International Conference on Innovative Research in Engineering, Management and Sciences
International Journal of Scientific Research in Computer Science, Engineering and Information Technology

Machine Learning Model Building for Predicting Roughness of Prototype built using Rapid Prototyping

Vinayak P B
Assistant Professor, Department of Mechanical Engineering, New Horizon college of Engineering, Bangalore.

ABSTRACT

Linear Regression is one the most common algorithm for prediction of continuous response variables. But the accuracy with the prediction is less because of the multi collinearity effects involved in the model. In the case study presented a dataset on predicting the roughness of a rapid prototype is done using multi linear regression model building. The accuracy of the prediction is increased by removing the effects of multi collinearity from the model.
Keywords : Multi Linear Regression, Multi- Collinearity

I. INTRODUCTION

Machine learning is one of the fastest growing interdisciplinary areas of Engineering, with wide range of applications. The project aims to introduce machine learning, and the algorithmic paradigms it offers, in a principled way. It is one of the technique from which computers can learn from input available to them. The algorithm learns by the input data we are giving which inturn represents an experience on a particular task. The learning depends majorly of the data we have collected. We are trying to study particular pattern a data is exhibiting. So the data collection has to be genuine.

MULTIPLE LINEAR REGRESSION IN MACHINE LEARNING:

Regrssion problems are an important category of problems in analytics in which the response variable (Y) takes a continuous value. Multiple linear regression means linear in regression parameters (beta values). The following is the examples of multiple linear regression:

$$Y = \beta_0+\beta_1X_1+\beta_2X_2+\ldots\ldots\ldots\ldots+\beta_kX_K$$

Ordinary Least Squares Estimation for Multiple Linear Regression

The assumptions that are made in multiple linear regression model are as follows:

- The regression model is linear in parameter.
- The explanatory variable, *X*, is assumed to be non-stochastic (that is, *X* is deterministic).
- The conditional expected value of the residuals, $E(\varepsilon_i/X_i)$, is zero.
- In a time series data, residuals are uncorrelated, that is, $\text{Cov}(\varepsilon_i, \varepsilon_j) = 0$ for all $i \neq j$.
- The residuals, ε_i, follow a normal distribution.
- The variance of the residuals, $\text{Var}(\varepsilon_i/X_i)$, is constant for all values of X_i. When the variance of the residuals is constant for different values of X_i, it is called **homoscedasticity**. A non-constant variance of residuals is called **heteroscedasticity**.
- There is no high correlation between independent variables in the model (called **multi-collinearity**). Multi-collinearity can destabilize the model and can result in incorrect estimation of the regression parameters.

The regression coefficients β is given by

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

The estimated values of response variable are

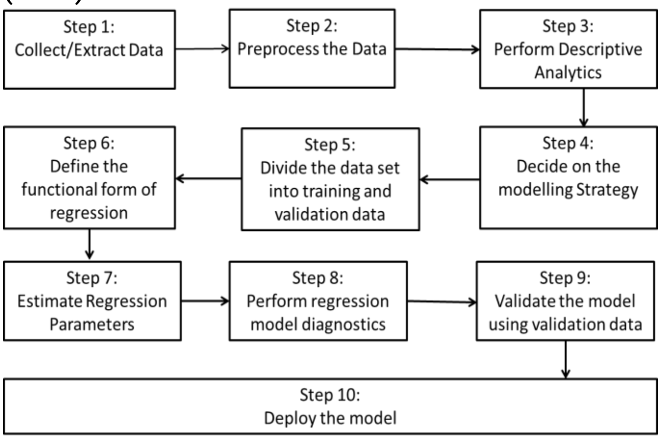
$$\hat{Y} = X\hat{\beta} = X(X^T X)^{-1} X^T Y$$

In above Eq. the predicted value of dependent variable is a linear function of Y_i. Equation can be written as follows:

$$H = X(X^T X)^{-1} X^T$$

is called the **hat matrix**, also known as the **influence matrix**, since it describes the influence of each observation on the predicted values of response variable

Framework for building multiple linear regression (MLR).



