Ex.no:9 **DHAMODARAN.B**

Date:31.09.2020 **1832016**

MODEL OPTIMIZATION

Problem Statement:

Breast Cancer is the most widespread cancer among the women. The mortality due to the breast cancer in females is the highest. 15% of deaths in females are due to the breast cancer. Detecting the breast cancer in early stages and giving the treatment can prevent the people from death. Thus, detecting the disease early becomes a mandatory. Fractal Dimension is the ratio of how a pattern changes accordingly with the scale at which it is being measured. Determining the fractal dimension is important as improper determination may lead to improper determination of the cancer severity. Here we aim at building a proper optimized model which can be used for predicting the fractal dimension. Model optimization can be achieved by optimizing the generalization error, bias-variance tradeoff and applying the cross validation and feature subset selection.

Problem Description:

The dependent variable or the target variable that we want to predict is ‘fractal dimension’ and the 30 independent variables are 'diagnosis', 'radius_mean', 'texture_mean', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness_mean', 'compactn ess_mean', 'concavity_mean','concave points_mean', 'symmetry_mean', 'fractal_dimension_me an','radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se','compactness_se', 'conc avity_se', 'concave points_se', 'symmetry_se','fractal_dimension_se', 'radius_worst', 'texture_w orst','perimeter_worst', 'area_worst', 'smoothness_worst','compactness_worst', 'concavity_wor st', 'concave points_worst','symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'.

Initially, the data is checked for null values and the ‘Unnamed: 32’ feature is dropped as it is full of missing values. The data are scaled using the Minmax scaling technique and later it was split into train and test in the ratio 85:15. Multiple Linear Regression model is fitted and later the model is optimized by optimizing the generalization error and the bias-variance tradeoff. Later, the model is further optimized with the help of cross validation and feature subset selecti on.

Code:

Data Preprocessing:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt
```

```
np.set_printoptions(suppress=True)
dataset = pd.read_csv('E:\College Education\sem 5\MPI\Breast Cancer Dataset.csv',index_col=0)
dataset.head(10)
print("Features : ",dataset.columns)
df = dataset.copy()
print("Missing Values \n",df.isnull().sum())
df = df.drop(columns=df.columns[-1])
df.head(10)
x = df.iloc[:,:-1].values
y = df['fractal_dimension_worst'].values.reshape(-1,1)
print("x shape",x.shape)
print("y shape :",y.shape)
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
binarizer = LabelBinarizer()
x[:,0] = binarizer.fit_transform(x[:,0]).flatten()
scaler = MinMaxScaler()
x = scaler.fit_transform(x.astype('float'))
y = scaler.fit_transform(y)
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.15,random_state=101)
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error as mse
```

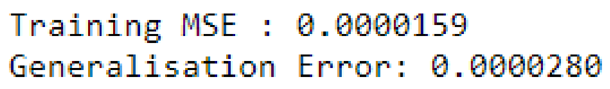
Model Fitting:

```
lin_reg = LinearRegression(n_jobs=-1)
lin_reg.fit(xtrain,ytrain)
ypred = scaler.inverse_transform(lin_reg.predict(xtest))
print("A Sample of Predictions: \n",ypred[:10])
```

GENERALIZTION ERROR:

```
train_mse = {'lin_reg':mse(scaler.inverse_transform(ytrain),
                    scaler.inverse_transform(lin_reg.predict(xtrain)))}
test_mse = {'lin_reg':mse(scaler.inverse_transform(ytest),ypred)}
print("Training MSE : {:.7f}".format(train_mse['lin_reg']))
print("Generalisation Error: {:.7f}".format(test_mse['lin_reg']))
```

Output:



