ASSIGNMENT -2 MACHINE LEARNING

Team members:

Saranya R-19S031

Neeraja U-19S020

REGULARIZATION, FEATURE TRANSFORMATION USING PCA AND SIMPLE REGRESSION FOR A MULTIPLE REGRESSION MODEL.

DATASET:

Dataset link: https://www.kaggle.com/kashnitsky/mlcourse?select=hostel_factors.csv

Dataset description:

This hostel_factor dataset contains 12 columns such as hostel name and 10 factors affecting hostel like city, cleanliness, distance, facilities, location, security, staff, atmosphere, price, location where the values of these factors are normalized and renamed column into f1, f2…f10.Then the last rating column of different hostels. There are 18 records of different hostels.

DESCRIPTION:

Preprocess the dataset like finding the missing values, changing colums, finding y column and x data and split the data into x_train,y_train,x_test and y_test with test size of 20% and train data of 80% by importing train_test_split.

1)MULTIPLE REGRESSION:

- Multiple regression is a machine learning algorithm to predict a dependent variable with two or more predictors.
- Fit the LinearRegression class of the linear_model library from the scikit learn.using fit() method to fit our multiple regression model object (model) to training set.
- Predict the test results, printing the r2 score,mean squared error, root MSE,mean absolute error(MAE), model intercept and coefficients.

2)RIDGE REGRESSION

Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. In a case where observations are fewer than predictor variables, ridge regression is the most appropriate technique.

- Fit the Ridge class with alpha=0.5 of linear_model library from scikit learn. using fit() method to fit our regression model object (model1) to training set.
- Predict the test results, printing the r2 score, mean squared error, root MSE, mean absolute error(MAE).
- To fine tune the hyperparameter of alpha, we use GridSearchCV method to test values between 0.0 and 1.0 with a grid separation of 0.01.define search() with model2 as ridge.
- Predict the test results, printing the r2 score, mean squared error, root MSE, mean absolute error(MAE) for best fit alpha got using grid search.

3)PCA (PRINCIPAL COMPONENT ANALYSIS):

Principal Component Analysis (PCA) is an unsupervised, non-parametric statistical technique primarily used for dimensionality reduction in machine learning.

- Fit the PCA class with n_component=5 of decomposition library from scikit learn. using fit,transform(x) method to fit our regression model object (df) to training set and choose 5 columns value1 to value5 and create new dataframe.
- Print the pca.explained_variance ratio of reduced features and pca.components_
- Now fit the LinearRegression class of the linear_model library from the scikit learn.using fit() method to fit our multiple regression model object (model3) for new training set of reduced data.
- Predict the test results, printing the r2 score,mean squared error, root MSE,mean absolute error(MAE).

SOURCE CODE:

```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
```

```
#read the dataset.
data = pd.read_csv('hostel_factor.csv')
data.head()
```

	hostel	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	rating
0	hostel1	0.675000	0.100000	0.300000	0.875000	0.250000	0.425000	0.350000	0.725000	0.400000	0.275000	9.0
1	hostel2	0.500000	0.000000	0.058824	0.573529	0.117647	0.382353	0.000000	0.161765	0.308824	0.000000	8.3
2	hostel3	0.520833	0.041667	0.020833	0.666667	0.229167	0.437500	0.270833	0.250000	0.395833	0.270833	8.6
3	hostel4	0.692308	0.038462	0.038462	0.346154	0.076923	0.307692	0.500000	0.115385	0.153846	0.269231	7.6
4	hostel5	0.620690	0.000000	0.000000	0.517241	0.172414	0.344828	0.172414	0.379310	0.103448	0.310345	8.7

```
#find the data shape.
data.shape
```

```
(18, 12)
```

```
#.find missing values.
missing = data.isnull().sum()
missing
```

```
hostel    0
f1        0
f2        0
f3        0
f4        0
f5        0
f6        0
f7        0
f8        0
f9        0
f10       0
rating    0
dtype: int64
```

```
#dropping the hostel name.
data.drop(columns='hostel',inplace=True)
data.shape
```

```
(18, 11)
```

```
#y column by selecting the result data
y = pd.DataFrame(data['rating'])
```

```
#x column by removing y data
X = data
X.drop(columns='rating',inplace=True)
```

```
#simple regression predicting the rating as the output and with 11 inputs by using train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train,y_train)
```

```
LinearRegression()
```

```
#predicting x
pred = model.predict(X_test)
pred
```

```
array([[9.53728099],
       [8.83083241],
       [8.09256775],
       [9.20372261]])
```