II. Experiment

Model Building:

The variables available for the evaluation of 'Roughn ess' are 'layer_height', 'wall_thickness','infill_density','infill_pattern','nozzl e_temperature','bed_temperature','print_speed','mate rial','fan_speed'

A total of 50 observations were recorded.

```
import pandas as pd
import numpy as np
```

```
RAPID_PROT_PRINT = pd.read_csv(r"C:\Users\dell\Desktop\Current\Mach
```

```
RAPID_PROT_PRINT.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 12 columns):
layer_height          50 non-null float64
wall_thickness        50 non-null int64
infill_density        50 non-null int64
infill_pattern        50 non-null object
nozzle_temperature    50 non-null int64
bed_temperature       50 non-null int64
print_speed           50 non-null int64
material              50 non-null object
fan_speed             50 non-null int64
roughness             50 non-null int64
tension_strenght      50 non-null int64
elongation            50 non-null float64
dtypes: float64(2), int64(8), object(2)
memory usage: 4.8+ KB
```

```
RAPID_PROT_PRINT.head(5)
```

	layer_height	wall_thickness	infill_density	infill_pattern	nozzle_temperature	bed_temperature	print_speed	material	fan_speed	roughness	t
0	0.02	8	90	grid	220	60	40	abs	0	25	
1	0.02	7	90	honeycomb	225	65	40	abs	25	32	
2	0.02	1	80	grid	230	70	40	abs	50	40	
3	0.02	4	70	honeycomb	240	75	40	abs	75	68	
4	0.02	6	90	grid	250	80	40	abs	100	92	

```
X_features=RAPID_PROT_PRINT.columns
print(X_features)
```

```
Index(['layer_height', 'wall_thickness', 'infill_density', 'infill_pattern',
       'nozzle_temperature', 'bed_temperature', 'print_speed', 'material',
       'fan_speed', 'roughness', 'tension_strenght', 'elongation'],
      dtype='object')
```

```
X_features = ['layer_height','wall_thickness','infill_density','infill_pattern','nozzle_temperature','bed_temperature','print_spee
```

```
RAPID_PROT_PRINT['infill_pattern'].unique()
```

```
array(['grid', 'honeycomb'], dtype=object)
```

```
categorical_features=['infill_pattern','material']
```

```
RAPID_PROT_PRINT_encoded=pd.get_dummies(RAPID_PROT_PRINT[X_features],columns=categorical_features,drop_first=True)
```

```
RAPID_PROT_PRINT_encoded
```

```
X_features = RAPID_PROT_PRINT_encoded.columns
print(X_features)
```

```
Index(['layer_height', 'wall_thickness', 'infill_density',
       'nozzle_temperature', 'bed_temperature', 'print_speed', 'fan_speed',
       'infill_pattern_honeycomb', 'material_pla'],
      dtype='object')
```

```
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
X=sm.add_constant(RAPID_PROT_PRINT_encoded)
Y=RAPID_PROT_PRINT['roughness']
train_X,test_X,train_Y,test_Y=train_test_split(X,Y,train_size=0.8,random_state=42)
```

```
RAPID_PROT_PRINT_model=sm.OLS(train_Y,train_X).fit()
```

```
RAPID_PROT_PRINT_model.summary()
```