```
Training MSE : 0.0000159
Generalisation Error: 0.0000280
```

Inference:

Generalization error is the error which happens at the time of predicting an unseen data. It is the measure of how well the model is able to generalize the unseen data. Our model has a very low generalization error of 0.00028 which explains that the model is very good in generalizing the unseen data. Thus, our model makes very good predictions on the unseen data.

BIAS VARIANCE TRADEOFF:

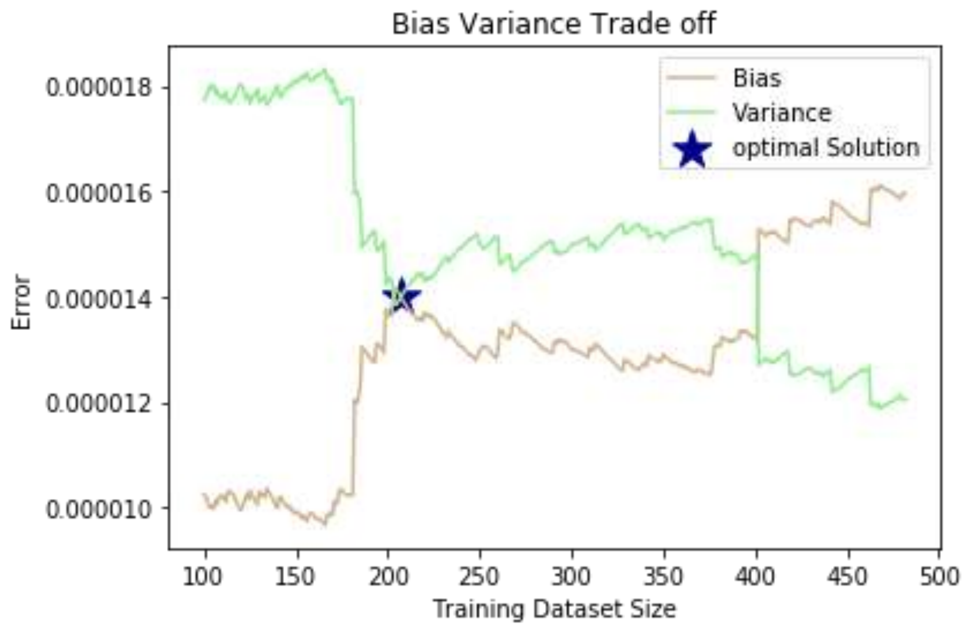
Code:

```
bias = []
variance = []
for i in range(100,len(xtrain)):
   xdata = xtrain[:i,:];ydata = ytrain[:i,:]
   model = LinearRegression(n_jobs=-1)
   model.fit(xdata,ydata)
   bias.append(mse(scaler.inverse_transform(ydata),
                    scaler.inverse_transform(lin_reg.predict(xdata))))
   variance.append(abs(bias[-1]-mse(scaler.inverse_transform(ytest),
                    scaler.inverse_transform(lin_reg.predict(xtest)))))
bias = pd.Series(data=bias,index=range(100,len(xtrain)))
variance = pd.Series(data=variance,index=range(100,len(xtrain)))
diff = pd.Series(data=abs((bias-variance)),index=range(100,len(xtrain)))
print("The Training datasize with lowest bias and variance is : {}".format(diff.idxmin()))
```

```
plt.plot(range(100,len(xtrain)),bias,label='Bias',color='tan')
plt.plot(range(100,len(xtrain)),variance,label='Variance',color='lightgreen')
plt.scatter(diff.idxmin(),bias[diff.idxmin()],marker='*',color='darkblue',s=300,label='optimal Solution')
plt.legend()
plt.title('Bias Varaince Trade off')
plt.ylabel('Error')
plt.xlabel('Training Dataset Size')
plt.show()
```

<u>Output:</u>

