greatlearning
Learning for Life

A Short Monograph on Regression Model Selection

TO SERVE AS A REFRESHER FOR PGP-DSBA

Index

Contents

List of Figures

List of Tables

1. Introduction

1.1 Introduction to Predictive Modelling

Technology has given us the power to capture data from almost everything – road accidents, grocery store bar codes, customer loyalty programs as well as political opinions expressed on Twitter and Facebook. However, amassing data does not help us in any way unless we can understand the information contained in the data. Predictive modeling and data mining help to detect the hidden patterns in the data and to forecast for yet unobserved situations.

In this monograph and few subsequent monographs, various topics in predictive modeling and data mining will be taken up. But before we go into the details of predictive modeling, several major concepts need to be addressed.

What is the objective of predictive modeling?

Let us examine two different cases. Identification of spam or detection of fraudulent credit card transactions is two situations where the application of predictive modeling is important. In the former case, the objective is to detect spam email through a filter and classify it as such. In the latter case, the banks would want to identify frauds immediately, and red flag the transactions. In both cases, prediction accuracy needs to be very high, though how the spam was filtered or fraud was detected may not be so important. In this case, the model may be complex and the interpretability of the model low. Often these models are known as 'black box' models. Automation will work well here.

Consider another case where a physician needs to predict whether a post-menopausal woman above 65 years of age is at a risk of knee replacement surgery given various other health indicators. Here a 'black box' model may not be acceptable despite having high accuracy. The reason being, it is not enough to know who is at risk but it is mandatory to mitigate her risk of knee replacement. Unless a physician can understand which health indicators are more important, she will not be able to provide an effective treatment regime for her patient. Interpretability is a must in this situation. A black box model may be rejected in favor of an interpretable model, albeit accuracy in the former may be higher.

That is not to say predictive accuracy needs to be sacrificed totally to improve interpretability. Somewhere a balance needs to be struck. The suitability of the predictive model is an important issue that may need to be addressed case by case.

Predictive Analytics or Predictive Modelling is a process of extracting information from large complex data sets using a variety of computational tools to make predictions and estimates about future outcomes.

1.2 Supervised and Unsupervised Learning

All problems of data mining, pattern recognition, or predictive modeling come under the umbrella known as *Statistical Learning*.

Statistical learning problems can be partitioned into *Supervised* or *Unsupervised Learning*.

The problems discussed in the previous section belong to the first category. The objective in each case is to predict a response, typically denoted by Y. Corresponding to each unit of observation there is a set of independent variables or predictors (X), based on which Y is estimated (see the monograph on Regression). Whenever in the data set the response is available, the problem falls under supervised learning.

Supervised learning can be further divided into two classes, depending on the nature of the response Y. If Y is a continuous variable, the problem falls under the category Regression. On the other hand, if the response is qualitative, binary, or multi-class, the problem falls under the category Classification. Spam identification (Yes/No) and detection of fraud (Yes/No) are both classification problems.

The problem of assigning a risk value to a patient may be considered a classification problem if the risk is defined as low, medium, or high. If, however, a continuous risk probability is to be estimated for each patient, the problem is considered a regression problem.

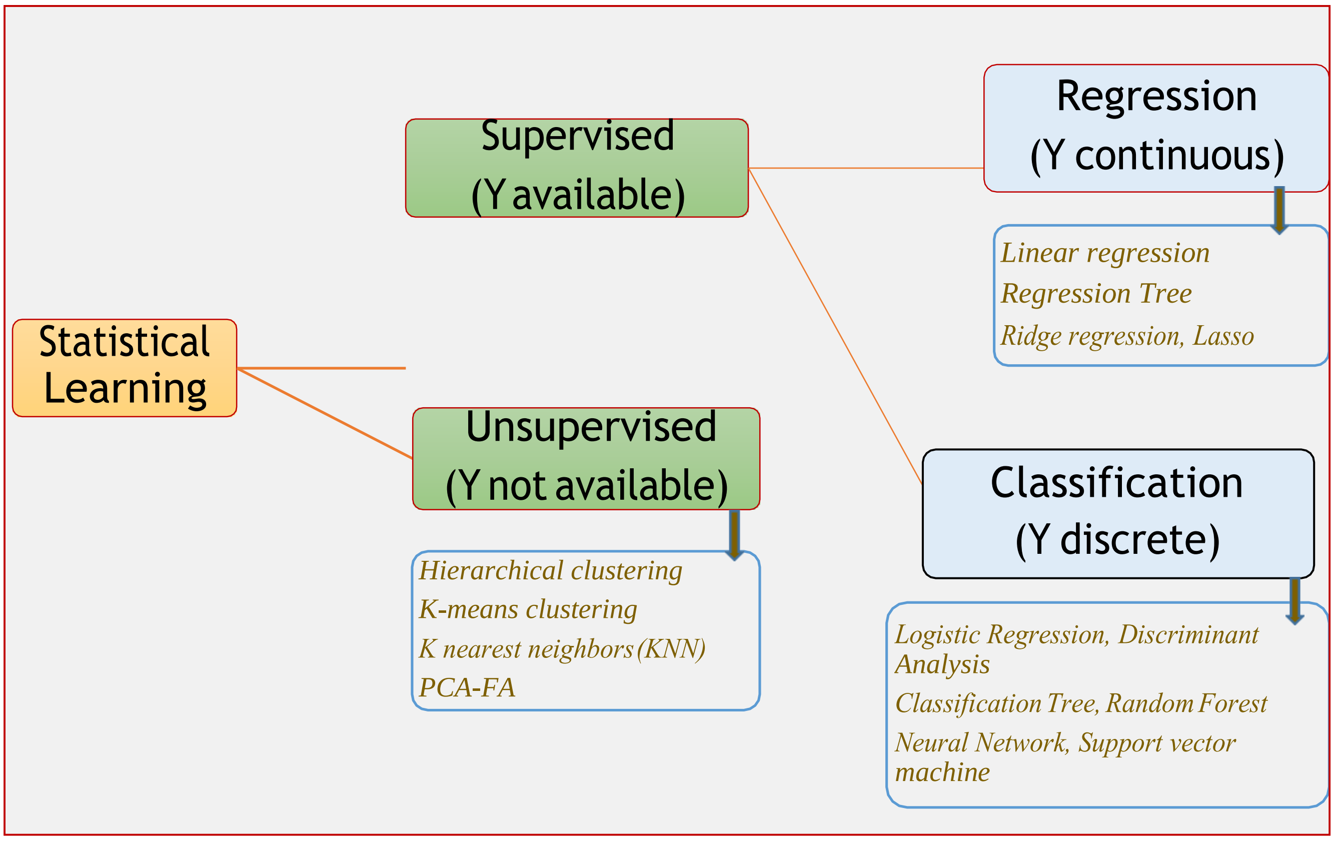
Unsupervised learning problems are those where there is no response. One example of an unsupervised learning problem is to categorize loyalty customers in a Gold, Silver, and Bronze classification, depending on their propensity of spending in a store. Detection of possible clusters in a multivariate data set is an unsupervised learning problem.

Another example of unsupervised learning is the extraction of factors from a complex data set (see the monograph on Dimension Reduction).

An illustrative figure is shown on the next page with a few techniques of different types of learning.

A few examples of classification and regression techniques. By no means the examples provided are comprehensive.

(Special note: *Neural network may also be used where the response is continuous and logistic regression may also include lasso regularization. But for initial understanding these examples are illustrative*.)



Statistical Learning
Supervised (Y available)
Unsupervised (Y not available)
Regression (Y continuous)
Linear regression
Regression Tree
Ridge regression, Lasso
Classification (Y discrete)
Logistic Regression, Discriminant Analysis
Classification Tree, Random Forest
Neural Network, Support vector machine
Hierarchical clustering
K-means clustering
K nearest neighbors (KNN)
PCA-FA

2. The Problem of Prediction: Bias Variance Trade-off

However accurate the prediction models are, there will always be a *prediction error*, the difference between the predicted and actual observations. Recall that for regression this error is also known as residual. Prediction error has two components: *bias* and *variance*. To minimize prediction error, both bias and variance need to be at a low level. Unfortunately, however, both bias and variance cannot be minimized simultaneously.

Identification of a suitable prediction model involves a trade-off between its bias and variance.

Gaining a proper understanding of bias and variance would help in avoiding the mistakes of overfitting or under-fitting. The problem of bias and variance are intimately associated with the problem of under-fitting and overfitting. A good model is one for which the bias and variance are as small as possible, and for which, the predictive ability is good.

The notions of bias and variance of a model are explained below.

