Assignment-1: A mathematical essay on Linear Regression

Vasudev Gupta
Department of Mechanical Engineering
Indian Institute of Technology Madras
Chennai, India
7vasudevgupta@gmail.com

***Abstract*—In this essay, we will be presenting a mathematical study on linear regression models. We will be applying the linear regression model to predict the mortality rate and will try to analyse if fitting the linear regression model was the right decision. We will also try to do exploratory analysis on the data and do feature selection to feed the right data to the model. Finally, we will try to analyse if our model can be presented to the client by analysing the generalization capacity of the model on unseen data.**

I. Introduction

Machine learning techniques are quite popular these days and is helping several companies in solving quite hard problems based on data-driven approaches. Machine learning models try to approximate the function on the given training data by capturing the underlying distribution of the training data and then predicts on unseen data using the learnt distribution. While preparing the model, one assumes that training data is similar to unseen data and hence the model can generalize well to the future data.

In this essay, we will be focusing on the linear regression model. Linear models work under the assumption that residual error is normally distributed and a line can be fitted on the training data. We will discuss a detailed mathematical study in the next section.

In this essay, we are provided with the data of several regions and we need to present a study on whether low-income groups are more prone to fatality because of cancer. We are given several features based on which we can make the predictions but we need to analyse if some particular feature is affecting the model's performance significantly.

In this essay, we will be deriving solutions to linear regression models mathematically and will use existing Python libraries to fit the linear model on the given data. We will also discuss the insights we have gained from data and how we decided to choose certain features. Finally, we will see if our predictions follow the assumption we will make before fitting the model to data.

II. Linear Regression

In linear regression, model's response is approximated as a linear function of the inputs:

$$y(x) = w^T x + \epsilon = \sum_{i=1}^{D} w_i x_i + \epsilon$$

where $w^T x$ represents the inner or scalar product between the input vector x and the model's weight vector w, and ϵ is the residual error between our linear predictions and the true response.

We often assume that ϵ has a Gaussian distribution. We denote this by $N_\epsilon(\mu, \sigma^2)$, where μ is the mean and σ^2 is the variance. Mathematically, we can say:

$$p(y|x, \theta) = N(y|\mu(x), \sigma^2(x))$$

In the simplest case, we assume μ is a linear function of x, so $\mu = w^T x$, and that the noise is fixed, $\sigma^2(x) = \sigma^2$. In this case, $\theta = (w, \sigma^2)$ are the parameters of the model.

$$\mu(x) = w_0 + w_1 x = w^T x$$

where w_0 is the bias term, w_1 is the slope, and where we have defined the vector $x = (1, x)$. If w_1 is positive, it means we expect the output to increase as the input increases.

Linear regression can be made to model non-linear relationships by replacing x with some non-linear function of the inputs, $\phi(x)$. That is, we use

$$p(y|x, \theta) = N(y|w^T \phi(x), \sigma^2)$$

A. Maximum likelihood estimation

A common way to estimate the parameters of a statistical model is to compute the MLE, which is defined as

$$\theta = \arg max_\theta \log p(D|\theta)$$

It is common to assume the training examples are independent and identically distributed. This means we can write the log-likelihood as follows:

$$l(\theta) = \log p(D|\theta) = \sum_{i=1}^{N} \log p(y_i|x_i, \theta)$$

Instead of maximizing the log-likelihood, we can equivalently minimize the negative log likelihood:

$$NLL(\theta) = -\sum_{i=1}^{N} logp(y_i|x_i, \theta)$$

Now, if we apply the method of MLE to the linear regression and insert the definition of the Gaussian into the above, we find that the log likelihood is given by

$$l(\theta) = \sum_{i=1}^{N} \log[(\frac{1}{2\pi\sigma^2})^{\frac{1}{2}} \exp^{(-\frac{1}{2\sigma^2}(y_i - w^T x_i)^2)}]$$

$$= -\frac{1}{2\sigma^2} RSS(w) - \frac{N}{2} log(2\pi\sigma^2)$$

where RSS stands for **residual sum of squares** and is defined by

$$RSS(w) = \sum_{i=1}^{N} (y_i - w^T x_i)^2$$

It can also be written as the square of the l2 norm of the vector of residual errors:

$$RSS(w) = ||\epsilon||_2^2 = \sum_{i=1}^{N} \epsilon_i^2$$

where $\epsilon_i = (y_i - w^T x_i)$.