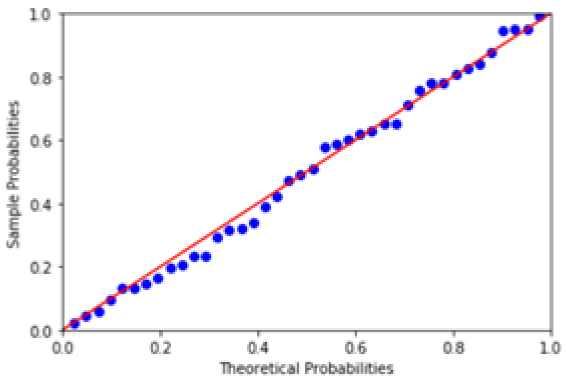
OLS Regression Results

Dep. Variable:	roughness	**R-squared:**	0.881
Model:	OLS	**Adj. R-squared:**	0.851
Method:	Least Squares	**F-statistic:**	28.79
Date:	Wed, 18 Dec 2019	**Prob (F-statistic):**	2.79e-12
Time:	11:10:49	**Log-Likelihood:**	-199.87
No. Observations:	40	**AIC:**	417.7
Df Residuals:	31	**BIC:**	432.9
Df Model:	8		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.8641	0.186	-4.645	0.000	-1.243	-0.485
layer_height	1257.5903	100.569	12.505	0.000	1052.478	1462.703
wall_thickness	2.4151	2.555	0.945	0.352	-2.796	7.626
infill_density	-0.0205	0.272	-0.075	0.940	-0.575	0.534
nozzle_temperature	13.6205	2.968	4.589	0.000	7.567	19.674
bed_temperature	-50.3923	10.881	-4.631	0.000	-72.583	-28.201
print_speed	0.6742	0.239	2.820	0.008	0.187	1.162
fan_speed	7.2580	1.423	5.099	0.000	4.355	10.161
infill_pattern_honeycomb	-5.8959	13.531	-0.436	0.666	-33.492	21.700
material_pla	265.9751	69.087	3.850	0.001	125.071	406.879

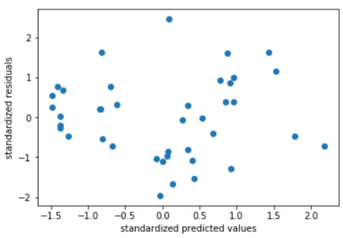
```
import matplotlib.pyplot as plt
import seaborn as sn
```

```
RAPID_PROT_PRINT_resid=RAPID_PROT_PRINT_model.resid
probplot=sm.ProbPlot(RAPID_PROT_PRINT_resid)
probplot.ppplot(line='45')
plt.show()
```




```
def get_standardized_values(vals):
    return(vals-vals.mean())/vals.std()
```

```
plt.scatter(get_standardized_values(RAPID_PROT_PRINT_model.fittedvalues),get_standardized_values(RAPID_PROT_PRINT_resid))
plt.xlabel("standardized predicted values")
plt.ylabel("standardized residuals")
```

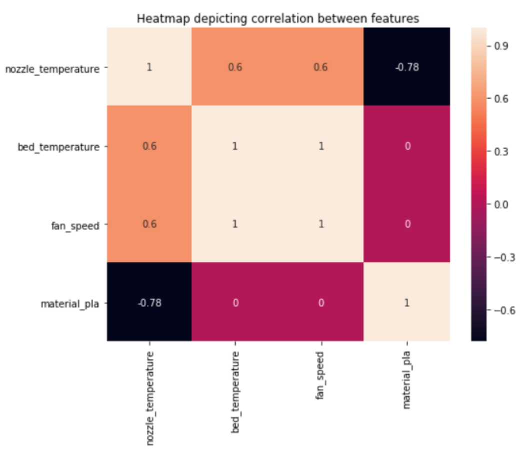



```
from statsmodels.stats.outliers_influence import variance_inflation_factor
def get_vif_factors(X):
    X_matrix=X.values
    vif=[variance_inflation_factor(X_matrix,i) for i in range (X_matrix.shape[1])]
    vif_factors=pd.DataFrame()
    vif_factors['column']=X.columns
    vif_factors['VIF']=vif
    return vif_factors
```