Let us consider a model $Y=f(X)+\epsilon$, where f is an arbitrary function of the independent variables X and ϵ is the random error component. (Most models in predictive problems are of this type). The function f may be completely unknown or one may only have partial knowledge about its form. For example, suppose f to be linear, i.e. $f(x)=a+bx$. Unless the numerical values of a or b are known, only the type of the function f is known but not the complete function. Let an estimate $\hat{f}$ of f is obtained. For any given value of X, the predicted value $\hat{Y}=\hat{f}\,(X)$. Usually, $\hat{f}$ is computed using the data y, and hence $\hat{f}\,(x)$ is a random variable.

All the subsequent definitions and explanations in the next three sections will be based on this model.

2.1 *Bias of a model*

Bias is the average difference between the predicted values of a model and the observed values.

$$Bias=E(\hat{f}\,(x)-f(x))$$

where E denotes expectation or mean. A model is called unbiased if $Bias=0$. Bias can be both negative and positive, so a desirable condition for bias is that the absolute value of Bias is close to 0.

Recall here that a multiple linear regression model is unbiased since the average value of the least-squares estimate of residuals is 0.

A model with high bias pays very little attention to the dataset and oversimplifies the model. This phenomenon is called **under-fitting**. Under-fitting may happen for various reasons. The most common reasons include having only a limited amount of data to model a complex structure or fitting a linear model to non-linear data. An oversimplified model exhibits small variability.

2.2 Variance of a model

Variance is defined to be the variability of predicted values which quantifies the instability of the model.
Formally,

$$Variance=E(\hat{f}\ (x)-E(\hat{f}\ (x)))^2$$

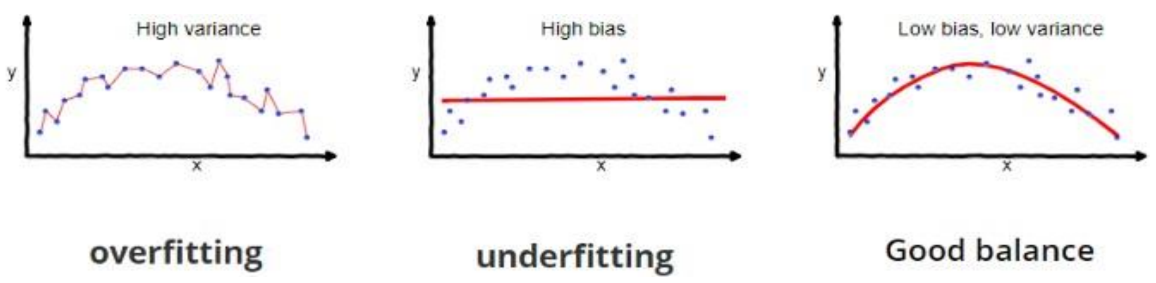
This is the error variance and is often denoted by the mean square error (MSE). Unlike bias, the variance can never be negative.

A model with high variance pays a lot of attention to the available data. Such models perform very well on data used to fit the model but have limited power to predict for the data that has not been observed. Hence it has high error rates on data used for prediction. This phenomenon is called **overfitting**. Overfitting happens when a model is too close to the observed data and captures the noise along with the underlying pattern in data.

Let us demonstrate the concepts of bias and variance through a visualization.

Observe n paired data points $(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)$. The model $y = f(x) + \epsilon$ is fitted to the data and the estimate $\hat{f}$ is obtained. If a new observation x is one of $x_1,x_2,\ldots,x_n$ then $\hat{f}$ can be chosen so that $E(\hat{f}\ (x)) = f(x)$ exactly and hence the bias is zero when evaluated at the observed x value. But if x is different from x_1, x_2,…, or x_n, then our estimate $\hat{f}$ may behave poorly. The reason is that $\hat{f}$ becomes very short-sighted since it tries to fit the observed data (x_i,y_i) as perfectly as possible, and does not take into consideration any new data point which could be used in the future.

The following picture illustrates these concepts.



overfitting underfitting Good balance

The leftmost panel shows an almost perfect fir to the observed dataset and thus bias is very small. But for this model the variability is very high. As soon as one new observation is added to the data, the function *f(x)* may change considerably.

The middle panel shows a model that completely ignores the curvature in the dataset and plots a straight line through it. Clearly it does not fit the data at all. This model varies only slightly for a different sample and impact of addition of a number of points may be negligible. Though in theory variance of a model cannot be zero, for all practical purposes it may be considered such. But the bias for this model is very high.

The rightmost panel shows a much better fit. It does not have zero bias, nor does it enjoy negligible variance, but it strikes a balance somewhere between the two and also fits the data well.

The objective of predictive modelling is to find such an optimum f(x).

2.3 Bias Variance Trade-off

If a model is too simple and has only a few parameters, then it may have high bias and low variance. On the other hand if a model has a large number of parameters then it is likely to have high variance and low bias. So it is essential to find the right balance without overfitting or under-fitting the data. This trade-off between bias and variance is closely associated with a trade-off in complexity of the model.

The total squared error of a model can be expressed as

$$Total\ squared\ error = Bias^2 + Variance + Irreducible\ error$$

It is not possible for any statistical model to manipulate irreducible error, which is inherent in the data.

The goal of predictive modelling is to reduce $Bias^2 + Variance$. As described above, due to the bias-variance trade-off, no model can reduce both bias and variance simultaneously, and therefore, a good model will try to minimize the sum of $Bias^2 + Variance$.

In general, an unbiased model with smallest possible variance is the preferred. However, in special situations, a biased model is deliberately chosen for which $Bias^2 + Variance$ is smaller than any unbiased model.

Such a model has the best predictive power, i.e. it is able to provide the best possible estimated value for a new observation.

2.4 Training and Test datasets

The discussion above establishes an important fact. Unless predictive ability of a model is tested on an independent data set, which is different than the one used to build the model, a vital aspect of the model is ignored. This necessitates splitting of the existing data set into two or more parts.

Training data: A training dataset is used to fit one or more models and estimate their parameters.

Test data: A test dataset is used to assess the performance of the developed model. The test set should be as close as possible to the training dataset (more formally, having the same distribution) but no overlap with the training dataset.

Typically training and test data are partitions of the observed data into a random 80:20 split. Other possible splits may be 75:25 or 70:30 or in some other ratio, all taken randomly. If the data is very large even 50:50 split into training and test is also possible. Training data is larger so that the model parameters are estimated with considerable accuracy. The purpose of having a test data is to check how close the predicted values are to the observed values.

Choice of test data may also be modified according to special applications. In certain predictive methods that are intended to be applied for a period of time (e.g. credit scoring), the test data is taken to be the most recent period. For example: Q1, Q2, and Q3 data are used in training set, but Q4 data is used as test set for a model that is intended to project credit risk for the next one quarter. In time series, one of the most specialized predictive models, only the most recent periods are used as test data. (This is discussed in detail in Time Series Monograph).

In the test data set no parameter estimation is performed.

2.5 Cross Validation

One criticism of training and test data split is that, the proposed model may depend on the split, since the split is done once only. One suggested alternative is to perform a k-fold cross-validation, which is an extension of the train-test split.

k-fold cross validation is a method of getting multiple sets of training and test data out of the original data set. The steps are as follows.

i) Split the whole dataset randomly into k equal (or almost equal) parts.

ii) Choose one of these k parts as the test dataset and the other $k-1$ parts together as the training dataset.

iii) Fit the model on the training dataset thus obtained and assess its performance on the test dataset. Usually, one predicts the values in the test dataset using this model and takes the square root of the error sum of squares (RMSE), but other measures of prediction errors are also possible

iv) Repeat this process k times, once for each of the k splits as the test data, and the complementary set as the training data

v) Finally, take average of all the k RMSEs to get final estimate of prediction error.

The most important advantage of k-fold cross-validation is that, each data point is included at least once in the training and in the test data. Compared to the training-test split method, cross-validation is more computation-intensive.

The value of k is usually taken to be 10. But due to computational complexity k = 5 is also a common choice. Note that the higher the value of k is, the smaller is the size of test data. That should be another consideration is choosing an optimum value of k.

3. Model Selection

In this section several approaches to regression model selection is discussed.

3.1 *Transformation of the response*

In the monograph on Linear Regression, after fitting the linear regression models, a residual analysis was performed to check if the regression assumptions (the LINE assumptions, see Sec 4.2.1) remained valid. Recall that one of the assumptions was normality. Another important assumption was that the variances of the response were constant.

If any or both of these two assumptions are violated, which can be detected from the residual plot, a transformation of the response may be necessary. Many transformations of the response are possible, but the most popular choice of transformations are given by the Box-Cox family of transformations.

Let λ be a constant, $\lambda \neq 0$. The response Y is transformed to a new variable Z (say) where Z = $(y^{\lambda}-1) / \lambda$. If $\lambda = 0$ then the response is changed to $\log(y)$. The transformed response Z fulfils the regression assumptions. However, it is clear from the functional form of the transformation, that Z can be simplified as Y^{λ}. This simplification works because $1 / \lambda$ being a constant, gets absorbed into the intercept term, and the regression slope parameters are multiplied (scaled) by the constant λ. If any predictor was significant in predicting $(y^{\lambda}-1) / \lambda$ then it is expected to be significant in predicting y^λ. Therefore, the inference remains unchanged whether we use y^{λ} or $(y^{\lambda}-1) / \lambda$.

