CS F320 - FOUNDATIONS OF DATA SCIENCE

Semester I, 2021-2022

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI

HYDERABAD CAMPUS

Yasovar Tammareddy	- 2019AAPS0226H
A P Karthikeya	- 2019AAPS0276H

Introduction:

Finding a linear relationship between a target and one or more variables is done using linear regression. Simple and multiple linear regression are the two types of linear regression. Here we use multiple linear regression as there are more than one variable.

Here in the data set we have 13 variables which can be used to predict the output. We used different preprocessing methods to clean and prepare the data for the line and we also used some feature selection algorithms to reduce the number of features in the regression. More about the dataset is given below.

Dataset:

The given dataset has 14 columns namely

-Bedrooms

-Bathrooms

-Sqft-living

- Sqft-lot

-floors

-waterfront

-view

-condition

-grade

-Sqft-above

-Sqft-basement

-Sqft-living

-Sqft-lot15

-Price

We have a total of 1188 examples. After Preprocessing we get 1164 examples out of which we take 815 examples for training and 349 for testing. Our job is to predict the Price using the 13 features mentioned above.

Data Pre-Processing:

The following preprocessing techniques have been used to clean and preprocess the data.

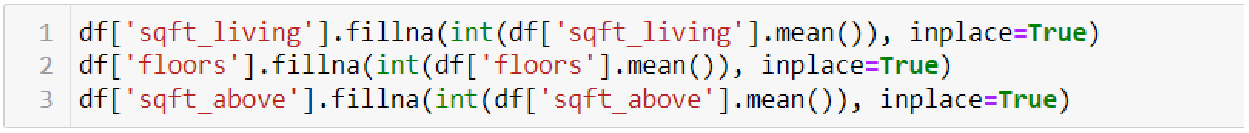
1. Standardization:

First we standardized the data by columns. This is a scaling technique which is helpful in dealing with high valued data as the data gets centered around zero with mean and unit variance.

$$X' = \frac{X - \mu}{\sigma}$$

2. Dealing with missing values:

After standardizing we found some missing values and we took care of them by replacing them with the mean of the column.



```
df['sqft_living'].fillna(int(df['sqft_living'].mean()), inplace=True)
df['floors'].fillna(int(df['floors'].mean()), inplace=True)
df['sqft_above'].fillna(int(df['sqft_above'].mean()), inplace=True)
```

3. Detecting outliers:

After dealing with the missing values we deal with the outliers. Machine learning algorithms' training processes can be distorted and misled by outliers in input data, resulting in longer training times, less accurate models, and ultimately inferior results.

We used Inter-quantile range to deal with the outliers.

```
Q1 = df['price'].quantile(0.25)
Q3 = df['price'].quantile(0.75)
IQR = Q3 - Q1

filter = (df['price'] >= Q1 - 3 * IQR) & (df['price'] <= Q3 + 3 *IQR)
df=df.loc[filter]
```

4. Data shuffling and splitting:

   We shuffle the data and split the data in the ratio of 70:30 which is used for training the model and testing the model respectively.

```
df_train=df.sample(frac=0.7)
df_test=df.drop(df_train.index)
df_train
```

Model and algorithms:

We have thirteen features related to various parameters related to housing to predict the price of each house. We are using Linear regression with gradient descent to predict the output price. We use both Greedy Forward feature selection and Greedy backward feature selection to find out optimal features that help to predict the output. After getting the model for predicting the output we run the test data we obtained by 70:30 split and compare both training and testing errors.

1. Error function:

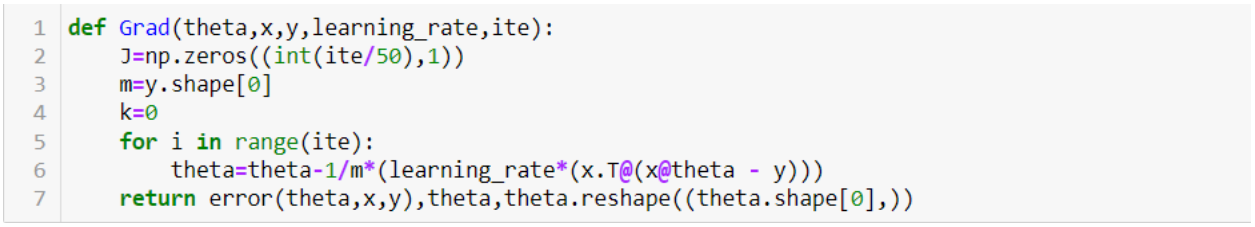
The error function we used to calculate the errors is given below:

