CSE1015 – Machine Learning Essentials

Assessment – 1

Pinni Venkata Abhiram
20BAI1132

Assessment – 1

Pinni Venkata Abhiram

20BAI1132

Questions

1. Describe the dataset and visualize the data using suitable plots and write the inference.
2. Perform the exploratory data analytics with necessary explanation
3. Perform the necessary pre-processing steps and mention them clearly
4. Perform correlation analysis and visualize using necessary plots. Write the inference clearly.
5. Select the appropriate variables and justify your selection.
6. Choose the appropriate regression model to predict the target variable that yields best performance by using 80/20 split.
7. Compare all regression models (linear and polynomial) and tabulate the results in terms of MAE, MSE, R-square and Adjusted R-square. Also, plot the comparison graph (X-axis : Method names, Y-axis: R-square value)
8. Write the conclusion by mentioning the selected variable and the obtained equation(wherever applicable).

Introduction

Building a Linear Regression model and a Polynomial Regression model to predict the strain of a device given the values of Actuator , Load and Time.

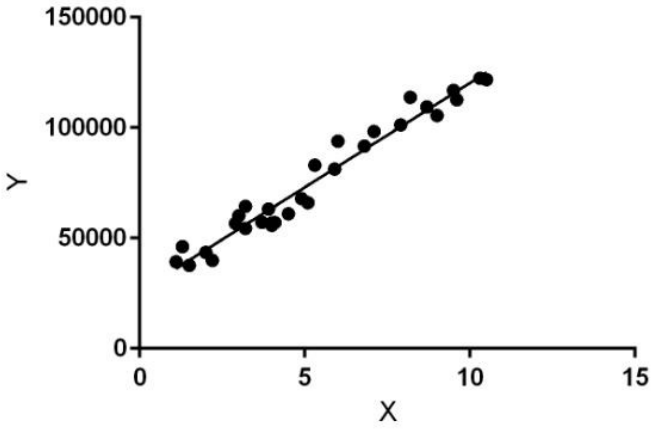
The fields in the data are – Actuator , Load , Time and Strain.

The model is built using sklearn linear regression module and polynomial features module for polynomial regression and various plots for correlation provided by seaborn module and statsmodels for statistical summary.

Methodology

The methodology used in here is the concept of Linear regression

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.



Polynomial regression is a form of Linear regression where only due to the

Non-linear relationship between dependent and independent variables we add some polynomial terms to linear regression to convert it into Polynomial regression.

Suppose we have X as Independent data and Y as dependent data. Before feeding data to a mode in preprocessing stage we convert the input variables into polynomial terms using some degree.

Consider an example my input value is 35 and the degree of a polynomial is 2 so I will find 35 power 0, 35 power 1, and 35 power 2 And this helps to interpret the non-linear relationship in data.

The equation of polynomial becomes something like this.

$$y = a_o + a_1x_1 + a_2{x_1}^2 + \ldots + a_n{x_1}^n$$

The degree of order which to use is a Hyperparameter, and we need to choose it wisely. But using a high degree of polynomial tries to overfit the data and for smaller values of degree, the model tries to underfit so we need to find the optimum value of a degree.

Dataset

The dataset used is a .csv file which contains the following rows

Load , Actuator , Time and Strain

We use Load , Actuator , Time for the development of the model and we predict value of the strain.

	A	B	C	D	E	F
1	Load	Actuator	Time	Strain		
2	-3.06825	0	0.09	1.01E-06		
3	-2.84884	0	0.157	1.73E-06		
4	-2.38585	0	0.223	8.11E-06		
5	-1.91418	0	0.29	1.21E-05		
6	-1.61167	0.006944	0.357	1.18E-05		
7	-1.30291	0.020833	0.423	1.81E-05		
8	-1.24391	0.027778	0.49	1.79E-05		
9	-1.07775	0.041667	0.557	1.83E-05		
10	-0.85787	0.048611	0.623	2.42E-05		
11	-0.67655	0.0625	0.69	2.49E-05		
12	-0.42816	0.076389	0.757	2.41E-05		
13	-0.09457	0.097222	0.823	3.13E-05		
14	0.09762	0.111111	0.89	3.32E-05		
15	0.382422	0.131944	0.957	3.39E-05		
16	0.716991	0.152778	1.023	4.04E-05		
17	0.950542	0.173611	1.09	4.30E-05		