That value of λ is chosen such that the log-likelihood curve reaches its maximum. This will be explained in further detail through the case study.

Case Study

[Refer to the monograph on Multiple Linear Regression]

A top wine manufacturer wants to invest in new technologies to improve its wine quality. Wine quality is directly dependent on the amount of alcohol in wines and the smoothness which, in turn, is controlled by various chemicals either directly added during the manufacturing process or generated through various chemical reactions. Wine certification and quality assessment are key elements for wine gradation and its price ticket. Wine certification is determined by various physiochemical elements in the wine. Therefore, the company wants to estimate the percentage (%) of alcohol in a bottle of wine as a function of various chemical components of the wine.

Statement of the Problem: Develop the best possible model to predict alcohol content in a red wine values of whose chemical components are known.

Description of the data (Data Dictionary):

Variables	Description
fixed acidity (FA)	Number of grams of Tartaric acid per cubic decimetre, $(g(tartaric\ acid)/dm^3)$
volatile acidity (VA)	Number of grams of Acetic acid per cubic decimetre, $(g(acetic\ acid)/\ dm^3)$
citric acid (CA)	Number of grams of Citric acid per cubic decimetre, (g/dm^3)
residual sugar (RS)	Number of grams of Residual sugar per cubic decimetre, $(g/\ dm^3)$
Chlorides	Number of grams of Sodium chloride per cubic decimetre, $(g(sodium\ chloride)/dm^3)$
free sulphur dioxide (FSD)	Number of milligram of Free sulphur dioxide per cubic decimetre, (mg/dm^3)
total sulphur dioxide (TSD)	Number of milligram of total sulphur dioxide per cubic decimetre, (mg/dm^3)
Density	Number of grams per cubic centimetre (g/cm^3)
Ph	pH is a scale of acidity from 0 to 14.
Sulphates	Number of grams of Potassium sulphate per cubic decimetre, $(g(potassium\ sulphate)\ /dm^3)$
Brand (categorical)	three different brands of wine are considered where 1 represents “Grover Zampa”, 2 represents “Seagram” and 3 represents “Sula Vineyards”.
Alcohol (response)	percentage volume of alcohol in wine $(\%\ vol.)$

We have considered the problem of building a multiple linear regression model in the monograph on Linear Regression. In this monograph, the problem of selecting the best possible predictive model using this dataset is considered.

The dataset is split into a training and a test data set according to a random 80:20 allocation. Training data contains 1279 observations since 80% of 1599 is approximately 1279. A simple random sample (without replacement) of size 1279 is taken from the first 1599 positive integers, and the training dataset is formed by selecting the rows of WineData corresponding to these random numbers. The remaining (1599-1279=) 320 rows will constitute the test dataset.

```
import numpy as np
import pandas as pd
import seaborn as sns
from scipy import stats
from sklearn.model_selection import train_test_split
from statsmodels.formula.api import ols
import statsmodels.regression.linear_model as sm
import matplotlib.pyplot as plt
df = pd.read_csv('WineData.csv')
X = df.drop(['alcohol','ID'],axis = 1)
y = df['alcohol']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
train_WineData = pd.concat([X_train,y_train],axis=1)
test_WineData = pd.concat([X_test,y_test],axis=1
train_WineData =
pd.concat([train_WineData,pd.get_dummies(train_WineData['Brand'],drop_first=True)],axis=1)
test_WineData =
pd.concat([test_WineData,pd.get_dummies(test_WineData['Brand'],drop_first=True)],axis=1)
train_WineData['SulaVineyards'] = train_WineData['Sula Vineyards']
test_WineData['SulaVineyards'] = test_WineData['Sula Vineyards']
train_WineData.drop(['Brand','Sula Vineyards'],axis = 1,inplace = True)
test_WineData.drop(['Brand','Sula Vineyards'],axis = 1,inplace = True)
```

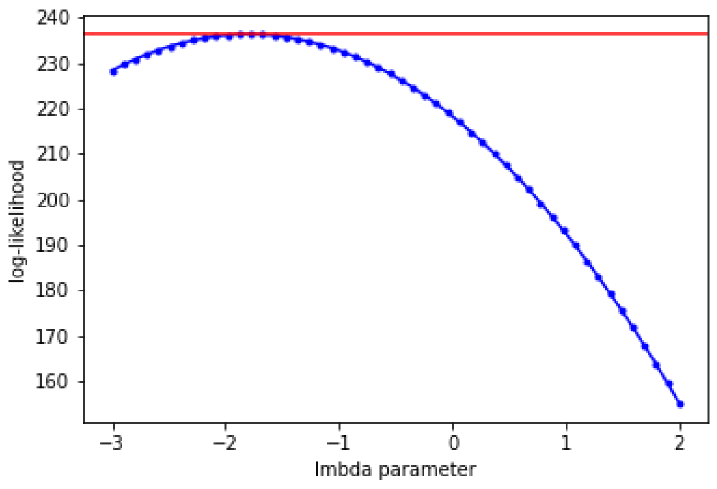
Thus the training dataset train_WineData in created on which the candidate models will be built. The test dataset is test_WineData on which the models will be validated by comparing their predictive ability.

Let us first determine the value λ to see whether any transformation is necessary. Note that henceforth all model building activities will be carried on the training data set.

```
MLR_wine =
ols(formula="alcohol~FA+VA+CA+RS+chloride+FSD+TSD+density+sulphate+pH+Seagram+SulaVi
neyards",data=train_WineData).fit()

lmbdas = np.linspace(-3, 2)
llf = np.zeros(lmbdas.shape, dtype=float)
for ii, lmbda in enumerate(lmbdas):
    llf[ii] = stats.boxcox_llf(lmbda, MLR_wine.fittedvalues)

fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(lmbdas, llf, 'b.-')
ax.axhline(stats.boxcox_llf(lmbda_optimal, MLR_wine.fittedvalues), color='r')
ax.set_xlabel('lmbda parameter')
ax.set_ylabel('log-likelihood')
```



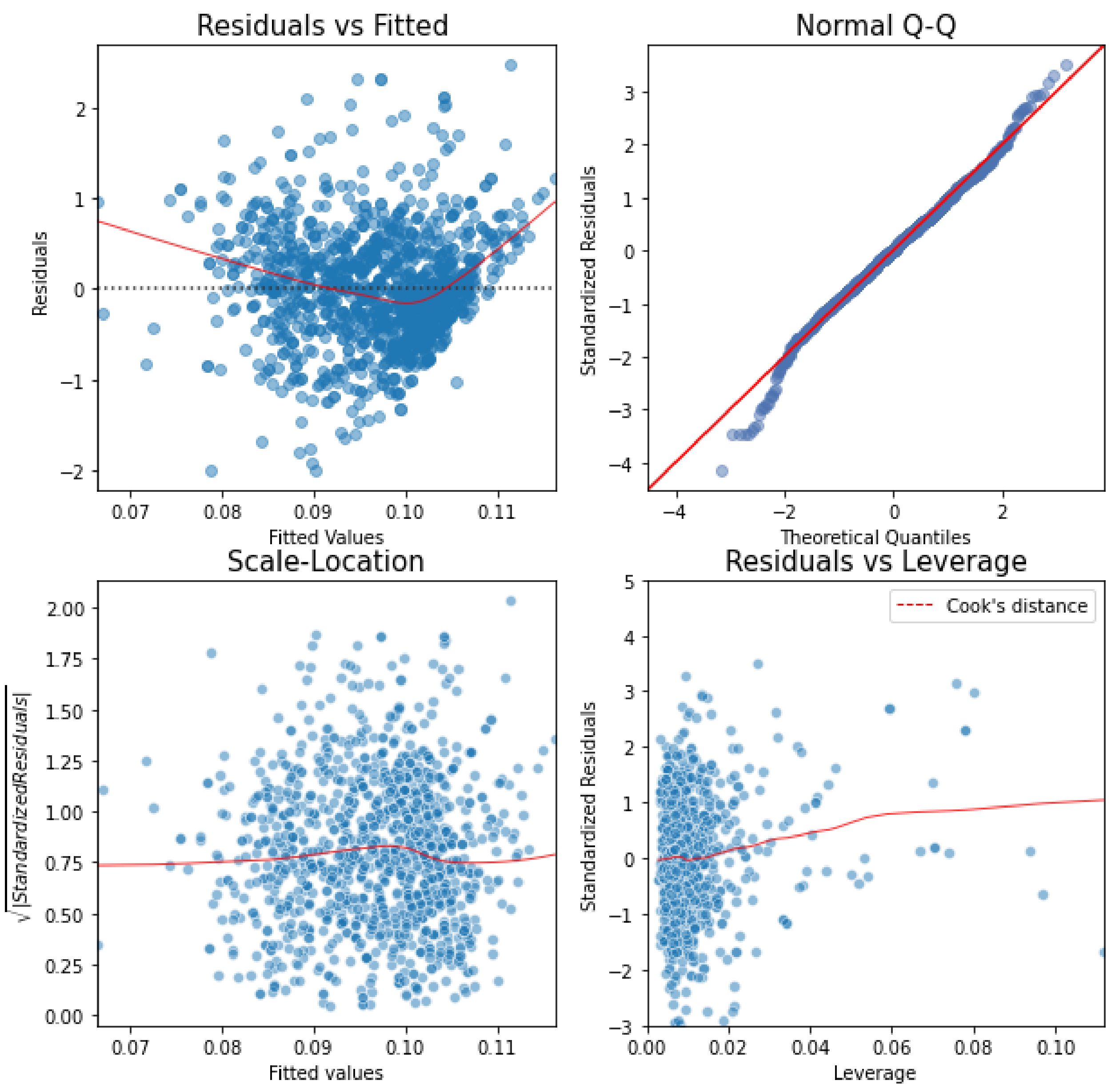
That value(s) of λ is (are) chosen for which the graph plotted above reaches its maximum. For this data the maximum is attained somewhere between −1 and −2. For clarity of interpretation, any fractional value of λ is ignored.