Here we are calculating the half of mean squared error with respect to our predictions. Here ‘theta’ is the parameters matrix, ‘x’ is the input matrix,y is the expected output matrix.

```
def error(theta,x,y):
    E=1/2 *(np.sum((x@theta-y)**2))/y.shape[0]
    return E
```

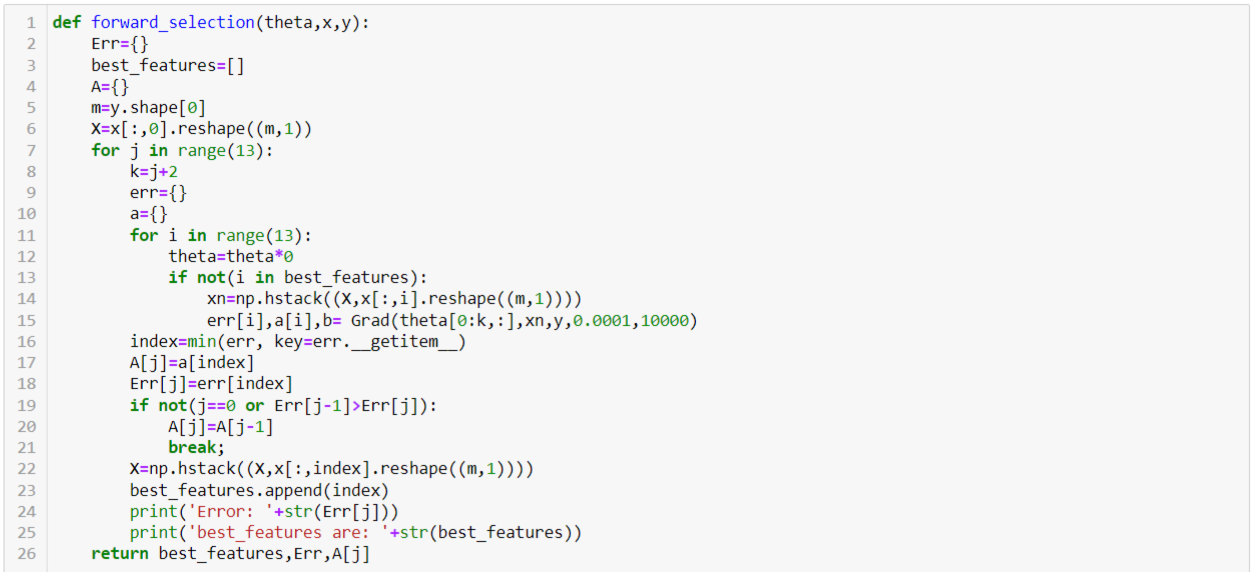
2. Gradient descent:

We minimize the error function by updating the parameters after every iteration as shown below. In this we take all the parameters in the partial derivation.



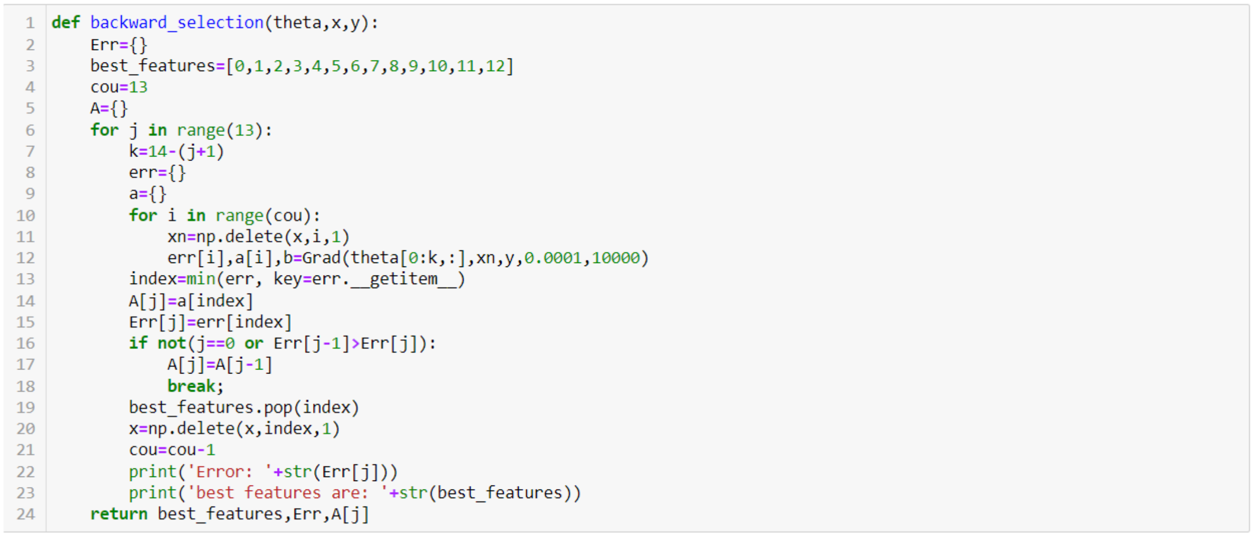
```python
def Grad(theta,x,y,learning_rate,ite):
    J=np.zeros((int(ite/50),1))
    m=y.shape[0]
    k=0
    for i in range(ite):
        theta=theta-1/m*(learning_rate*(x.T@(x@theta - y)))
    return error(theta,x,y),theta,theta.reshape((theta.shape[0],))
```