Multiple regression results:

```
#predict y
y_test
```

	rating
1	8.3
6	8.4
8	8.9
10	7.0

```
print("r2 score of model:")
np.round(model.score(X_test,y_test),4)
```

```
r2 score of model:

-2.6674
```

```
from sklearn.metrics import mean_squared_error
print("RMSE OF REG MODEL: ")
np.round(mean_squared_error(y_test,pred,squared=False),4)
```

```
RMSE OF REG MODEL:

1.344
```

```
from sklearn.metrics import mean_squared_error
print("MSE OF REG MODEL: ")
np.round(mean_squared_error(y_test,pred),4)
```

```
MSE OF REG MODEL:

1.8062
```

```
from sklearn.metrics import mean_absolute_error
print("MAE OF REG MODEL: ")
np.round(mean_absolute_error(y_test,pred),4)
```

```
1.1698
```

```
model.intercept_
```

```
array([7.42077627])
```

```
model.coef_
```

```
array([[ 3.3667148 , -0.89509656,  1.61731777,  0.42317472, -5.31891411,
         1.2505617 , -1.65379495, -0.27828993,  0.93232524, -2.18290507]])
```

Ridge regression results with alpha 0.5:

```
# Implementation of Ridge Regression
from sklearn.linear_model import Ridge
model1 = Ridge(alpha=0.50)
model1.fit(X_train,y_train)
```

```
Ridge(alpha=0.5)
```

```
pred1 = model1.predict(X_test)
pred1
```

```
array([[8.20519863],
       [8.47277993],
       [8.48934836],
       [8.22181262]])
```

```
print("r2 score of ridge with alpha=0.5")
np.round(model1.score(X_test,y_test),4)
```

```
r2 score of ridge with alpha=0.5

0.1494
```

```
print("RMSE of ridge with alpha=0.5")
np.round(mean_squared_error(y_test,pred1,squared = False),4)
```

```
RMSE of ridge with alpha=0.5

0.6473
```

```
print("MSE of ridge with alpha=0.5")
np.round(mean_squared_error(y_test,pred1),4)
```

```
MSE of ridge with alpha=0.5

0.4189
```

```
print("MAE of ridge with alpha=0.5")
np.round(mean_absolute_error(y_test,pred1),4)
```

```
MAE of ridge with alpha=0.5

0.45
```

Ridge regression results with best fit alpha using GridSearchCv:

```
model2 = Ridge()
```

```
from sklearn.model_selection import GridSearchCV
from numpy import arange
grid = dict()
grid['alpha'] = arange(0, 1, 0.01)
# define search
search = GridSearchCV(model2, grid, scoring='neg_mean_absolute_error', n_jobs=-1)
results = search.fit(X_train, y_train)
```

```
results
```

```
GridSearchCV(estimator=Ridge(), n_jobs=-1,
             param_grid={'alpha': array([0.  , 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ,
       0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 , 0.21,
       0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 , 0.31, 0.32,
       0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 , 0.41, 0.42, 0.43,
       0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 , 0.51, 0.52, 0.53, 0.54,
       0.55, 0.56, 0.57, 0.58, 0.59, 0.6 , 0.61, 0.62, 0.63, 0.64, 0.65,
       0.66, 0.67, 0.68, 0.69, 0.7 , 0.71, 0.72, 0.73, 0.74, 0.75, 0.76,
       0.77, 0.78, 0.79, 0.8 , 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87,
       0.88, 0.89, 0.9 , 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98,
       0.99])},
             scoring='neg_mean_absolute_error')
```

```
print('MAE: %.3f' % results.best_score_)
print('Config: %s' % results.best_params_)
```

```
MAE: -0.723
Config: {'alpha': 0.99}
```

```
pred3 = search.predict(X_test)
```

```
print("RMSE of ridge with best fit alpha:")
np.round(mean_squared_error(y_test,pred3,squared = False),4)
```

```
RMSE of ridge with best fit alpha:

0.6709
```

```
print("r2 score of ridge with best fit alpha:")
np.round(search.score(X_test,y_test),4)
```

```
r2 score of ridge with best fit alpha:

-0.4491
```

```
print("MSE of ridge with best fit alpha:")
np.round(mean_squared_error(y_test,pred3),4)
```

```
MSE of ridge with best fit alpha:

0.4501
```

```
print("MAE of ridge with best fit alpha:")
np.round(mean_absolute_error(y_test,pred3),4)
```

```
MAE of ridge with best fit alpha:

0.4491
```