Taking $\lambda = -1$, the response is transformed to get new_resp1 and is modelled by MLR on the predictors from the training dataset.

```
#lambda=-1
new_resp1=1/train_WineData['alcohol']
train_WineData['new_resp'] = new_resp1

MLR_wine1 =
ols(formula="new_resp~FA+VA+CA+RS+chloride+FSD+TSD+density+sulphate+pH+Seagram+SulaVineyar
ds",data=train_WineData).fit()

regression_plots(MLR_wine1,train_WineData) # code for this method is made available in the
Regression Monograph
```



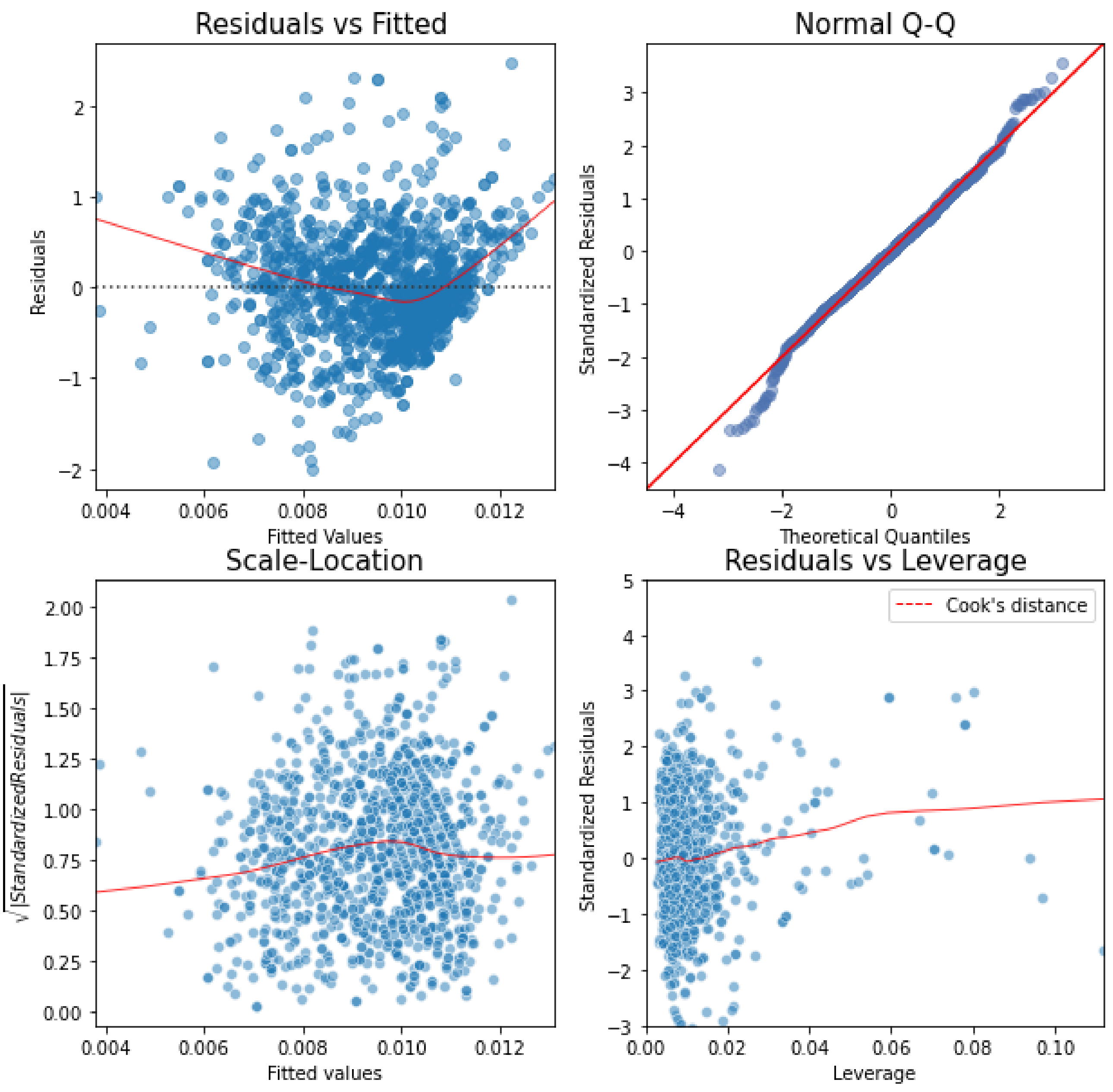
It is clear that the regression assumptions are not violated. Next, $\lambda = -2$ transformation is considered.

```
#lambda=-2
```

```
new_resp2=1/train_WineData['alcohol']**2
train_WineData['new_resp'] = new_resp2

MLR_wine2 =
ols(formula="new_resp~FA+VA+CA+RS+chloride+FSD+TSD+density+sulphate+pH+Seagram+SulaVineyar
ds",data=train_WineData).fit()

regression_plots(MLR_wine2,train_WineData)
```



The regression assumptions are not violated in this case also. So both the transformations with λ=−1 and λ=−2 may be considered. Whereas λ=−1 involves taking the reciprocal of the response (alcohol), λ=−2 implies squaring the response (alcohol) and taking its reciprocal. This extra step, in absence of any definite improvement, lacks interpretability. Therefore, λ=−1 transformation is chosen.

In all the subsequent discussions, the transformation of the response with $\lambda = -1$ has been used.

```
#lambda=-1
new_resp=1/train_WineData['alcohol']
train_WineData['new_resp'] = new_resp
MLR_wine =
ols(formula="new_resp~FA+VA+CA+RS+chloride+FSD+TSD+density+sulphate+pH+Seagram+SulaVineyards",
data=train_WineData).fit()
print(MLR_wine.summary())

OLS Regression Results
==============================================================================
Dep. Variable:               new_resp   R-squared:                       0.658
Model:                            OLS   Adj. R-squared:                  0.655
Method:                 Least Squares   F-statistic:                     202.8
Date:                Wed, 06 Jan 2021   Prob (F-statistic):          1.83e-284
Time:                        14:57:09   Log-Likelihood:                 4871.4
No. Observations:                1279   AIC:                            -9717.
Df Residuals:                    1266   BIC:                            -9650.
Df Model:                          12
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
Intercept        -4.6667      0.134    -34.741      0.000      -4.930      -4.403
FA               -0.0042      0.000    -20.435      0.000      -0.005      -0.004
VA               -0.0040      0.001     -3.498      0.000      -0.006      -0.002
CA               -0.0076      0.001     -5.582      0.000      -0.010      -0.005
RS               -0.0023      0.000    -18.613      0.000      -0.002      -0.002
chloride          0.0164      0.004      4.357      0.000       0.009       0.024
FSD            1.723e-05   2.07e-05      0.832      0.406   -2.34e-05    5.79e-05
TSD            2.061e-05   6.91e-06      2.983      0.003    7.06e-06    3.42e-05
density           4.9303      0.138     35.752      0.000       4.660       5.201
sulphate         -0.0098      0.001     -9.566      0.000      -0.012      -0.008
pH               -0.0311      0.002    -20.252      0.000      -0.034      -0.028
Seagram           0.0020      0.000      4.738      0.000       0.001       0.003
SulaVineyards    -0.0002      0.000     -0.579      0.563      -0.001       0.001
==============================================================================
Omnibus:                       29.494   Durbin-Watson:                   1.998
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               48.885
Skew:                          -0.185   Prob(JB):                     2.42e-11
Kurtosis:                       3.883   Cond. No.                     7.67e+04
==============================================================================
```

Using the training data, we get the following regression equation:

$1/Alcohol = -4.667 - 0.004 \times FA - 0.004 \times VA - 0.0076 \times CA - 0.002 \times RS + 0.016 \times chloride + 0.00001 \times FSD + 0.00002 \times TSD + 4.93 \times density - 0.0098 \times sulphate - 0.031 \times pH + 0.002 \times I(Brand = Seagram) - 0.0002 \times I(Brand = Sula\ Vineyards)$

Note that the predictors FSD and $I(Brand = Sula\ Vineyards)$ are not significant in predicting 1/alcohol.