B. Derivation of the MLE

First, we rewrite the objective in a form that is more amenable to differentiation:

$$NLL(w) = \frac{1}{2}(y - Xw)^T (y - Xw) = \frac{1}{2} w^T (X^T X) w - w^T (X^T y)$$

where

$$X^T X = \sum_{i=1}^{N} x_i x_i^T = \sum_{i=1}^{N} \begin{pmatrix} x_{i,1}^2 & \cdots & x_{i,1} x_{i,D} \\ & \ddots & \\ x_{i,D} x_{i,1} & \cdots & x_{i,D}^2 \end{pmatrix}$$

is the sum of squares matrix and

$$X^T y = \sum_{i=1}^{N} x_i y_i .$$

Using results from above equation, we see that the gradient of this is given by

$$g(w) = [X^T X w - X^T y] = \sum_{i=1}^{N} x_i (w^T x_i - y_i)$$

equating to zero we get,

$$X^T X w = X^T y$$

This is known as the normal equation. The corresponding solution w^ to this linear system of equations is called the ordinary least squares or OLS solution, which is given by

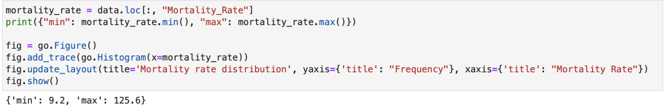
$$w_{OLS} = (X^T X)^{-1} X^T y$$

III. The Problem

In subsequent sections, we will be analysing and cleaning the data provided by the client. We will also fit the machine learning model to the data and will inspect if model assumptions meet its application on real-world data.

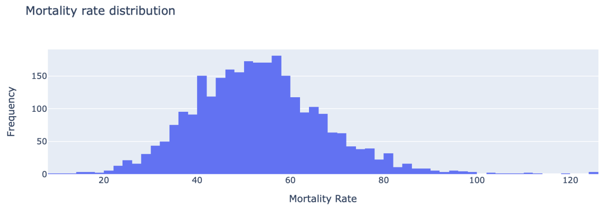
A. Data Analysis

As we discussed in the previous section that linear regression works based on assumption that the conditional distribution of the target variable is normal. Hence to verify if it's a good idea to implement linear regression to this problem, we plotted the histogram of the target variable (i.e. Mortality Rate).



```
mortality_rate = data.loc[:, "Mortality_Rate"]
print({"min": mortality_rate.min(), "max": mortality_rate.max()})

fig = go.Figure()
fig.add_trace(go.Histogram(x=mortality_rate))
fig.update_layout(title='Mortality rate distribution', yaxis={'title': "Frequency"}, xaxis={'title': "Mortality Rate"})
fig.show()
```

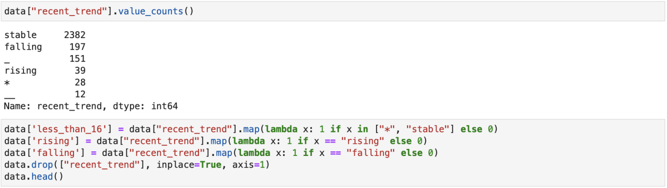
```
{'min': 9.2, 'max': 125.6}
```



The histogram of mortality rate looked like a normal curve for most of the samples, hence we decided to approximate the mortality rate with the linear model in the rest of the essay.

Now, we decided to clean the data as several features had missing values while few features had string values. Machine learning models can understand only numbers, hence we encoded string features into numbers and called them as categorical variables. We also dropped the samples which had many missing values and imputed the few features with mean of that feature.

