ROHAN NYATI

500075940

R177219148

BATCH – 5 (Ai & Ml)

Experiment -2

Ques.

EXPLAIN LASSO REGRESSION.WHAT IS REGULARIZATION?EXPLAIN THE COST FUNCTION IN LASSO REGRESSION.IMPLEMENT LASSO REGRESSION WITH A SUITABLE DATASET. EXPLAIN THE CODE IN COMMENTS. EXPLAIN LASSO REGRESSION.WHAT IS REGULARIZATION?EXPLAIN THE COST FUNCTION IN LASSO REGRESSION.IMPLEMENT LASSO REGRESSION WITH A SUITABLE DATASET. EXPLAIN THE CODE IN COMMENTS.

Lasso regression performs L1 regularization, which **adds a penalty equal to the absolute value of the magnitude of coefficients**. This type of regularization can result in sparse models with few coefficients; Some coefficients can become zero and eliminated from the model.

Lasso regression is a regularization technique. It is **used over regression methods for a more accurate prediction**. This model uses shrinkage. Shrinkage is where data values are shrunk towards a central point as the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters).

Regularization is a form which shrinks the coefficient towards zero or in other words Regularization is a technique which is used to minimize the errors by fitting the function over the training dataset and this helps in avoid overfitting. The types of regularization which are used commonly are L1 and L2 Regularization.

L1 regularization is called LASSO Regression.

L2 regularization is called RIDGE Regression.

The cost function for Lasso (least absolute shrinkage and selection operator) regression can be written as

$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \times x_{ij}\right)^2 + \lambda \sum_{j=0}^{p} |w_j| \qquad (1.4)$$

Cost function for Lasso regression

$$\text{For some } t > 0,\ \sum_{j=0}^{p} |w_j| < t$$

Implementation of LASSO Regression-

In [1]:
```
# Importing Necessary libraries
import numpy as np
import pandas as pd
from sklearn.linear_model import Ridge, Lasso
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn import model_selection
import matplotlib.pyplot as plt
from sklearn.linear_model import RidgeCV, LassoCV
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# loading the dataset through pandas
df = pd.read_csv("Hitters.csv")
df.head()
```

Out[2]:

	AtBat	Hits	HmRun	Runs	RBI	Walks	Years	CAtBat	CHits	CHmRun	CRuns	CRBI	CWalks	League	Division	PutOuts	Assists	Errors	Salary	NewLe
0	293	66	1	30	29	14	1	293	66	1	30	29	14	A	E	446	33	20	NaN	
1	315	81	7	24	38	39	14	3449	835	69	321	414	375	N	W	632	43	10	475.0	
2	479	130	18	66	72	76	3	1624	457	63	224	266	263	A	W	880	82	14	480.0	
3	496	141	20	65	78	37	11	5628	1575	225	828	838	354	N	E	200	11	3	500.0	
4	321	87	10	39	42	30	2	396	101	12	48	46	33	N	E	805	40	4	91.5	

In [3]:
```
# checking the datatype and size of each column
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 322 entries, 0 to 321
Data columns (total 20 columns):
AtBat        322 non-null int64
Hits         322 non-null int64
HmRun        322 non-null int64
Runs         322 non-null int64
RBI          322 non-null int64
Walks        322 non-null int64
Years        322 non-null int64
CAtBat       322 non-null int64
CHits        322 non-null int64
CHmRun       322 non-null int64
CRuns        322 non-null int64
CRBI         322 non-null int64
CWalks       322 non-null int64
League       322 non-null object
Division     322 non-null object
PutOuts      322 non-null int64
Assists      322 non-null int64
Errors       322 non-null int64
Salary       263 non-null float64
NewLeague    322 non-null object
dtypes: float64(1), int64(16), object(3)
memory usage: 50.4+ KB
```

In [4]:
```
# checking weather there is any NULL or NaN values in dataset
df.isnull().sum()
```
Out[4]:
```
AtBat         0
Hits          0
HmRun         0
Runs          0
RBI           0
Walks         0
Years         0
CAtBat        0
CHits         0
CHmRun        0
CRuns         0
CRBI          0
CWalks        0
League        0
Division      0
PutOuts       0
Assists       0
Errors        0
Salary       59
NewLeague     0
dtype: int64
```
In [5]:
```
# dropining all NaN values from dataframe
df = df.dropna()
```
In [6]:
```
# changing the Object values to integer values using dummies
dms = pd.get_dummies(df[['League', 'Division', 'NewLeague']])
dms.head()
```
Out[6]:

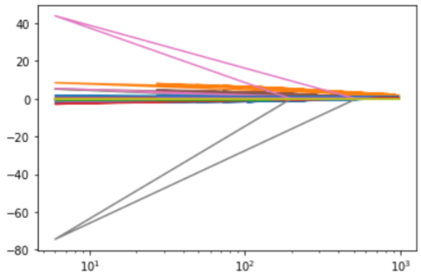
	League_A	League_N	Division_E	Division_W	NewLeague_A	NewLeague_N
1	0	1	0	1	0	1
2	1	0	0	1	1	0
3	0	1	1	0	0	1
4	0	1	1	0	0	1
5	1	0	0	1	1	0

In [7]:
```
# Seprating or storing Dependent(y) and independent features(x) in differnt variables
y = df["Salary"]
X_ = df.drop(['Salary', 'League', 'Division', 'NewLeague'], axis=1).astype('float64')
X = pd.concat([X_, dms[['League_N', 'Division_W', 'NewLeague_N']]], axis=1)
```
In [8]:
```
# Dividing the dataset into training and testing data
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=42)
```
In [9]:
```
# creating the lasso model and fitting the training data into the model
lasso_model = Lasso().fit(X_train,y_train)
```
In [10]:
```
# checking the intercept value of model
lasso_model.intercept_
```
Out[10]:
```
-5.587450677335255
```
In [11]:
```
# checking the coefficient values of each independent feature
lasso_model.coef_
```
Out[11]:
```
array([-1.74875691e+00,  8.59204135e+00,  6.67993798e+00, -3.06715333e+00,
       -1.91843070e+00,  5.32372890e+00,  8.39184117e+00, -1.63172447e-01,
       -8.22311277e-02, -3.93602861e-01,  1.71118530e+00,  6.55730545e-01,
       -6.48379405e-01,  2.59815358e-01,  2.73041157e-01, -4.41440454e-01,
        8.54474011e+01, -9.59701213e+01, -2.13086605e+01])
```

In [12]:
```
# plotting the cofficients values obtained on the gca plot using matplotlib
lasso = Lasso()
coefs = []
alphas = np.random.randint(0,1000,100)

for a in alphas:
  lasso.set_params(alpha = a)
  lasso.fit(X_train,y_train)
  coefs.append(lasso.coef_)
  ax = plt.gca()

ax.plot(alphas, coefs)
ax.set_xscale("log")
```



In [13]:
```
# predicting the 5 target values on the basis of training data
lasso_model.predict(X_train)[:5]
```

Out[13]:
```
array([377.26270596, 786.51524513, 495.14140718, 117.19492966,
       429.04228506])
```

In [14]:
```
# predicting the 5 target values on the basis of testing data
lasso_model.predict(X_test)[:5]
```

Out[14]:
```
array([ 609.18826367,  696.96810702, 1009.06157391,  412.22773375,
        409.25851712])
```

In [15]:
```
# storing all predicted values on the basis of test set in a cluster named y_pred
y_pred = lasso_model.predict(X_test)

# getting mean squared error by comparing and squaring y_test and y_pred values
np.sqrt(mean_squared_error(y_test,y_pred))
```

Out[15]:
```
356.09758845540324
```

In [16]:
```
# checking the accuracy of the model
r2_score(y_test,y_pred)
```

Out[16]:
```
0.414227981323662
```

In [17]:
```
# initializing the tuning model with some hyperparameter included(hyperparameter tuning) and fitting the same on training data
# crossvalidation is used for the hyperparameter tuning
lasso_cv_model = LassoCV(alphas = np.random.randint(0,1000,100), cv = 10, max_iter = 100000).fit(X_train,y_train)
```

In [18]:
```
# checking the value of alpha for the model initialized above
lasso_cv_model.alpha_
```

Out[18]:
```
194
```

In [19]:
```
# now applying the tuned hyperparameters to thr lasso model created at start and again fitting it on the training data
lasso_tuned = Lasso().set_params(alpha = lasso_cv_model.alpha_).fit(X_train,y_train)

# getting the tuned or changed values of coefficents for each column in the data
# and checking which column get neglected or have 0 value
pd.Series(lasso_tuned.coef_, index = X_train.columns )
```

Out[19]:
```
AtBat        -1.091837
Hits          5.443230
HmRun         0.000000
Runs          0.000000
RBI           0.000000
Walks         2.716615
Years         0.000000
CAtBat       -0.172062
CHits         0.239130
CHmRun       -0.000000
CRuns         1.049221
CRBI          0.470993
CWalks       -0.199767
PutOuts       0.272765
Assists       0.173924
Errors       -0.000000
League_N      0.000000
Division_W   -0.000000
NewLeague_N   0.000000
dtype: float64
```