Earlier, the non-significant predictors were simply eliminated (see Regression monograph). Now, more formal statistical methods will be used to decide which variable(s) should be eliminated or retained in the final predictive model.

3.2 *Information criteria: AIC and BIC*

Once several candidate models are found, it is necessary to be able to compare among them. There are several options for comparison. Two of those, namely R^2 and adjusted R^2, have already been introduced (see Regression Monograph) and their merits and demerits have been discussed.

In this section two more criteria are introduced both of which are based on information lost in fitting a model. A model is a simplification of the process from which the observed data is generated. The closer the model is to the real process; the more information it contains. Nevertheless, no model will ever be able to emulate the real process and hence, some amount of information will always be lost. The errors or residuals of the model fit provide quantification of the information lost.

Both the information criteria, AIC and BIC, are popular for comparison of models. Both criteria are based on the error sum of squares and both penalize models on the number of predictors included. In a way, they try to strike a balance between bias and variance.

Consider the multiple linear regression model with p parameters (including intercept term) on n data points and let the residual sum of squares be denoted by SSE.

The ***Akaike Information Criteria (AIC)*** is defined by

$$AIC = n\log(SSE) - n\log(n) + 2p$$

However, the AIC tends to overfit models. This criticism of AIC has led to the development of BIC.

The ***Bayesian Information Criteria (BIC)*** is defined by

$$BIC = n\log(SSE) - n\log(n) + p\log(n)$$

Since $\log(n)$ is usually much larger than 2, it follows that BIC imposes greater penalty if the number of parameters, p, is too large. Thus BIC maintains a greater balance in the number of parameters used to fit the model. In general, BIC is preferred to AIC in model building exercises.

Naturally, the smaller the value of AIC or BIC, the better is the model, since the model with minimum value of these criteria identifies the smallest quantity of information lost.

Next the alternative model selection procedures are discussed.

3.3 Forward Selection (FS) Algorithm

Forward selection algorithm for regression model building is an automated algorithm which selects one predictor at a time, conditional on which other predictors are already included in the model. This model is not expected to include the redundant predictors. Forward selection algorithm suggests a minimal set of predictors following certain optimality criteria.

The forward selection algorithm is one of the most popular algorithm because of its easy interpretability.

Assume there are p available predictors. The steps of forward selection algorithm are as follows:

- Start with a null (intercept-only) model.
- Perform p simple linear regressions with one predictor at a time. For each model, compute the AIC value. Include that predictor for which the corresponding model has the smallest AIC value, smaller than the null model. If no such predictor is found, then the null model is considered final.
- Assume that a certain predictor, say X_1, did enter the model. At the end of this step the model contains an intercept term and X_1.
- Perform p-1 multiple linear regressions with the remaining p-1 predictors $X_2,\ldots,X_p$,
- Check which predictor among $X_2,\ldots,X_p$ gives smallest AIC in presence of X_1, smaller than the AIC obtained with just X_1 as predictor. That predictor is included in the model. If no predictor among $X_2,\ldots,X_p$ satisfies the entry criterion, then the process is stopped and the final model includes X_1 as the only predictor for Y
- This process is continued till either all predictors are included in the regression or a subset of the predictors is included and inclusion of no other predictor found to give smaller AIC value than that of the model determined by this subset.

Forward selection is performed on the training dataset with response new_resp.

```
# forward selection
new_resp=1/train_WineData['alcohol']
train_WineData['new_resp'] = new_resp
train_X_WineData = train_WineData.drop(['alcohol','new_resp'],axis = 1)
def forwardSelection_aic(X, y):
    X["intercept"] = 1
    cols = X.columns.tolist()
    cols = cols [-1:] + cols[:-1]
    X = X[cols]

    iterations_log = ""
    cols = X.columns.tolist()

    selected_cols = ["intercept"]
    remaining_cols = cols.copy()
    remaining_cols.remove("intercept")
    print("Remaining columns : ",remaining_cols)
    model = sm.OLS(y, X[selected_cols]).fit()
    criteria = model.aic
    print("Null model AIC critetia :",criteria)
    while(len(remaining_cols) > 0):
        aic_dict = {}
        for i in range(len(remaining_cols)):
            new_col = remaining_cols[i]
            selected_cols.append(new_col)
            model = sm.OLS(y, X[selected_cols]).fit()
            new_criteria = model.aic
            if new_criteria < criteria:
                aic_dict[new_col] = new_criteria
            selected_cols.remove(new_col)
        if len(aic_dict) == 0:
            break
        entered_col = sorted(aic_dict.items(), key=lambda x: x[1])[0][0]
        print()
        print("Entered column :",entered_col)
        selected_cols.append(entered_col)
        print("Selected columns :",selected_cols)
        remaining_cols.remove(entered_col)
        print("Remaining columns : ",remaining_cols)
        model = sm.OLS(y, X[selected_cols]).fit()
        criteria = model.aic
        print("New AIC critetia :",criteria)
    print("\n\n")
    print("Final selected columns :",selected_cols)
    print("Final removed columns : ",remaining_cols)
    return selected_cols

frwd_slctn_cols = forwardSelection_aic(train_X_WineData,train_WineData['new_resp'
])
```

```
Remaining columns :  ['FA', 'VA', 'CA', 'RS', 'chloride', 'FSD', 'TSD', 'density'
, 'pH', 'sulphate', 'Seagram', 'SulaVineyards']
Null model AIC critetia : -8369.496080838044

Entered column : density
Selected columns : ['intercept', 'density']
Remaining columns :  ['FA', 'VA', 'CA', 'RS', 'chloride', 'FSD', 'TSD', 'pH', 'su
lphate', 'Seagram', 'SulaVineyards']
New AIC critetia : -8686.871678269768

Entered column : FA
Selected columns : ['intercept', 'density', 'FA']
Remaining columns :  ['VA', 'CA', 'RS', 'chloride', 'FSD', 'TSD', 'pH', 'sulphate
', 'Seagram', 'SulaVineyards']
New AIC critetia : -8896.704471654608

Entered column : pH
Selected columns : ['intercept', 'density', 'FA', 'pH']
Remaining columns :  ['VA', 'CA', 'RS', 'chloride', 'FSD', 'TSD', 'sulphate', 'Se
agram', 'SulaVineyards']
New AIC critetia : -9202.05736648084

Entered column : RS
Selected columns : ['intercept', 'density', 'FA', 'pH', 'RS']
Remaining columns :  ['VA', 'CA', 'chloride', 'FSD', 'TSD', 'sulphate', 'Seagram'
, 'SulaVineyards']
New AIC critetia : -9509.221077244645

Entered column : sulphate
Selected columns : ['intercept', 'density', 'FA', 'pH', 'RS', 'sulphate']
Remaining columns :  ['VA', 'CA', 'chloride', 'FSD', 'TSD', 'Seagram', 'SulaViney
ards']
New AIC critetia : -9625.470158323134

Entered column : Seagram
```

```
Selected columns : ['intercept', 'density', 'FA', 'pH', 'RS', 'sulphate', 'Seagra
m']
Remaining columns :  ['VA', 'CA', 'chloride', 'FSD', 'TSD', 'SulaVineyards']
New AIC critetia : -9674.361041210175

Entered column : FSD
Selected columns : ['intercept', 'density', 'FA', 'pH', 'RS', 'sulphate', 'Seagra
m', 'FSD']
Remaining columns :  ['VA', 'CA', 'chloride', 'TSD', 'SulaVineyards']
New AIC critetia : -9685.905253086048

Entered column : CA
Selected columns : ['intercept', 'density', 'FA', 'pH', 'RS', 'sulphate', 'Seagra
m', 'FSD', 'CA']
Remaining columns :  ['VA', 'chloride', 'TSD', 'SulaVineyards']
New AIC critetia : -9696.160147228982

Entered column : chloride
Selected columns : ['intercept', 'density', 'FA', 'pH', 'RS', 'sulphate', 'Seagra
m', 'FSD', 'CA', 'chloride']
Remaining columns :  ['VA', 'TSD', 'SulaVineyards']
New AIC critetia : -9704.733713671727

Entered column : VA
Selected columns : ['intercept', 'density', 'FA', 'pH', 'RS', 'sulphate', 'Seagra
m', 'FSD', 'CA', 'chloride', 'VA']
Remaining columns :  ['TSD', 'SulaVineyards']
New AIC critetia : -9711.47318715933

Entered column : TSD
Selected columns : ['intercept', 'density', 'FA', 'pH', 'RS', 'sulphate', 'Seagra
m', 'FSD', 'CA', 'chloride', 'VA', 'TSD']
Remaining columns :  ['SulaVineyards']
New AIC critetia : -9718.548176185843


Final selected columns : ['intercept', 'density', 'FA', 'pH', 'RS', 'sulphate', '
Seagram', 'FSD', 'CA', 'chloride', 'VA', 'TSD']
Final removed columns :  ['SulaVineyards']
```

Start with the null model (*null_mod*) that has only the intercept term as the predictor. If at any stage, it is seen that inclusion of a new predictor does not improve the AIC value, the process stops.