For example, column recent_trend had random values in several samples, hence we decided to combine all those samples and imputed them with 0.



```
data["recent_trend"].value_counts()
```

```
stable     2382
falling     197
_           151
rising       39
*            28
__           12
Name: recent_trend, dtype: int64
```

```
data['less_than_16'] = data["recent_trend"].map(lambda x: 1 if x in ["*", "stable"] else 0)
data['rising'] = data["recent_trend"].map(lambda x: 1 if x == "rising" else 0)
data['falling'] = data["recent_trend"].map(lambda x: 1 if x == "falling" else 0)
data.drop(["recent_trend"], inplace=True, axis=1)
data.head()
```

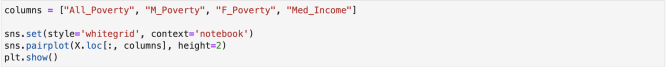
We applied similar procedure to most of the features and obtained the cleaned version of the data. Now, we decided to analyse data further and select all the relevant features out of the bulk of features. We will discuss more on that in the next section.

B. Data Analysis

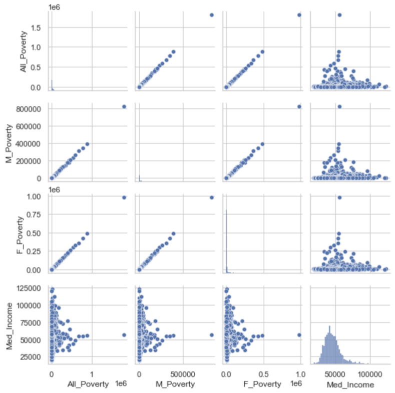
It is well known said in Machine Learning that ”feeding junk in, get the junk out”. Keeping this in mind, we tried to analyse and find out all the important features from our cleaned version of the data and decided to train our model only on that data.

To find out important features, we first discarded all the features that are highly correlated since highly correlated features will make our model unnecessarily complex and the model may get more prone to over-fitting to those particular features.

In the following graph, we tried to plot a scatter plot between several features and tried to visualize if a pair of features follow similar trend.

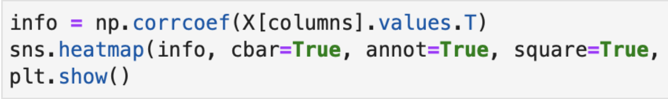


```
columns = ["All_Poverty", "M_Poverty", "F_Poverty", "Med_Income"]

sns.set(style='whitegrid', context='notebook')
sns.pairplot(X.loc[:, columns], height=2)
plt.show()
```

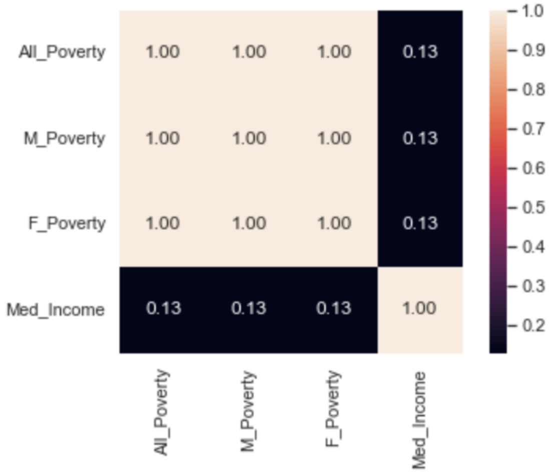


Based on the above graph, we concluded that ‘M_Poverty‘ and ‘F_Poverty‘ are highly correlated to ‘All_Poverty‘. Hence we decided to drop these features.

Further to get show correlation mathematically, we plotted a heat map of correlation among several features in the following figure.

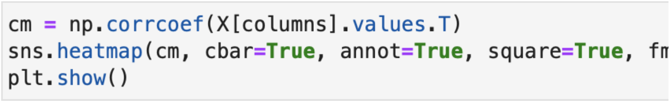


```
info = np.corrcoef(X[columns].values.T)
sns.heatmap(info, cbar=True, annot=True, square=True,
plt.show()
```