	column	VIF
0	layer_height	1.067264
1	wall_thickness	1.371753
2	infill_density	1.180552
3	nozzle_temperature	47.069693
4	bed_temperature	14577.380676
5	print_speed	1.253881
6	fan_speed	196.700154
7	infill_pattern_honeycomb	1.087509
8	material_pla	29.112041

```
columns_with_large_vif=vif_factors[vif_factors.VIF>4].column
```

```
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(20,18))
sn.heatmap(X[columns_with_large_vif].corr(), annot=True);
plt.title("Heatmap depicting correlation between features");
```




```
columns_to_be_removed=["nozzle_temperature","bed_temperature"]
```

```
X_new_features=list(set(X_features)-set(columns_to_be_removed))
```

```
get_vif_factors(X[X_new_features])
```

	column	VIF
0	layer_height	3.188074
1	print_speed	4.042619
2	fan_speed	2.771714
3	wall_thickness	3.998287
4	infill_density	4.989614
5	infill_pattern_honeycomb	2.137185
6	material_pla	2.165480

```
train_X=train_X[X_new_features]
RAPID_PROT_PRINT_2=sm.OLS(train_Y,train_X).fit()
RAPID_PROT_PRINT_2.summary()
```

OLS Regression Results

Dep. Variable:	roughness	R-squared:	0.944
Model:	OLS	Adj. R-squared:	0.932
Method:	Least Squares	F-statistic:	78.78
Date:	Wed, 18 Dec 2019	Prob (F-statistic):	9.45e-19
Time:	11:16:52	Log-Likelihood:	-210.58
No. Observations:	40	AIC:	435.2
Df Residuals:	33	BIC:	447.0
Df Model:	7		

	coef	std err	t	P>|t|	[0.025	0.975]
layer_height	1226.9310	117.289	10.461	0.000	988.305	1465.557
print_speed	0.4514	0.233	1.941	0.061	-0.022	0.924
fan_speed	0.6716	0.233	2.888	0.007	0.199	1.145
wall_thickness	0.9515	2.718	0.350	0.729	-4.578	6.481
infill_density	0.0840	0.301	0.279	0.782	-0.528	0.696
infill_pattern_honeycomb	-14.8142	16.957	-0.874	0.389	-49.314	19.686
material_pla	-47.9192	16.855	-2.843	0.008	-82.210	-13.628

Omnibus:	3.064	Durbin-Watson:	1.643
Prob(Omnibus):	0.216	Jarque-Bera (JB):	2.658
Skew:	0.533	Prob(JB):	0.265
Kurtosis:	2.324	Cond. No.	1.49e+03

III. Conclusion

The most significant factors after removing multi-collinearity are “Layer_height”. The model has an R-Square value of 0.715. The model also satisfies the normality condition.

IV. REFERENCES

[1] Kavitha S, A comparative analysis on linear regression and support vector regression, IEEE, 04 May 2017.

[2] Nehal N Ghosalkar, Real Estate Value Prediction Using Linear Regression, IEEE, 25 April 2019

[3] Chandrasegar Thirumalai, Heuristic prediction of rainfall using machine learning techniques, 2017 International Conference on Trends in Electronics and Informatics (ICEI), 11-12 May 2017

[4] T. Praveen Kumar, Fault Diagnosis of Automobile Gearbox Based on Machine Learning Techniques, Elsevier, vol 97, 2014, pp.2092-2098.

[5] Lei Liu, Research on Logistic Regression Algorithm of Breast Cancer Diagnose Data by Machine Learning, International Conference of Robots and Intelligent systems, July 2018.

[6] Glenn. V. Ostir, "Logistic Regression: A Nontechnical Review", American Journal of Physical Medicine & Rehabilitation., vol. 6, 2000, pp. 565-572.
[7] C. Carter, J. Catlett, "Assessing credit card applications using machine learning", IEEE Expert, vol. 2, no. 3, 1987, pp. 71-79.
[8] G. S. Fang, "A note on optimal selection of independent observations", IEEE Trans. on Systems Man and Cybernetics, vol. 9, May 1979, pp. 309-311.