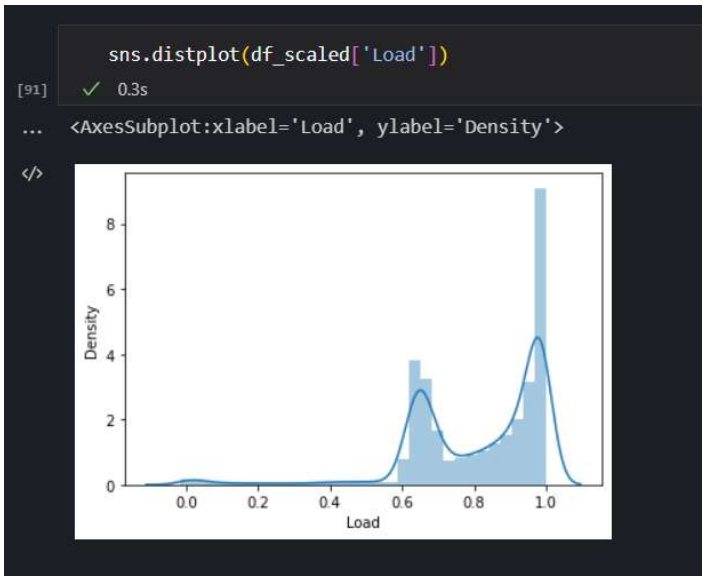
Experiments and Results

The experiment required numpy , pandas , seaborn , sklearn , statsmodels , matplotlib libraries for analysis and model building.

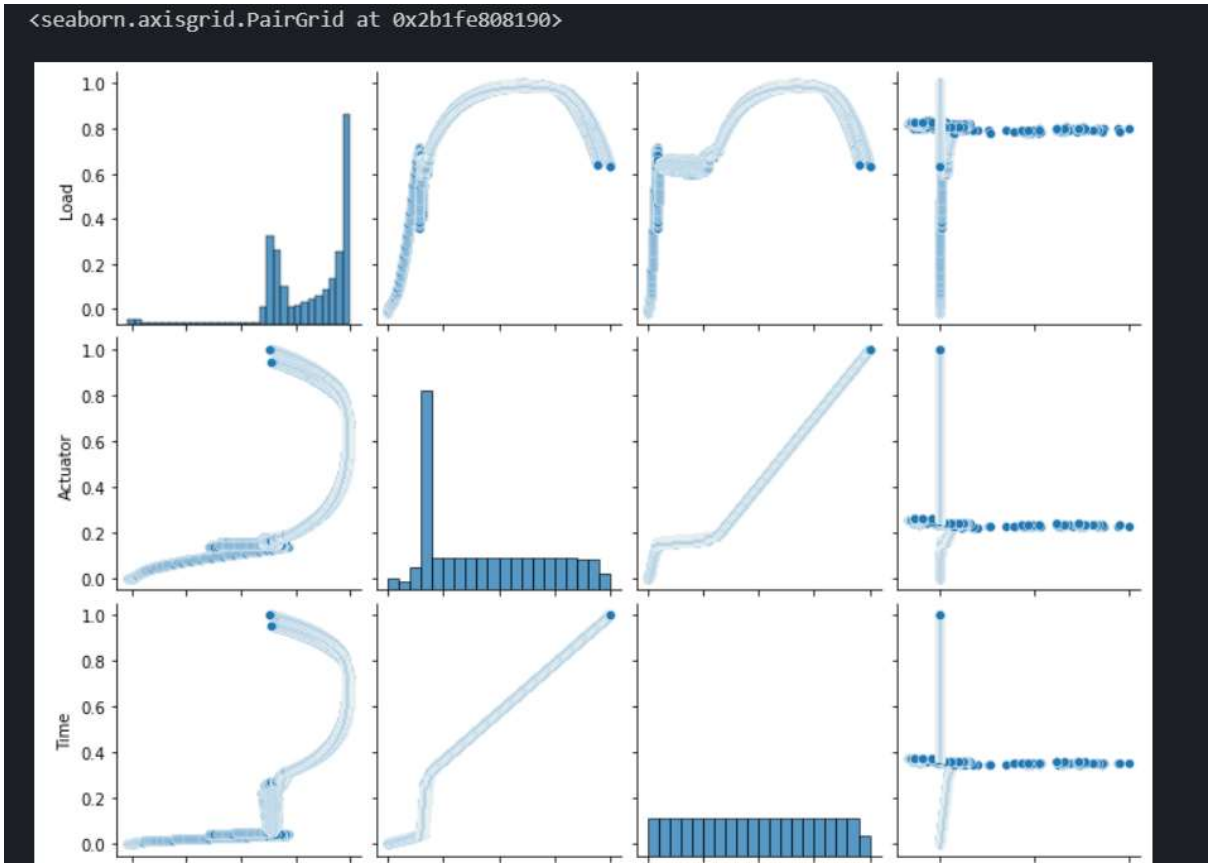
The data is imported using pandas then we make plots based on the data using seaborn for our better understanding