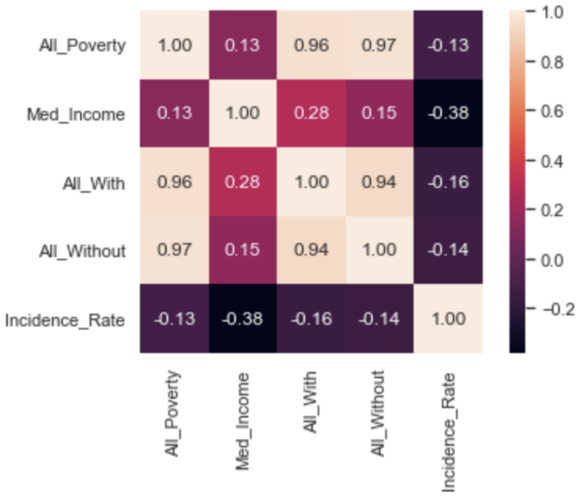


	All_Poverty	M_Poverty	F_Poverty	Med_Income
All_Poverty	1.00	1.00	1.00	0.13
M_Poverty	1.00	1.00	1.00	0.13
F_Poverty	1.00	1.00	1.00	0.13
Med_Income	0.13	0.13	0.13	1.00

We did a similar thing with another subset of features and realized that ‘All_With‘ and ‘All_Without‘ can be dropped further as they are highly correlated with ‘All_Poverty‘.



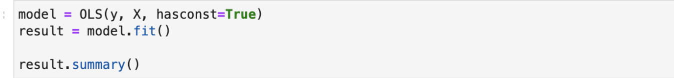
```
cm = np.corrcoef(X[columns].values.T)
sns.heatmap(cm, cbar=True, annot=True, square=True, fm
plt.show()
```



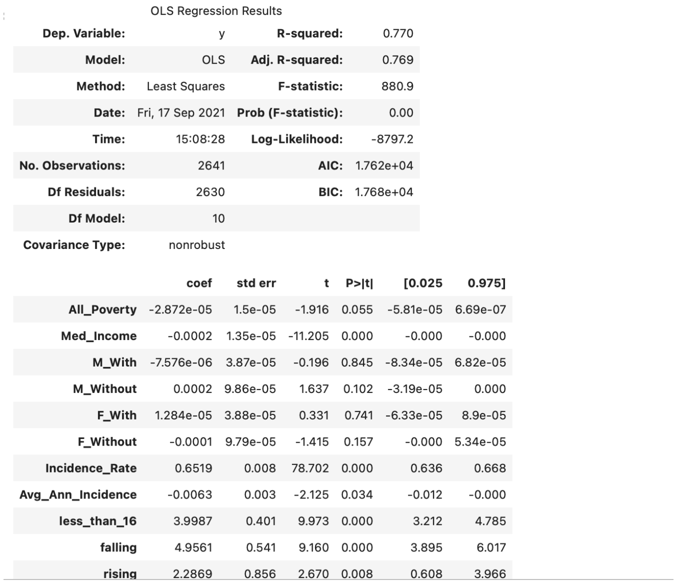
	All_Poverty	Med_Income	All_With	All_Without	Incidence_Rate
All_Poverty	1.00	0.13	0.96	0.97	-0.13
Med_Income	0.13	1.00	0.28	0.15	-0.38
All_With	0.96	0.28	1.00	0.94	-0.16
All_Without	0.97	0.15	0.94	1.00	-0.14
Incidence_Rate	-0.13	-0.38	-0.16	-0.14	1.00

C. Model Fitting

After selecting the subset of features from complete data, we decided to use ‘statsmodels‘ library for fitting the linear model over the training data. We got an R-squared score of 0.77. After seeing the coefficients of the model, we realized that several features had coefficients close to 0. Hence we decided to train our next model on a subset of features whose coefficients had significant values.



```
model = OLS(y, X, hasconst=True)
result = model.fit()

result.summary()
```

