
```

[50]:
```
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
def find_bestmodel_using_gridsearchcv(X,y):
    algos = {
        'linear_regression' : {
            'model': LinearRegression(),
            'params': {
                'normalize': [True, False]
            }
        },
        'lasso': {
            'model': Lasso(),
            'params': {
                'alpha': [1,2],
                'selection': ['random', 'cyclic']
            }
        },
        'decision_tree': {
            'model': DecisionTreeRegressor(),
            'params': {
                'criterion' : ['mse','friedman_mse'],
```

```
                'splitter': ['best','random']
            }
        },
        'randomforest': {
            'model': RandomForestRegressor(),
            'params': {'n_estimators':[100]}
        }
    }
    scores = []
    cv = ShuffleSplit(n_splits=4, test_size=0.2, random_state=0)
    for algo_name, config in algos.items():
        gs =  GridSearchCV(config['model'], config['params'], cv=cv,␣
↪return_train_score=False)
        gs.fit(X,y)
        scores.append({
            'model': algo_name,
            'best_score': gs.best_score_,
            'best_params': gs.best_params_
        })

    return pd.DataFrame(scores,columns=['model','best_score','best_params'])

res=find_bestmodel_using_gridsearchcv(x_train,y_train)
res
```

[50]:

	model	best_score	best_params
0	linear_regression	0.766070	{'normalize': True}
1	lasso	0.698768	{'alpha': 1, 'selection': 'random'}
2	decision_tree	0.766591	{'criterion': 'friedman_mse', 'splitter': 'best'}
3	randomforest	0.868753	{'n_estimators': 100}

[51]:
```
#weigthloss
data=pd.read_csv(r'E:\M&L excel\diet_data.csv')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 151 entries, 0 to 150
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Date         150 non-null    object
 1   Stone        142 non-null    float64
```

```
 2   Pounds       142 non-null    float64
 3   Ounces       142 non-null    float64
 4   weight_oz    149 non-null    float64
 5   calories     140 non-null    float64
 6   cals_per_oz  147 non-null    object
 7   five_donuts  140 non-null    float64
 8   walk         140 non-null    float64
 9   run          140 non-null    float64
 10  wine         140 non-null    float64
 11  prot         140 non-null    float64
 12  weight       140 non-null    float64
 13  change       147 non-null    float64
dtypes: float64(12), object(2)
memory usage: 16.6+ KB
```

[52]:
```
data.dropna(inplace=True)
data['cals_per_oz'] = data.cals_per_oz.astype('float')
data.drop('Date',axis=1,inplace=True)
data.isna().sum()
```

[52]:
```
Stone          0
Pounds         0
Ounces         0
weight_oz      0
calories       0
cals_per_oz    0
five_donuts    0
walk           0
run            0
wine           0
prot           0
weight         0
change         0
dtype: int64
```

[53]:
```
from sklearn.model_selection import train_test_split
X=data.drop('change',axis=1)
Y=data['change']
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.6)
```

[54]:
```
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
lr=LinearRegression()
ri=Ridge()
la=Lasso()
lr.fit(x_train,y_train)
ri.fit(x_train,y_train)
```

```
la.fit(x_train,y_train)
```

[54]:
```
Lasso()
```

[55]:
```
from sklearn.metrics import r2_score
from sklearn.metrics import accuracy_score
lrpred=lr.predict(x_test)
print("Accuracy of Testing in LinearRegression",r2_score(y_test,lrpred))
ripred=ri.predict(x_test)
print("Accuracy of Testing in RidgeRegression",r2_score(y_test,ripred))
lapred=la.predict(x_test)
print("Accuracy of Testing in LassoRegression",r2_score(y_test,lapred))
```

```
Accuracy of Testing in LinearRegression 0.5784892955343586
Accuracy of Testing in RidgeRegression 0.7241059503308078
Accuracy of Testing in LassoRegression 0.7497977612591469
```