Before we move on to the visual analysis we first scale our data using maximum absolute scaling method because the database is big and we need to scale it for our model to work faster.

Displot



Pairplot



And many more plots.

We check if the data has any NaN values i.e. missing values and we try to remove them , this is the data cleaning process for both test dataset and the train dataset.

Test dataset

```
[98]  df2_scaled.isnull().sum()
      ✓ 0.1s

...   Load        0
      Actuator    0
      Time        0
      Strain      0
      dtype: int64
```

Train dataset

```
df_scaled.isnull().sum()
```

[87] ✓ 0.9s

```
Load        0
Actuator    0
Time        0
Strain      0
dtype: int64
```

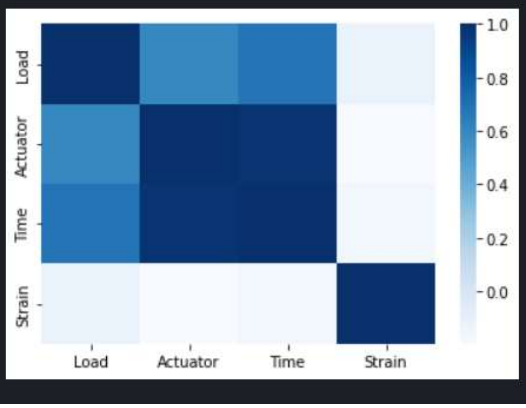
We use correlation value we get from sklearn library and plot a correlation heatmap using seaborn library.

Correlation Heatmap

```
sns.heatmap(df_scaled.corr() , cmap="Blues")
```

[89] ✓ 0.2s

```
<AxesSubplot:>
```



We find the linear regression using the sklearn linear regression module

Importing the linear regression module

```
linear_reg_model = linearreg()
```

[109] ✓ 0.9s

Fitting the train dataset in the module

```
linear_reg_model.fit(X,Y)
```

[110] ✓ 0.7s

```
... LinearRegression()
```

```
reg_equation = "Y = " + str(intercept.round(5)) + " + "
reg_equation += "(" + str(coefficients.coef['Time'].round(5)) + ")" + X.columns[0]
for i in range(1, len(X.columns)):
    col = X.columns[i]
    reg_equation += " + (" + str(coefficients.coef[col].round(5)) + ")" + col
print(reg_equation)
```

] ✓ 0.9s

```
Y = 0.05512 + (0.22724)Load + (-0.2526)Actuator + (0.22724)Time
```

Now we check the Statistical summary to check the P – values and if the P – value of a column is greater than 0.7 we need to remove it. We observe that P – Values are less so we don't remove any column of our data.

	coef	std err	t	P>|t|	[0.025	0.975]
Load	0.0337	0.002	14.580	0.000	0.029	0.038
Actuator	-0.1740	0.013	-13.142	0.000	-0.200	-0.148
Time	0.1282	0.014	9.369	0.000	0.101	0.155

Now we check the Varience Inflation Factor or VIF values and see if they are large. If there are large values then we remove one column and look at the VIF values again and we remove the columns till the VIF values are in double or single digit values.