PCA results:

```
# implementation of Principal Component Analysis (PCA)
from sklearn.decomposition import PCA
pca = PCA(n_components=5)
df = pca.fit_transform(X)
data1 = pd.DataFrame(data = df,columns = ['value1','value2','value3','value4','value5'])
data1.head()
```

	value1	value2	value3	value4	value5
0	0.466985	-0.018261	0.081957	-0.221870	-0.237810
1	-0.253824	-0.198247	0.103438	0.021403	0.082932
2	-0.029601	-0.097736	-0.026067	-0.263728	0.043346
3	-0.289022	0.288935	-0.216703	-0.182448	-0.060536
4	-0.037661	0.138007	-0.030108	0.002654	0.063701

```
data1.shape
```

```
(18, 5)
```

```
pca.explained_variance_ratio_
```

```
array([0.37473996, 0.18441123, 0.14576779, 0.10018579, 0.07968089])
```

```
pca.components_
```

```
array([[ 1.47931477e-01, -3.63087930e-03,  9.62003167e-02,
         4.63124129e-01,  1.52420642e-01,  4.08348065e-01,
         1.27393931e-01,  7.19758421e-01, -2.53097324e-02,
         1.71646322e-01],
       [ 7.52098689e-01, -9.07281888e-02, -7.02723464e-02,
        -3.45596976e-01, -8.90794967e-02, -2.14280381e-01,
         2.36562803e-01,  8.54466201e-02, -2.65880622e-01,
         3.37540602e-01],
       [ 3.85774872e-01, -1.83661060e-01, -6.55025012e-02,
         4.80674520e-01,  2.34807820e-01, -5.02416703e-01,
        -4.19679054e-01, -8.68780028e-06,  1.48334278e-01,
        -2.76436193e-01],
       [-5.71657469e-02, -4.37055243e-02, -1.63518584e-01,
        -3.73765113e-01, -1.37238424e-01,  4.20190826e-02,
        -3.88886817e-01,  4.52342786e-01, -4.45457741e-01,
        -5.03491488e-01],
       [ 1.67102377e-01, -2.05166168e-01, -2.97460204e-01,
         2.76573582e-02,  2.61103077e-01,  5.79493462e-01,
        -4.47577028e-01, -3.80154079e-01, -1.96923623e-01,
         2.30493847e-01]])
```

After PCA analysis, fitting linearregression() model from sckit learn for reduced dimensions:

```
model3 = LinearRegression()
```

```
X_train1,X_test1,y_train1,y_test1 = train_test_split(data1, y, test_size=0.20, random_state=0)
```

```
model3.fit(X_train1,y_train1)
```

```
LinearRegression()
```

```
pred4 = model3.predict(X_test1)
pred4
```

```
array([[7.75682747],
       [8.58640723],
       [8.75382884],
       [7.73320619]])
```

```
y_test1
```

	rating
1	8.3
6	8.4
8	8.9
10	7.0

```
print("RMSE OF RIDGE WITH BEST FIT ALPHA:")
np.round(mean_squared_error(y_test1,pred4,squared = False),4)
```

```
RMSE OF RIDGE WITH BEST FIT ALPHA:

0.4714
```

```
print("R2 score OF RIDGE WITH BEST FIT ALPHA:")
np.round(model3.score(X_test1,y_test1),4)
```

```
R2 score OF RIDGE WITH BEST FIT ALPHA:

0.5489
```

```
print("MSE OF RIDGE WITH BEST FIT ALPHA:")
np.round(mean_squared_error(y_test,pred4),4)
```

```
MSE OF RIDGE WITH BEST FIT ALPHA:

0.2222
```

```
print("MAE OF RIDGE WITH BEST FIT ALPHA:")
np.round(mean_absolute_error(y_test,pred3),4)
```

```
MAE OF RIDGE WITH BEST FIT ALPHA:

0.4491
```

COMPARISON AND INFERENCE:

MODEL	Alpha	Accuracy	RMSE	MSE	MAE
Simple multiplRegression	0	-2.6674	1.344	1.8062	1.1698
Ridge Regression with alpha = 0.5	0.5	0.1494	0.6473	0.4189	0.450
Ridge Regression with best fit alpha	0.99	0.4491	0.6709	0.4501	0.4491
PCA	0	0.5489	0.4714	0.2222	0.4491

Hence for this Dataset we prefer the model with the ridge regression with the alpha = 0.99 and also pca along regression which gives accuracy of 0.5489.

RESULT:

Thus l2 regularization along regression, feature space transformation with pca along regression and simple regression are implemented and ridge with best fit alpha and regression after performing pca are chosen as best models for this hostel_factor dataset.