```
The Training datasize with lowest bias and variance is : 207
```



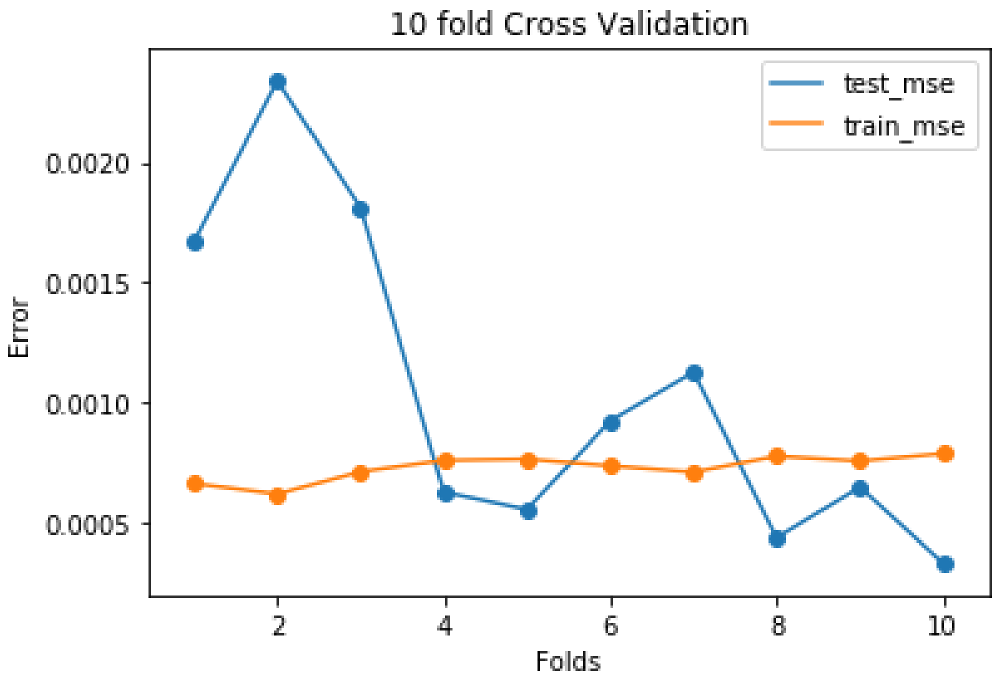
<u>Inference:</u>

Bias is the error due to the assumptions made by the model. Variance is the error due to the different datasets. The sum of bias and variance would be the reducible error in the model. There occurs a tradeoff between the bias and variance. Thus, choosing an optimum level of low bias and low variance is necessary. Studying the effect of training dataset size against the bias and variance, we can see that the choosing a very low dataset size and very high dataset size has resulted destructive to the model. Lower training dataset size has made the linear regression model overfit, resulting in higher variations. The larger training dataset size made the model too generalized resulting in underfitting and leading to high bias. The model has an ideal low bias and low variance when trained with a dataset of 207 samples as it makes the ideal assumptions here.

Cross validation:

```
from sklearn.model_selection import cross_validate

cv_results = cross_validate(lin_reg,x,y,cv=10,n_jobs=-
1,scoring='neg_mean_squared_error',return_train_score=True)

plt.plot(range(1,len(cv_results['test_score'])+1),abs(cv_results['test_score']),label='test_mse')

plt.scatter(range(1,len(cv_results['test_score'])+1),abs(cv_results['test_score']))

plt.plot(range(1,len(cv_results['train_score'])+1),abs(cv_results['train_score']),label='train_mse')

plt.scatter(range(1,len(cv_results['train_score'])+1),abs(cv_results['train_score']))

plt.title('10 fold Cross Validation')

plt.xlabel('Folds')

plt.ylabel('Error')

plt.legend()

plt.show()
```

Output:



```
Average Train error :  0.0007264427580041986
Average test error :  0.001045172790649945
```

Inference:

Cross validation splits the data into k-folds and trains the model with k-1 folds and tests with single fold. 10 fold (k=10) cross validation has been implemented here.Each fold will be in the training data for 9 times and each fold will be an test set for 1 time. The model performance is evaluated at each step. The average training error of the model is 0.00072 and the mean test error is 0.00104. Considering, the train and test error we can decide that the model performance is good on average. For 2nd fold, the test error is high, which indicates that the model didn't generalize well. For 10th fold, the test error is less, which indicates that the training data has contained all the range (or class) of values leading to a better fit which resulted in a low generalizing error.

Feature Subset Selection:

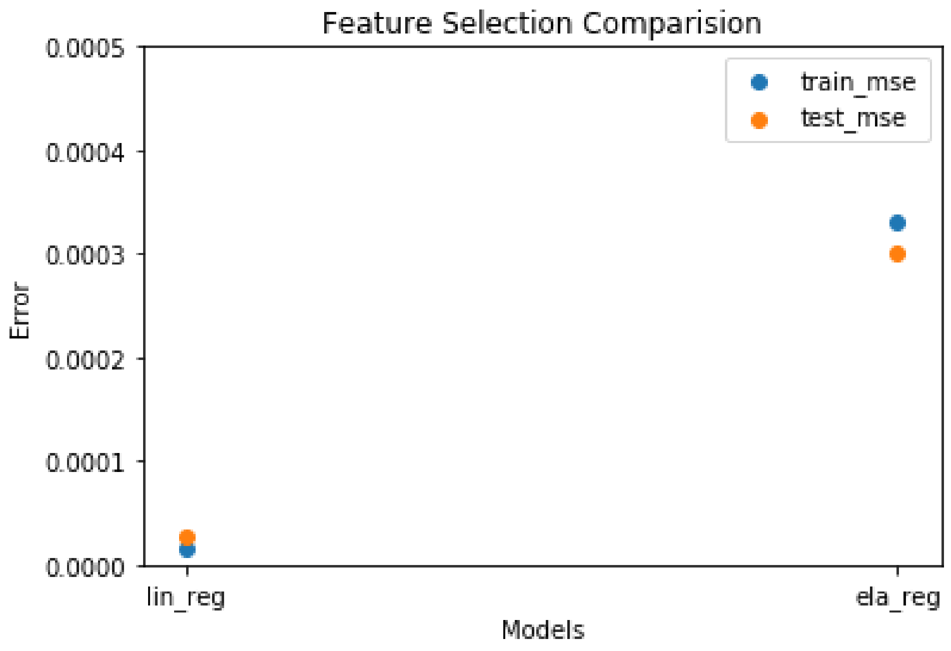
Code:

```
from sklearn.linear_model import ElasticNet

ela_reg = ElasticNet()

ela_reg.fit(xtrain,ytrain)

train_mse['ela_reg'] = mse(scaler.inverse_transform(ytrain),

                scaler.inverse_transform(ela_reg.predict(xtrain).reshape(-1,1)))

test_mse['ela_reg'] = mse(scaler.inverse_transform(ytest),

                scaler.inverse_transform(ela_reg.predict(xtest).reshape(-1,1)))

print("Train mse : ",train_mse['ela_reg'])

print("Test mse : ",test_mse['ela_reg'])

plt.scatter(train_mse.keys(),train_mse.values(),label='train_mse')

plt.scatter(test_mse.keys(),test_mse.values(),label='test_mse')

plt.ylim(0,0.0005)

plt.legend()

plt.title('Feature Selection Comparision')

plt.show()

print("Train_mse : ",train_mse)

print("Test_mse : ",test_mse)
```

Output:

```
ElasticNet(alpha=1.0, copy_X=True, fit_intercept=True, l1_ratio=0.5,
      max_iter=1000, normalize=False, positive=False, precompute=False,
      random_state=None, selection='cyclic', tol=0.0001, warm_start=False)
```

```
Train mse :  0.00033033873975112413
Test mse :  0.00030035971812840786
```



```
Train_mse :  {'lin_reg': 1.5910146955877265e-05, 'ela_reg': 0.00033033873975112413}
Test_mse :  {'lin_reg': 2.7985396662453613e-05, 'ela_reg': 0.00030035971812840786}
```

Inference:

Elastic net is one of the embedded methods to implement the feature selection. Elastic net models are naturally resistant to non-informative independent variables as it introduces the penalty. For the given dataset, the elastic net model is fitted and it removes the non-informative features by introducing penalty. Thus, the feature selection implemented model can now be used to make predictions. Even after feature selection using elastic net, we can see that the model performs well with training MSE of 0.00033 and a test MSE of 0.0003. Comparing the feature selected model (elastic net) and the normal linear regression, the normal linear regression model has a very slight better accuracy over the elastic net model. But in other criteria such as the amount of information required and the computational time, the elastic net model is the clear winner. Thus, the feature selection has made the model computations faster with lesser amount of information(features) along with a good accuracy in predictions too.

Conclusion:

The initially fitted multiple linear regression model had a very low generalization error of 0.00002 which explains that the model is also very well able capable of generalizing the unseen data. Studying the effect of bias and variance trade off against the training dataset size, lesser training dataset size made the model overfit leading to high variance; the larger dataset size had resulted in underfit leading to very high bias. The model has ideal low bias and low variance when trained with 207 samples, as it makes ideal assumptions and generalizes the model in sufficient levels. The 10-fold cross validation has shown that the model on average has a training MSE of 0.0007 and a mean test MSE of 0.001 which shows that the model is able to make very good predictions on different datasets too. Feature selection using elastic net has helped in removing the non-informative predictor variables. The elastic net model has a training MSE of 0.00033 and test MSE of 0.0003 which may be slightly lesser than the performance measure of the normal linear regression model with no feature selection. Because of the feature selection in elastic net, the amount of information required to make a prediction has become lesser and also the computational time has also become lesser making it better than the normal multiple linear regression model where the feature selection is not been implemented. On overall, the optimization techniques have helped in improvising the performance of the model.