VIF values at the start

	variables	VIF
0	Load	7.642338
1	Actuator	97.674635
2	Time	122.669998

VIF values after removing Time because it's huge value

	variables	VIF
0	Load	5.236635
1	Actuator	5.236635

Now that we have remove time the values came down a lot and now we can build our model using sklearn's linear regression agin and then we can view the result using the performance metrics module of sklearn library.

```
MAE = metrics.mean_absolute_error(Y_test, predictions)
MSE = metrics.mean_squared_error(Y_test,predictions)
RMSE = np.sqrt(MSE)
R_squared = result.rsquared
adjusted_R_squared = result.rsquared_adj

print("MSE (Mean Squared Error) : ", + MSE)
print("MAE (Mean Absolute Error): ", + MAE)
print("RMSE (Root Mean Squared Error ) : ", + RMSE)
print("Adjusted R squared value : " , adjusted_R_squared)
```

✓ 0.9s

```
MSE (Mean Squared Error) :  0.49089390550979306
MAE (Mean Absolute Error):  0.5030271563547885
RMSE (Root Mean Squared Error ) :  0.7006382129956894
Adjusted R squared value :  0.08166011217622104
```

Final Linear Regression will be

Input Variable Names	Regression	MSE	MAE	RMSE	R-Squared	Adjusted R-Squared
['Load', 'Actuator']	Y = 0.033+ (0.003)Load + (-0.045)Actuator	0.490894	0.503027	0.700638	0.0819094	0.0816601

Accuracy of the model is

Final Accuracy score will be

```
r2_score = regression_model.score(X_test,Y_test)
print(-(r2_score*100),'%')
```

✓ 0.1s

```
88.90679416276302 %
```

Finally we can say that the model is 88.9% accurate when we use the test dataset to predict when the model is built with the given train dataset.

Polynomial Regression

For the polynomial regression we use the polynomial function to scale our database according to polynomial regression and implement linear regression module on it.

```
poly = PolynomialFeatures(degree=5)
✓ 0.1s
```

Using the poly fit to transform the given X dataset for pol

```
x_poly = poly.fit_transform(X)
x_poly
✓ 0.1s
```

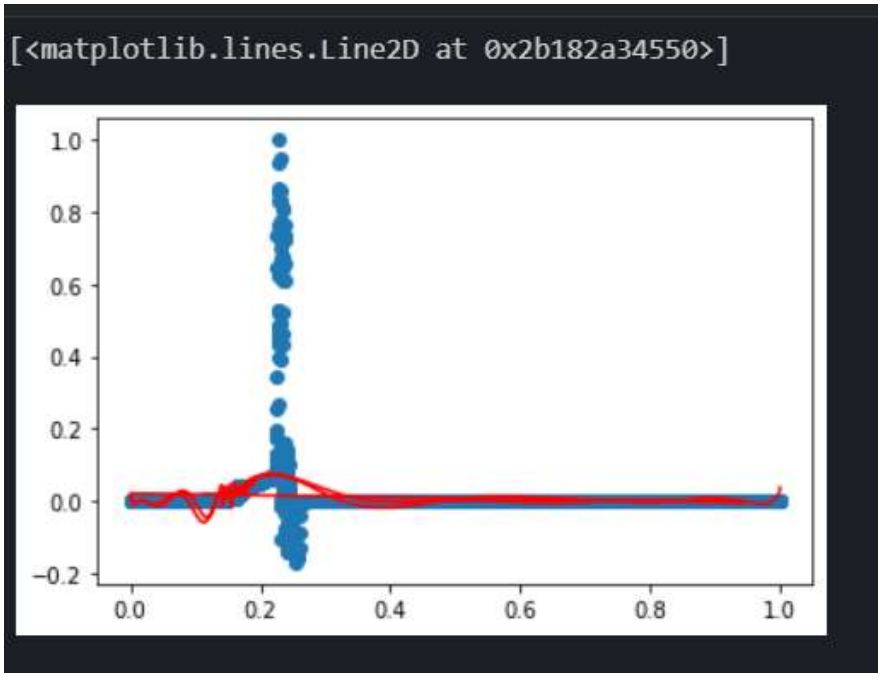
```
array([[ 1.00000000e+00, -1.19246678e-02,  0.00000000e+00, ...,
         0.00000000e+00, -0.00000000e+00,  0.00000000e+00],
       [ 1.00000000e+00, -1.19551004e-02,  1.56166789e-04, ...,
         5.44342816e-16, -7.11062785e-18,  9.28845334e-20],
       [ 1.00000000e+00, -1.16183971e-02,  1.56166789e-04, ...,
         5.14112866e-16, -6.91036422e-18,  9.28845334e-20],
       ...,
       [ 1.00000000e+00,  6.37644257e-01,  9.99218941e-01, ...,
         4.05638229e-01,  6.35654439e-01,  9.96100802e-01],
       [ 1.00000000e+00,  6.35310048e-01,  9.99687621e-01, ...,
         4.03240730e-01,  6.34516591e-01,  9.98439083e-01],
       [ 1.00000000e+00,  6.31070824e-01,  1.00000000e+00, ...,
         3.98250385e-01,  6.31070824e-01,  1.00000000e+00]])
```