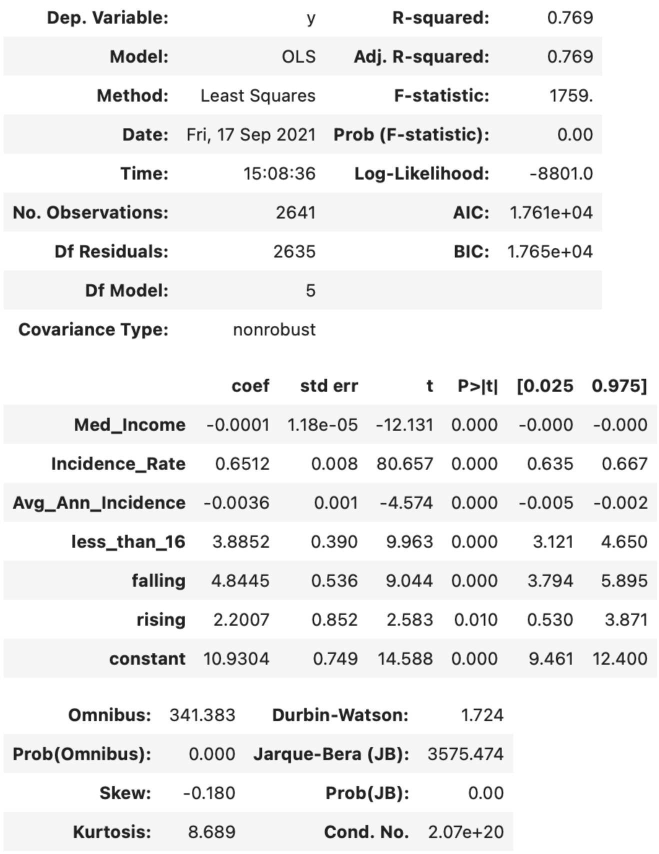


OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.770
Model:	OLS	**Adj. R-squared:**	0.769
Method:	Least Squares	**F-statistic:**	880.9
Date:	Fri, 17 Sep 2021	**Prob (F-statistic):**	0.00
Time:	15:08:28	**Log-Likelihood:**	-8797.2
No. Observations:	2641	**AIC:**	1.762e+04
Df Residuals:	2630	**BIC:**	1.768e+04
Df Model:	10		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
All_Poverty	-2.872e-05	1.5e-05	-1.916	0.055	-5.81e-05	6.69e-07
Med_Income	-0.0002	1.35e-05	-11.205	0.000	-0.000	-0.000
M_With	-7.576e-06	3.87e-05	-0.196	0.845	-8.34e-05	6.82e-05
M_Without	0.0002	9.86e-05	1.637	0.102	-3.19e-05	0.000
F_With	1.284e-05	3.88e-05	0.331	0.741	-6.33e-05	8.9e-05
F_Without	-0.0001	9.79e-05	-1.415	0.157	-0.000	5.34e-05
Incidence_Rate	0.6519	0.008	78.702	0.000	0.636	0.668
Avg_Ann_Incidence	-0.0063	0.003	-2.125	0.034	-0.012	-0.000
less_than_16	3.9987	0.401	9.973	0.000	3.212	4.785
falling	4.9561	0.541	9.160	0.000	3.895	6.017
rising	2.2869	0.856	2.670	0.008	0.608	3.966

We iteratively did this i.e. removed 1 feature and visualize if it’s affecting the model’s performance by a significant amount. Based on this approach, we were able to remove 5 features without compromising the model’s performance. The significance of all the final features is shown in the following graph:



Dep. Variable:	y	**R-squared:**	0.769
Model:	OLS	**Adj. R-squared:**	0.769
Method:	Least Squares	**F-statistic:**	1759.
Date:	Fri, 17 Sep 2021	**Prob (F-statistic):**	0.00
Time:	15:08:36	**Log-Likelihood:**	-8801.0
No. Observations:	2641	**AIC:**	1.761e+04
Df Residuals:	2635	**BIC:**	1.765e+04
Df Model:	5		
Covariance Type:	nonrobust		

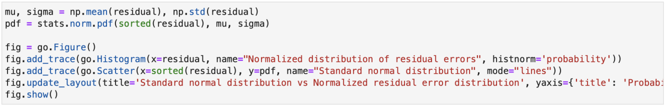
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Med_Income	-0.0001	1.18e-05	-12.131	0.000	-0.000	-0.000
Incidence_Rate	0.6512	0.008	80.657	0.000	0.635	0.667
Avg_Ann_Incidence	-0.0036	0.001	-4.574	0.000	-0.005	-0.002
less_than_16	3.8852	0.390	9.963	0.000	3.121	4.650
falling	4.8445	0.536	9.044	0.000	3.794	5.895
rising	2.2007	0.852	2.583	0.010	0.530	3.871
constant	10.9304	0.749	14.588	0.000	9.461	12.400