3. **Greedy forward feature selection:**
   This algorithm uses elimination to select the best features available for the algorithm. We start with 1 feature and select the best feature that gives the least error and then take two features with the above selected one and get the best two features and we continue until the error of the previous iteration is lesser than the current iteration.
   The same concept is coded below:



```python
def forward_selection(theta,x,y):
    Err={}
    best_features=[]
    A={}
    m=y.shape[0]
    X=x[:,0].reshape((m,1))
    for j in range(13):
        k=j+2
        err={}
        a={}
        for i in range(13):
            theta=theta*0
            if not(i in best_features):
                xn=np.hstack((X,x[:,i].reshape((m,1))))
                err[i],a[i],b= Grad(theta[0:k,:],xn,y,0.0001,10000)
        index=min(err, key=err.__getitem__)
        A[j]=a[index]
        Err[j]=err[index]
        if not(j==0 or Err[j-1]>Err[j]):
            A[j]=A[j-1]
            break;
        X=np.hstack((X,x[:,index].reshape((m,1))))
        best_features.append(index)
        print('Error: '+str(Err[j]))
        print('best_features are: '+str(best_features))
    return best_features,Err,A[j]
```

4. **Greedy backward feature selection:**
   This algorithm is similar to forward feature selection and uses elimination to select the best features. In this algorithm we start by leaving one feature and training the data. The model with least error is selected and the feature which is left in the model is eliminated and we continue the same for the remaining features.
   The code for the algorithm above is shown below:

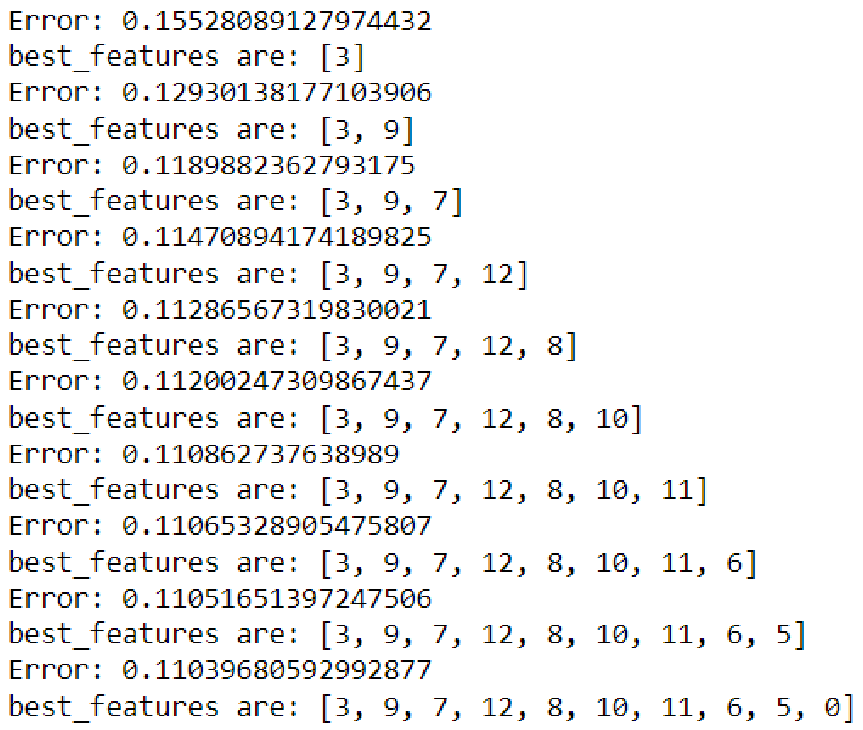


```python
def backward_selection(theta,x,y):
    Err={}
    best_features=[0,1,2,3,4,5,6,7,8,9,10,11,12]
    cou=13
    A={}
    for j in range(13):
        k=14-(j+1)
        err={}
        a={}
        for i in range(cou):
            xn=np.delete(x,i,1)
            err[i],a[i],b=Grad(theta[0:k,:],xn,y,0.0001,10000)
        index=min(err, key=err.__getitem__)
        A[j]=a[index]
        Err[j]=err[index]
        if not(j==0 or Err[j-1]>Err[j]):
            A[j]=A[j-1]
            break;
        best_features.pop(index)
        x=np.delete(x,index,1)
        cou=cou-1
        print('Error: '+str(Err[j]))
        print('best features are: '+str(best_features))
    return best_features,Err,A[j]
```