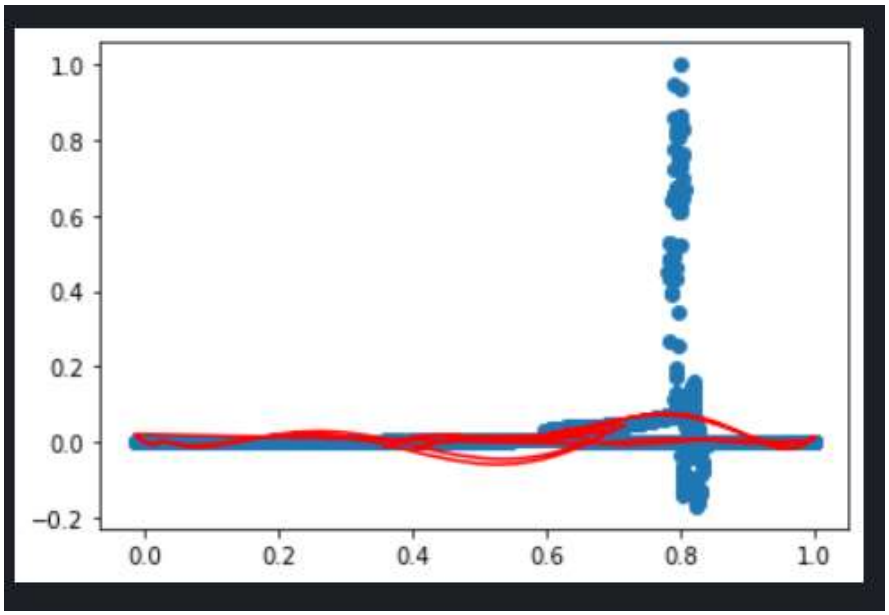
Predictions using the linear regression model

```
[138]  y_pred = regression_model.predict(x_poly)
       ✓ 0.1s
```

```
[139]  y_pred
       ✓ 0.1s
...    array([0.01469297, 0.01521425, 0.01460905, ..., 0.02586711, 0.02908737,
              0.03689913])
```

Regression line and original train dataset plots





Evaluation metrics

```
MAE = metrics.mean_absolute_error(Y_test, y_pred[:Y_test.shape[0]])
MSE = metrics.mean_squared_error(Y_test,y_pred[:Y_test.shape[0]])
RMSE = np.sqrt(MSE)
R_squared = result.rsquared
adjusted_R_squared = result.rsquared_adj

print("MSE (Mean Squared Error) : ", + MSE)
print("MAE (Mean Absolute Error): ", + MAE)
print("RMSE (Root Mean Squared Error ) : ", + RMSE)
print("Adjusted R squared value : " , adjusted_R_squared)
```

159] ✓ 0.1s

```
MSE (Mean Squared Error) :  0.4911505397194405
MAE (Mean Absolute Error):  0.5059719009816611
RMSE (Root Mean Squared Error ) :  0.7008213322377113
Adjusted R squared value :  0.08166011217622104
```

As we can see the linear regression model and the Polynomial linear regression model have the same MSE , MAE values and slight variation in the RMSE which means the Polynomial regression model is better and Polynomial regression model ignores the value of Load because it's very small and almost un significant.

Polynomial linear regression equation will be

```
Regression Equation:
Y = -0.007+ (0.0)Load + (-1.863)Actuator
```

Conclusion

Linear regression model and Polynomial Linear regression model are successfully implemented.

Therefore by the RMSE values we can say that Polynomial linear regression model is a more accurate model than the Linear Regression

References

https://www.geeksforgeeks.org/countplot-using-seaborn-in-python/

https://en.wikipedia.org/wiki/Polynomial_regression#:~:text=In%20statistics%2C%20polynomial%20regression%20is,nth%20degree%20polynomial%20in%20x.

https://www.analyticsvidhya.com/blog/2021/07/all-you-need-to-know-about-polynomial-regression/

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

https://www.statisticshowto.com/variance-inflation-factor/