Omnibus:	341.383	**Durbin-Watson:**	1.724
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3575.474
Skew:	-0.180	**Prob(JB):**	0.00
Kurtosis:	8.689	**Cond. No.**	2.07e+20

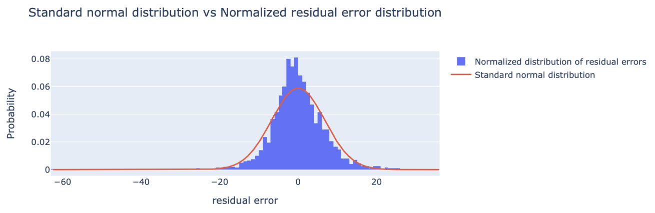
D. Model's Analysis

Further, we tried to analyse the model's predictions and tried to inspect if the model's results are as per the assumptions we made before fitting the model.

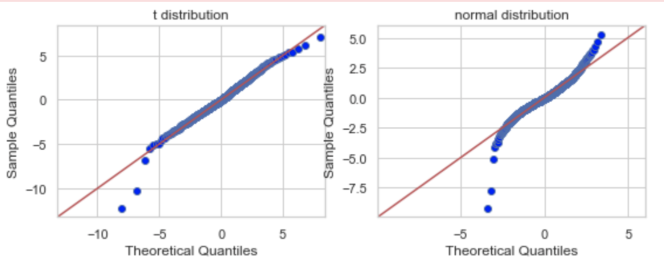
We plotted the distribution of the residuals. Note that we assumed that the linear model works based on assumption that residual error distribution is normal. We showed that this assumption holds with our model too.



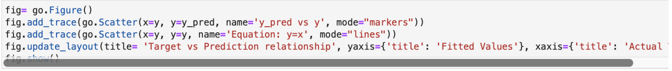
```
mu, sigma = np.mean(residual), np.std(residual)
pdf = stats.norm.pdf(sorted(residual), mu, sigma)

fig = go.Figure()
fig.add_trace(go.Histogram(x=residual, name="Normalized distribution of residual errors", histnorm='probability'))
fig.add_trace(go.Scatter(x=sorted(residual), y=pdf, name="Standard normal distribution", mode="lines"))
fig.update_layout(title='Standard normal distribution vs Normalized residual error distribution', yaxis={'title': 'Probabi
fig.show()
```



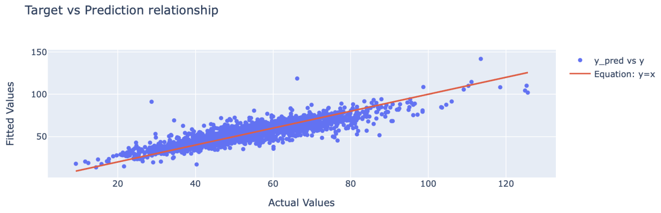
We also plotted the Q-Q plots to confirm if the residuals adhere more closely to the t-than normal distribution (fatter tails).



Finally, we plotted a graph between the model's prediction and target to visualize if predictions are following the trend of the target variable.



```
fig= go.Figure()
fig.add_trace(go.Scatter(x=y, y=y_pred, name='y_pred vs y', mode="markers"))
fig.add_trace(go.Scatter(x=y, y=y, name='Equation: y=x', mode="lines"))
fig.update_layout(title= 'Target vs Prediction relationship', yaxis={'title': 'Fitted Values'}, xaxis={'title': 'Actual '
fig.show()
```



IV. Conclusions

In this essay, we derived the solution of linear regression mathematically based on some assumptions. We also applied Linear regression techniques to real-world data and tried to inspect if the fitted model follows the assumptions. We conclude that a simple model like linear regression can be applied to real-world complex datasets if data follows the underlying assumptions.

References

[1] Christopher M. Bishop, ”Pattern Recognition and Machine Learning”
[2] Kevin P. Murphy, ”Machine Learning, A Probabilistic Perspective”
[3] Yunus Koloğlu, Hasan Birinci, Sevde Ilgaz Kanalmaz, Burhan Özyılmaz, ”A Multiple Linear Regression Approach For Estimating the Market Value of Football Players in Forward Position”