The null model has AIC −8369.496. Starting with null model, density has the smallest AIC value. The model containing density has AIC value −8686.871, which is an improvement on the previous AIC value of −8369.496. Therefore, density enters the model.

Similarly, once density is present in the model, the best improvement in AIC value is obtained by including FA. So at the next step, the model includes both density and FA. In this way, the process continues till the full model is obtained.

For this data set, all predictors except "SulaVineyards" enter the model through forward selection procedure.

Therefore, the regression equation also gets updated as the feature "SulaVineyards" will be removed from the equation. The regression equation will look as follows :

```
MLR_wine1 =
ols(formula="new_resp~FA+VA+CA+RS+chloride+FSD+TSD+density+sulphate+pH+Seagram",data=train_
WineData).fit()
```

```
print(MLR_wine1.summary())


OLS Regression Results
==============================================================================
Dep. Variable:               new_resp   R-squared:                       0.658
Model:                            OLS   Adj. R-squared:                  0.655
Method:                 Least Squares   F-statistic:                     221.3
Date:                Sun, 10 Jan 2021   Prob (F-statistic):         1.42e-285
Time:                        15:37:00   Log-Likelihood:                 4871.3
No. Observations:                1279   AIC:                            -9719.
Df Residuals:                    1267   BIC:                            -9657.
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -4.6612      0.134    -34.798      0.000      -4.924      -4.398
FA            -0.0042      0.000    -20.447      0.000      -0.005      -0.004
VA            -0.0040      0.001     -3.485      0.001      -0.006      -0.002
CA            -0.0076      0.001     -5.559      0.000      -0.010      -0.005
RS            -0.0022      0.000    -18.620      0.000      -0.002      -0.002
chloride       0.0163      0.004      4.341      0.000       0.009       0.024
FSD         1.621e-05   2.06e-05      0.786      0.432   -2.43e-05    5.67e-05
TSD         2.073e-05    6.9e-06      3.004      0.003    7.19e-06    3.43e-05
density        4.9243      0.137     35.822      0.000       4.655       5.194
sulphate      -0.0098      0.001     -9.554      0.000      -0.012      -0.008
pH            -0.0310      0.002    -20.286      0.000      -0.034      -0.028
Seagram        0.0022      0.000      6.264      0.000       0.001       0.003
==============================================================================
Omnibus:                       29.768   Durbin-Watson:                   1.999
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               49.688
Skew:                          -0.185   Prob(JB):                     1.62e-11
Kurtosis:                       3.892   Cond. No.                     7.65e+04
==============================================================================
```

$1/Alcohol = -4.661 - 0.0042 \times FA - 0.004 \times VA - 0.0076 \times CA - 0.0022\times RS + 0.0163 \times chloride + 0.000016 \times FSD + 0.00002 \times TSD + 4.924 \times density - 0.0098 \times sulphate - 0.031 \times pH + 0.0022 \times I(Brand = Seagram)$

3.4 Backward Elimination (BE) Algorithm

Forward selection includes the predictors one by one into the model. Backward elimination is an algorithm that goes the opposite way. The algorithm is described below.

- Start with the full model.
- Remove each predictor from the model and compute the AIC value of each of these new sub-models. That sub-model giving smallest AIC value, smaller than the AIC value of the full model, is selected. That is, the predictor whose absence created this selected sub-model, is eliminated. If this condition is not satisfied, the process stops and the full model is chosen to be the final model.
- Suppose at the first step a certain variable X_1 is eliminated from the model. The new *(p - 1)* - variable model is considered again and the variable showing smallest AIC value at that stage is removed from the model if the AIC value falls below even that of the *(p −1)* - variable model. Again if no variable is removed, conclude that the $p - 1$-variable

model is the final model.

- Continue until there is no predictor left in the model, i.e. the model becomes a null model or if there is no further improvement in AIC values.

The code and output for backward elimination procedure are given below. The process starts with the full model.

It was seen that the variable satisfying this condition was SulaVineyards. The AIC of the model obtained by removing SulaVineyards is −9718.548 which is smaller than AIC of the full model (−9716.88). Therefore, SulaVineyards was eliminated at the first stage. Similarly, in second stage, the AIC of the model obtained by removing FSD is −9719.924 which is smaller than AIC of the previous model (−9718.548). However, after removing FSD, no other predictor satisfied the exit criterion. The process stopped, keeping all the predictors except SulaVineyards & FSD in the model.

```
def backwardElimination_aic(X, y):
    X["intercept"] = 1
    cols = X.columns.tolist()
    cols = cols[-1:] + cols[:-1]
    X = X[cols]

    iterations_log = ""
    cols = X.columns.tolist()

    selected_cols = cols
    remaining_cols = cols.copy()
    remaining_cols.remove("intercept")
    model = sm.OLS(y, X[selected_cols]).fit()
    criteria = model.aic
    print("Initial AIC critetia :",criteria)

    while(True):
        rmvd_col_dict = {}
        for i in range(len(remaining_cols)):
            temp_rmvd_col = remaining_cols[i]
            selected_cols.remove(temp_rmvd_col)
            model = sm.OLS(y, X[selected_cols]).fit()
            new_criteria = model.aic
            selected_cols.append(temp_rmvd_col)
            if new_criteria < criteria:
                rmvd_col_dict[temp_rmvd_col] = new_criteria
        if len(rmvd_col_dict) == 0:
            break
        print()
        print("Columns eligible for elimination:",rmvd_col_dict)
        eliminated_col = sorted(rmvd_col_dict.items(), key=lambda x: x[1])[0][0]
        print("Removed col with least AIC:",eliminated_col)
        selected_cols.remove(eliminated_col)
        remaining_cols.remove(eliminated_col)
        model = sm.OLS(y, X[selected_cols]).fit()
        criteria = model.aic
        print("Updated AIC criteria : ",criteria)
        print(selected_cols)

    print("\n\n")
    print("Final selected columns :",selected_cols)
    return selected_cols

bckwrd_elmtn_cols = backwardElimination_aic(train_X_WineData,train_WineData['new_resp'])
```

```
Initial AIC critetia : -9716.886654499593

Columns eligible for elimination: {'FSD': -9718.187479229922, 'SulaVineyards': -
9718.548176185837}
Removed col with least AIC: SulaVineyards
Updated AIC criteria :  -9718.548176185837
['intercept', 'FA', 'VA', 'CA', 'RS', 'chloride', 'FSD', 'TSD', 'density', 'pH',
'sulphate', 'Seagram']

Columns eligible for elimination: {'FSD': -9719.924834061516}
Removed col with least AIC: FSD
Updated AIC criteria :  -9719.92483406152
['intercept', 'FA', 'VA', 'CA', 'RS', 'chloride', 'TSD', 'density', 'pH', 'sulphate',
'Seagram']


Final selected columns : ['intercept', 'FA', 'VA', 'CA', 'RS', 'chloride', 'TSD',
'density', 'pH', 'sulphate', 'Seagram']
```