Outputs obtained:

We printed the output after each iteration in the forward and backward feature selection methods. The outputs are given as follows:

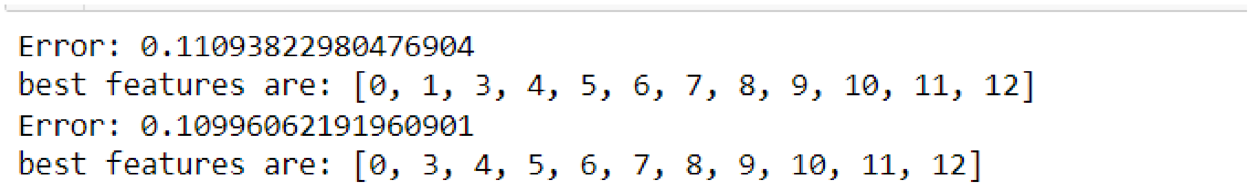
For forward feature selection:



```
Error: 0.15528089127974432
best_features are: [3]
Error: 0.12930138177103906
best_features are: [3, 9]
Error: 0.1189882362793175
best_features are: [3, 9, 7]
Error: 0.11470894174189825
best_features are: [3, 9, 7, 12]
Error: 0.11286567319830021
best_features are: [3, 9, 7, 12, 8]
Error: 0.11200247309867437
best_features are: [3, 9, 7, 12, 8, 10]
Error: 0.110862737638989
best_features are: [3, 9, 7, 12, 8, 10, 11]
Error: 0.11065328905475807
best_features are: [3, 9, 7, 12, 8, 10, 11, 6]
Error: 0.11051651397247506
best_features are: [3, 9, 7, 12, 8, 10, 11, 6, 5]
Error: 0.11039680592992877
best_features are: [3, 9, 7, 12, 8, 10, 11, 6, 5, 0]
```

Here 0 corresponds to the first feature and 12 corresponds to the 13th feature similarly. The algorithm gave us 10 best features and eliminated 3 features. The eliminated features are Bathrooms,sqft-living and waterfront.

For backward feature selection:



```
Error: 0.11093822980476904
best features are: [0, 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
Error: 0.10996062191960901
best features are: [0, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
```

The algorithm gave us 11 features and features and eliminated bathrooms and Sqft-living.

Comparison of Best Models:

Here non-preprocessed data contains the removed rows which have missing values because otherwise we would get no output and the output will be NaN.

Best model in forward feature selection had:

10 features which are -Bedrooms, sqft-lot, floors, view, condition, grade, Sqft-above, Sqft-basement, Sqft-living, Sqft-lot15

Best model in backward feature selection had:

11 features which are -Bedrooms, sqft-lot, floors, waterfront, view, condition, grade, Sqft-above, Sqft-basement, Sqft-living, Sqft-lot15

Comparison

The errors here are (MSE/2).

Model	Training Error	Testing Error
Greedy Forward Feature Selection	0.11039680592992877	0.15411505992023725
GreedyBackward Feature Selection	0.10996062191960901	0.10940507459251421
Gradient Descent on non-preprocessed Data with removed missing value rows and no feature selection	179893661197.60678	297616530487.4403
Gradient Descent on non-preprocessed Data and no feature selection	NaN	NaN