Library Import

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
sns.set()
```

Multivariate Regression

```
class MultiVariateLinearRegression:
    def simpleMultiVariate(self,X,Y):
        X_T=np.transpose(X)
        X_inverse=np.linalg.inv(np.dot(X_T,X))
        W=np.matmul(np.matmul(X_inverse,X_T),Y)
        Y_Pred=self.predict(W,X)
        loss=self.MSE(Y,Y_Pred)
        return loss,W,Y_Pred
    def MSE(self,Y,Y_Pred):
        error=0
        for i in range(len(Y)):
            error+=(Y[i]-Y_Pred[i])**2
        return error/len(Y)
    def test(self,X,Y,W):
        test_pred=self.predict(W,X)
        test_loss=(self.MSE(Y,test_pred))
        return test_loss,test_pred
    def train(self,X,Y,X_Val,Y_Val,epoch,alpha):
        cols=X.shape[1]
        W=np.zeros(cols)
        train_loss=[]
        val_loss=[]
        epochs=[]
        m=X.shape[0]
        for j in range(epoch):
            Y_Pred=self.predict(W,X)
            cost=(self.MSE(Y,Y_Pred))/2
            dW=(1/m)*np.dot(X.T,(Y_Pred-Y))
            W=W-(alpha*dW)
            Y_Val_Pred=self.predict(W,X_Val)
            cost_val=(self.MSE(Y_Val_Pred,Y_Val))/2
            train_loss.append(cost)
            val_loss.append(cost_val)
            epochs.append(j)
        return W,train_loss,val_loss,epochs
    def testGD(self,X,Y,W):
        test_pred=self.predict(W,X)
        test_loss=(self.MSE(Y,test_pred))/2
```

```
        return test_loss,test_pred
    def predict(self,W,X):
        Y_Pred=W.dot(X.T)
        return Y_Pred

def rSquared(Y,Y_Pred):
    a_mean=[np.mean(Y)]*Y.shape[0]
    E12=(Y_Pred-Y).T.dot(Y_Pred-Y)
    E22=(Y-a_mean).T.dot(Y-a_mean)
    R2=1-(E12/E22)
    #print(a_mean[0],E12, E22)
    return R2
```

Loading Dataset

```
data = pd.read_csv('Datasets/LinearRegression/linear-regression.csv')

data.head()
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar
chlorides  \
0            7.4              0.70         0.00             1.9
0.076
1            7.8              0.88         0.00             2.6
0.098
2            7.8              0.76         0.04             2.3
0.092
3           11.2              0.28         0.56             1.9
0.075
4            7.4              0.70         0.00             1.9
0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates
\
0                 11.0                  34.0   0.9978  3.51       0.56

1                 25.0                  67.0   0.9968  3.20       0.68

2                 15.0                  54.0   0.9970  3.26       0.65

3                 17.0                  60.0   0.9980  3.16       0.58

4                 11.0                  34.0   0.9978  3.51       0.56


   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
```

```
3       9.8         6
4       9.4         5
```

Shuffling Data

```
data=data.sample(frac=1)

data.shape
```

```
(1599, 12)
```

Train Test Split

```
def normalize(X):
    for column in X:
        mean=np.mean(X[column])
        std=np.std(X[column])
        X[column]=(X[column]-mean)/std
    return X

x=normalize(data)

x=data.iloc[:,:11]
y=data.iloc[:,11]

X_Train,X_Test,Y_Train,Y_Test=train_test_split(x,y,test_size=0.2,rando
m_state=1)
X_Train,X_Val,Y_Train,Y_Val=train_test_split(X_Train,Y_Train,test_size
=0.375,random_state=1)

X_Train=X_Train.to_numpy()
X_Test=X_Test.to_numpy()
X_Val=X_Val.to_numpy()
Y_Train=Y_Train.to_numpy()
Y_Test=Y_Test.to_numpy()
Y_Val=Y_Val.to_numpy()

print(X_Train.shape)
print(X_Test.shape)
print(X_Val.shape)
print(Y_Train.shape)
print(Y_Test.shape)
print(Y_Val.shape)
```

```
(799, 11)
(320, 11)
(480, 11)
(799,)
(320,)
(480,)
```

Model Training

```
model = MultiVariateLinearRegression()

loss,W,Y_Pred= model.simpleMultiVariate(X_Train,Y_Train)

loss

0.6659040810754848

W

array([ 0.02365897, -0.27025515, -0.03378935, -0.00200769, -
0.10541507,
        0.08673425, -0.17258628, -0.00752654, -0.09511516,
0.17036384,
        0.35254638])

plt.plot(Y_Pred,Y_Train,'.')
plt.xlabel('Predicted_Training_Values')
plt.ylabel('Actual Training Values')
plt.title('Plot Loss')
plt.show()
```

```
mse_Test,Y_Test_pred=model.test(X_Test,Y_Test,W)

plt.plot(Y_Test_pred,Y_Test,'.')
plt.xlabel('Predicted_Test_Values')
plt.ylabel('Actual Test Values')
plt.title('Plot Loss')
plt.show()
```


```
R2=rSquared(Y_Test,Y_Test_pred)
RMSE=np.sqrt(mse_Test)
print("R-Squared Score: ",R2)
print("RMSE Score: ",RMSE)

R-Squared Score:  0.35665420880193754
RMSE Score:  0.8272368441535175
```

Gradient Descent

Learning Rate:0.01