Backward elimination suggests that FA, VA, CA, RS, chloride, TSD, density, sulphate, pH, and Seagram are enough to be used as predictors and in their presence the variable FSD and SulaVineyards is redundant

```
MLR_wine1 =
ols(formula="new_resp~FA+VA+CA+RS+chloride+TSD+density+sulphate+pH+Seagram",data=train_Wine
Data).fit()
print(MLR_wine1.summary())

OLS Regression Results
==============================================================================
Dep. Variable:               new_resp   R-squared:                       0.658
Model:                            OLS   Adj. R-squared:                  0.655
Method:                 Least Squares   F-statistic:                     243.4
Date:                Sun, 10 Jan 2021   Prob (F-statistic):          1.21e-286
Time:                        15:53:56   Log-Likelihood:                 4871.0
No. Observations:                1279   AIC:                            -9720.
Df Residuals:                    1268   BIC:                            -9663.
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -4.6515      0.133    -34.877      0.000      -4.913      -4.390
FA            -0.0042      0.000    -20.494      0.000      -0.005      -0.004
VA            -0.0041      0.001     -3.661      0.000      -0.006      -0.002
CA            -0.0078      0.001     -5.806      0.000      -0.010      -0.005
RS            -0.0022      0.000    -18.700      0.000      -0.002      -0.002
chloride       0.0165      0.004      4.418      0.000       0.009       0.024
TSD         2.439e-05    5.1e-06      4.779      0.000    1.44e-05    3.44e-05
density        4.9141      0.137     35.913      0.000       4.646       5.182
sulphate      -0.0097      0.001     -9.532      0.000      -0.012      -0.008
pH            -0.0309      0.002    -20.365      0.000      -0.034      -0.028
Seagram        0.0022      0.000      6.225      0.000       0.001       0.003
==============================================================================
Omnibus:                       30.227   Durbin-Watson:                   1.999
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               50.976
Skew:                          -0.184   Prob(JB):                     8.52e-12
Kurtosis:                       3.906   Cond. No.                     7.30e+04
==============================================================================
```

The regression equation becomes

$1/Alcohol = -4.65 - 0.0042 \times FA - 0.0041 \times VA - 0.0078 \times CA - 0.0022\times RS + 0.0165 \times chloride + 0.000024 \times TSD + 4.924 \times density - 0.0097 \times sulphate - 0.0309 \times pH + 0.0022 \times I(Brand = Seagram)$

Several important points need to be noted here.

1. At every stage of FS or BE procedure, the candidate predictor to enter or to be removed, depends on the set of predictors already in the model. Both these procedures may be modified and one or more predictors may be forced to be included in the model, even if they are not otherwise significant. Under such condition, the final model will be different than if the process is continued without any condition.
2. There is no guarantee that FS and BE will propose identical model. The current data set is an example of that.
3. While the final model is developed using AIC, none of the other model optimality criteria, such as multiple R^2 or adjusted R^2 is used. Alternative criteria, such as BIC or adjusted R^2 can also be used to control entry and exit of predictors. Use of different criteria will result in different models.

3.5 Stepwise Regression

Stepwise regression is a combination of Forward Selection and Backward Elimination procedures.

- Start with the null model.
- At the next step, use Forward Selection to select a predictor showing smallest AIC value, smaller than that of the null model. If this criterion is not satisfied, then the null model is selected to be the final model.
- Suppose a predictor X_1 entered the model. For each of the remaining predictors, include it, compute AIC. Remove X_1 from the model and also compute the AIC value. Print the variables in increasing order of AIC values. If there is a variable whose entry/exit improves the AIC of the existing model, include/eliminate it from the model.
- Continue till all predictors are included or there is no further improvement of AIC.

It was seen that the outcome of Stepwise Regression was identical to that of Backward Elimination, i.e. only SulaVineyards & FSD was removed from the set of predictors to build the final model.

```
def combine_fs_be_stepaic(X,y):
    X["intercept"] = 1
    cols = X.columns.tolist()
    cols = cols[-1:] + cols[:-1]
    X = X[cols]

    iterations_log = ""
    cols = X.columns.tolist()

    selected_cols = ["intercept"]
    remaining_cols = cols.copy()
    remaining_cols.remove("intercept")
    print("Remaining columns : ",remaining_cols)
    model = sm.OLS(y, X[selected_cols]).fit()
    criteria = model.aic
    print("Null model AIC critetia :",criteria)

    temp_dict = {}
    for i in range(len(remaining_cols)):
        new_col = remaining_cols[i]
        selected_cols.append(new_col)
        model = sm.OLS(y, X[selected_cols]).fit()
        new_criteria = model.aic
        if new_criteria < criteria:
            temp_dict[new_col] = new_criteria
        selected_cols.remove(new_col)
    entered_col = sorted(temp_dict.items(), key=lambda x: x[1])[0][0]
    entered_aic = sorted(temp_dict.items(), key=lambda x: x[1])[0][1]
    print()
    print("Entered column :",entered_col)
    selected_cols.append(entered_col)
    temp_dict = {}
    temp_dict[entered_col] = entered_aic
    first_stage = True
    print("Selected columns :",selected_cols)
    remaining_cols.remove(entered_col)
    print(temp_dict)

    while(True):
        if first_stage:
            aic_dict = temp_dict
        else:
            first_stage = False
            aic_dict = {}
        flag = False
        be_remaining_cols = selected_cols.copy()
        be_selected_cols = selected_cols.copy()
        be_remaining_cols.remove('intercept')
        for i in range(len(be_remaining_cols)):
            temp_rmvd_col = be_remaining_cols[i]
            be_selected_cols.remove(temp_rmvd_col)
            model = sm.OLS(y, X[be_selected_cols]).fit()
            new_criteria = model.aic
            be_selected_cols.append(temp_rmvd_col)
            if new_criteria < criteria:
                aic_dict[temp_rmvd_col] = new_criteria
                flag = True
        print()
        col_elig_fr_elim = None
        if flag :
            col_elig_fr_elim = sorted(aic_dict.items(), key=lambda x: x[1])[0][0]
            print("Column that is eligible for elimination:",col_elig_fr_elim)
        else:
            print('No colums eligible for BE in this stage')

        for i in range(len(remaining_cols)):
            new_col = remaining_cols[i]
            selected_cols.append(new_col)
            model = sm.OLS(y, X[selected_cols]).fit()
            new_criteria = model.aic
            if new_criteria < criteria:
                aic_dict[new_col] = new_criteria
            selected_cols.remove(new_col)
        lowest_aic_col = sorted(aic_dict.items(), key=lambda x: x[1])[0][0]
        prev_criteria = criteria
        if lowest_aic_col == col_elig_fr_elim:
            selected_cols.remove(lowest_aic_col)
            remaining_cols.append(lowest_aic_col)
            model = sm.OLS(y, X[selected_cols]).fit()
            criteria = model.aic
            if prev_criteria<=criteria:
                selected_cols.append(lowest_aic_col)
                remaining_cols.remove(lowest_aic_col)
                break
            print("Eliminated column :",lowest_aic_col)
        else:
            selected_cols.append(lowest_aic_col)
            remaining_cols.remove(lowest_aic_col)
            model = sm.OLS(y, X[selected_cols]).fit()
            criteria = model.aic
            if prev_criteria<=criteria:
                selected_cols.remove(lowest_aic_col)
                remaining_cols.append(lowest_aic_col)
                break
            print("Entered column :",lowest_aic_col)
        print("Selected columns :",selected_cols)
        print("Update AIC criteria : ",criteria)

    return selected_cols


combine_fs_be_stepaic(train_X_WineData,train_WineData['new_resp'])
```

```
Remaining columns :  ['FA', 'VA', 'CA', 'RS', 'chloride', 'FSD', 'TSD', 'density', 'pH', 'su
lphate', 'Seagram', 'SulaVineyards']
Null model AIC critetia : -8369.496080838044

Entered column : density
Selected columns : ['intercept', 'density']
{'density': -8686.871678269768}

No colums eligible for BE in this stage
Entered column : FA
Selected columns : ['intercept', 'density', 'FA']
Update AIC criteria :  -8896.704471654608

No colums eligible for BE in this stage
Entered column : pH
Selected columns : ['intercept', 'density', 'FA', 'pH']
Update AIC criteria :  -9202.05736648084

No colums eligible for BE in this stage
Entered column : RS
Selected columns : ['intercept', 'density', 'FA', 'pH', 'RS']
Update AIC criteria :  -9509.221077244645

No colums eligible for BE in this stage
Entered column : sulphate
Selected columns : ['intercept', 'density', 'FA', 'pH', 'RS', 'sulphate']
Update AIC criteria :  -9625.470158323134

No colums eligible for BE in this stage
Entered column : Seagram
Selected columns : ['intercept', 'density', 'FA', 'pH', 'RS', 'sulphate', 'Seagram']
Update AIC criteria :  -9674.361041210175
```

```
No colums eligible for BE in this stage
Entered column : FSD
Selected columns : ['intercept', 'density', 'FA', 'pH', 'RS', 'sulphate', 'Seagram', 'FSD']
Update AIC criteria :  -9685.905253086048

No colums eligible for BE in this stage
Entered column : CA
Selected columns : ['intercept', 'density', 'FA', 'pH', 'RS', 'sulphate', 'Seagram', 'FSD',
'CA']
Update AIC criteria :  -9696.160147228982

No colums eligible for BE in this stage
Entered column : chloride
Selected columns : ['intercept', 'density', 'FA', 'pH', 'RS', 'sulphate', 'Seagram', 'FSD',
'CA', 'chloride']
Update AIC criteria :  -9704.733713671727

No colums eligible for BE in this stage
Entered column : VA
Selected columns : ['intercept', 'density', 'FA', 'pH', 'RS', 'sulphate', 'Seagram', 'FSD',
'CA', 'chloride', 'VA']
Update AIC criteria :  -9711.47318715933

No colums eligible for BE in this stage
Entered column : TSD
Selected columns : ['intercept', 'density', 'FA', 'pH', 'RS', 'sulphate', 'Seagram', 'FSD',
'CA', 'chloride', 'VA', 'TSD']
Update AIC criteria :  -9718.548176185843

Column that is eligible for elimination: FSD
Eliminated column : FSD
Selected columns : ['intercept', 'density', 'FA', 'pH', 'RS', 'sulphate', 'Seagram', 'CA', '
chloride', 'VA', 'TSD']
Update AIC criteria :  -9719.924834061512

Final selected columns : ['intercept', 'density', 'FA', 'pH', 'RS', 'sulphate', 'Seagram', '
CA', 'chloride', 'VA', 'TSD']
```