```
model1=MultiVariateLinearRegression()
```

```
W,train_loss,val_loss,epochs=model1.train(X_Train,Y_Train,X_Val,Y_Val,
150,0.01)

plt.plot(epochs,train_loss,label='Training Loss')
plt.plot(epochs,val_loss,label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(loc='best')
plt.show()
```


```
mse_Test,Y_Test_pred=model.testGD(X_Test,Y_Test,W)
R2=rSquared(Y_Test,Y_Test_pred)
RMSE=np.sqrt(mse_Test)
print("R-Squared Score: ",R2)
print("RMSE Score: ",RMSE)

R-Squared Score:  0.34281536302023286
RMSE Score:  0.5912026069374193
```

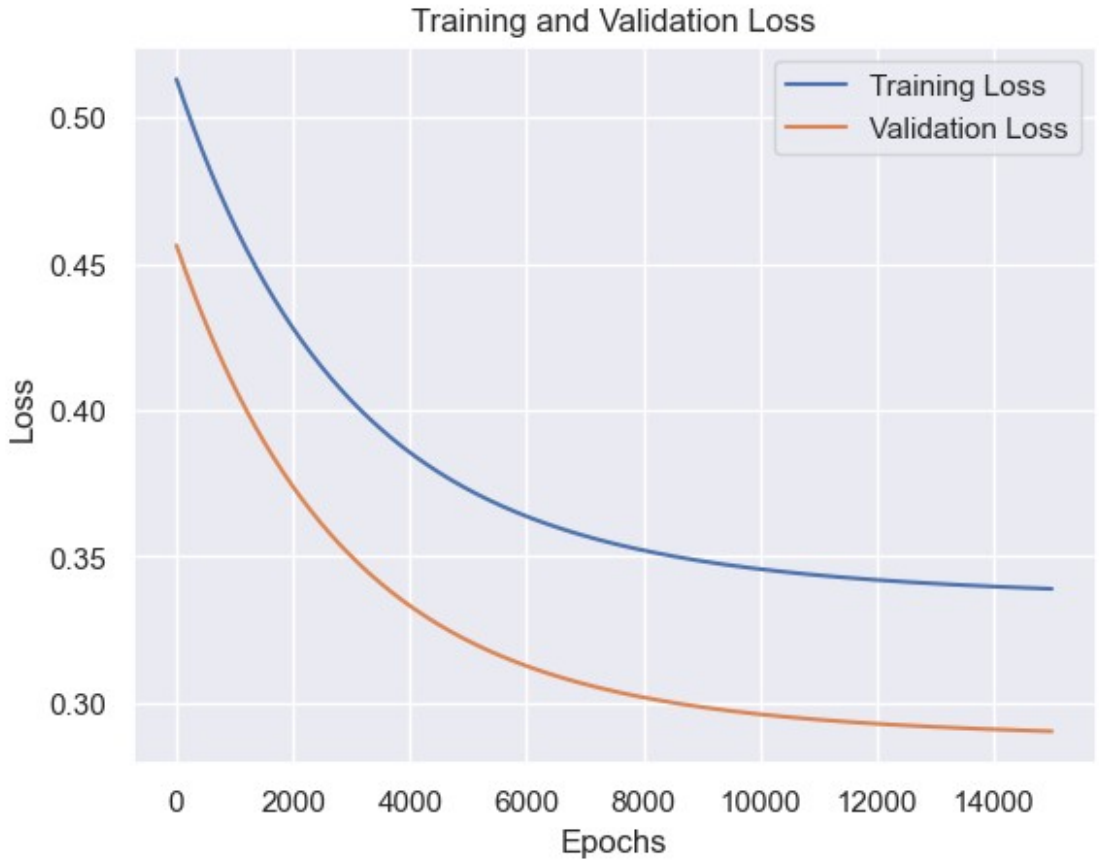
Learning Rate: 0.001

```
model2=MultiVariateLinearRegression()

W,train_loss,val_loss,epochs=model2.train(X_Train,Y_Train,X_Val,Y_Val,
1500,0.001)

plt.plot(epochs,train_loss,label='Training Loss')
plt.plot(epochs,val_loss,label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(loc='best')
plt.show()
```


```
mse_Test,Y_Test_pred=model2.testGD(X_Test,Y_Test,W)
R2=rSquared(Y_Test,Y_Test_pred)
RMSE=np.sqrt(mse_Test)
print("R-Squared Score: ",R2)
print("RMSE Score: ",RMSE)

R-Squared Score:  0.34267853065607967
RMSE Score:  0.5912641508593764
```

Learning Rate: 0.0001

```
model3=MultiVariateLinearRegression()

W,train_loss,val_loss,epochs=model3.train(X_Train,Y_Train,X_Val,Y_Val,
15000,0.0001)

plt.plot(epochs,train_loss,label='Training Loss')
plt.plot(epochs,val_loss,label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(loc='best')
plt.show()
```




```
mse_Test,Y_Test_pred=model3.testGD(X_Test,Y_Test,W)
R2=rSquared(Y_Test,Y_Test_pred)
RMSE=np.sqrt(mse_Test)
print("R-Squared Score: ",R2)
print("RMSE Score: ",RMSE)

R-Squared Score:  0.342664788477963
RMSE Score:  0.5912703314081547
```