The regression equation therefore becomes

$$1/Alcohol = -4.65 - 0.0042 \times FA - 0.0041 \times VA - 0.0078 \times CA - 0.0022\times RS + 0.0165 \times chloride + 0.000024 \times TSD + 4.924 \times density - 0.0097 \times sulphate - 0.0309 \times pH + 0.0022 \times I(Brand = Seagram)$$

No further discussion with Stepwise Regression is taken up since its output is identical to that of Backward Elimination. All computations performed with the model obtained from Backward Elimination will be identical with the model obtained from Stepwise Regression.

3.6 All Possible Regression or Regression Subset Selection and Mallows' Cp Criterion

The statistic known as the Mallows' *Cp* criterion is useful to measure bias in a multiple linear regression setting.

Consider as before a multiple linear regression model where the response is Y and there exists a set of predictors $\{X_1, X_2, \ldots, X_k\}$. With k predictors, 2^k different multiple linear regression models may be fit. For example, if $k = 10$, $2^{10} = 1024$ different multiple linear regression models (including the null model) may be fit.

If a set of *p-1* predictors is chosen out of this set so that there are in total p parameters (including intercept), then Mallows' C_p is computed by

$$C_p=[SSE_p/MSE_{all}] - (n-2p)$$

where n is the number of observations in the data set. Here SSE_p is the residual sum of squares obtained when the response is modelled with the p-1 chosen predictors.

If the full model is fitted, using all the k predictors, then there will be k+1 parameters. Then MSE_{all} or Mean Squared Error for the full model is computed by dividing the residual sum of squares (SSE) by $n-(k+1)$.

If the model works well, the numerical value of C_p is expected to be close to p. For a potentially good model $C_p \leq$ p. The full model always has Mallows' C_p equal to k+1. Aim is to select the smallest p and that subset of predictors for which C_p is smaller than but closest to p.

This procedure is also known as All Possible Regressions or Regression Subset Selection. However, for even a decent sized k, all 2^k subset of predictors are not possible to consider. Typically, only a few models at various values of k are compared.

One advantage of this procedure is that, the models are not conditional on what predictors are already included in the model. For the algorithms discussed previously, i.e. FS, BE or stepwise regression, the next predictor to be included or eliminated, *depends on* what other predictors are already included in the model. In All Possible Regression method, there does not exist any such dependency.

```
import os
from sklearn.linear_model import LinearRegression
from mlxtend.feature_selection import ExhaustiveFeatureSelector as EFS

X=train_X_WineData
y=train_WineData['new_resp']
# Perform an Exhaustive Search. The EFS and SFS packages use 'neg_mean_squared_error'. The 
'mean_squared_error' seems to have been deprecated. I think this is just the MSE with the 
a negative sign.
lr = LinearRegression()
efs1 = EFS(lr,
           min_features=1,
           max_features=13,
           scoring='neg_mean_squared_error',
           print_progress=True,
           cv=5)
# Create a efs fit
efs1 = efs1.fit(X.values, y.values)
print('Best negtive mean squared error: %.2f' % efs1.best_score_)
## Print the IDX of the best features
print('Best subset:', efs1.best_idx_)
```

```
Features: 8191/8191
```

```
Best negtive mean squared error: -0.00
Best subset: (0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 12)


X.columns[[0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 12]]
Index(['FA', 'VA', 'CA', 'RS', 'chloride', 'TSD', 'density', 'pH', 'sulphate',
       'Seagram', 'intercept'], dtype='object')
```

There are 12 predictors (treating BrandSeagram and BrandSula Vineyards as separate predictors), and so, along with the intercept term, k = 13. The model with 1 parameter is the null model containing only the intercept term, and the model with 13 parameters contains all the predictors along with the intercept term.

At every choice of the number of predictors (ranging from 1 to 12), the 'best' model according to Mallows' *Cp* criterion is calculated.

The best model with 11 predictors is therefore given by the model that includes all variables and intercept except FSD and Brand SulaVineyards.

```
MLR_wine1                                                                                    =
ols(formula="new_resp~FA+VA+CA+RS+chloride+TSD+density+sulphate+pH+Seagram",data=train_W
ineData).fit()
print(MLR_wine1.summary())
OLS Regression Results
==============================================================================
Dep. Variable:               new_resp   R-squared:                       0.658
Model:                            OLS   Adj. R-squared:                  0.655
Method:                 Least Squares   F-statistic:                     243.4
Date:                Sun, 10 Jan 2021   Prob (F-statistic):         1.21e-286
Time:                        15:53:56   Log-Likelihood:                 4871.0
No. Observations:                1279   AIC:                            -9720.
Df Residuals:                    1268   BIC:                            -9663.
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -4.6515      0.133    -34.877      0.000      -4.913      -4.390
FA            -0.0042      0.000    -20.494      0.000      -0.005      -0.004
VA            -0.0041      0.001     -3.661      0.000      -0.006      -0.002
CA            -0.0078      0.001     -5.806      0.000      -0.010      -0.005
RS            -0.0022      0.000    -18.700      0.000      -0.002      -0.002
chloride       0.0165      0.004      4.418      0.000       0.009       0.024
TSD         2.439e-05    5.1e-06      4.779      0.000    1.44e-05    3.44e-05
density        4.9141      0.137     35.913      0.000       4.646       5.182
sulphate      -0.0097      0.001     -9.532      0.000      -0.012      -0.008
pH            -0.0309      0.002    -20.365      0.000      -0.034      -0.028
Seagram        0.0022      0.000      6.225      0.000       0.001       0.003
==============================================================================
Omnibus:                       30.227   Durbin-Watson:                   1.999
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               50.976
Skew:                          -0.184   Prob(JB):                     8.52e-12
Kurtosis:                       3.906   Cond. No.                     7.30e+04
==============================================================================
```

The regression equation becomes:

$1/Alcohol = -4.65 - 0.0042 \times FA - 0.0041 \times VA - 0.0078 \times CA - 0.0022\times RS + 0.0165 \times chloride + 0.000024 \times TSD + 4.924 \times density - 0.0097 \times sulphate - 0.0309 \times pH + 0.0022 \times I(Brand = Seagram)$

We can see that all the predictors are significant.

Note again that, all possible regression mechanism gives a third choice of model to predict alcohol (%).

3.7 Choosing the Final Model

Once one (or more) candidate models are identified using the training data, the final step in model selection procedure is validation on test data. In this case, all four models are considered for validation.

There are multiple criteria for model validation. The one chosen criterion is RMSE because of its easy interpretation.

```
from sklearn.metrics import mean_squared_error
# Forward selection
cols = ['density', 'FA', 'pH', 'RS', 'sulphate', 'Seagram', 'FSD', 'CA', 'chloride', 'VA',
'TSD']
X = train_X_WineData[cols]
X_test = test_WineData[cols]
y = train_WineData['new_resp']
y_test1 = 1/y_test
MLR_wine_fs = LinearRegression().fit(X,y)
y_pred = MLR_wine_fs.predict(X_test)
print("Forward Selection MSE :",np.sqrt(mean_squared_error(y_test1,y_pred)))
Forward Selection MSE : 0.005322077111287733

#Backward elimination / Both combination / Mallows Cp
cols = ['density', 'FA', 'pH', 'RS', 'sulphate', 'Seagram', 'CA', 'chloride', 'VA', 'TSD']
X = train_X_WineData[cols]
X_test = test_WineData[cols]
y = train_WineData['new_resp']
MLR_wine_be = LinearRegression().fit(X,y)
y_pred = MLR_wine_be.predict(X_test)
print("Backward Elimination / Both combination / Mallows Cp MSE
:",np.sqrt(mean_squared_error(y_test1,y_pred)))
Backward Elimination / Both combination / Mallows Cp MSE : 0.005317362696545827
```

As we know that Backward elimination, combination of backward & forward as well as Mallows Cp results in same features as the best features, we have calculated same RMSE for all 3 as shown above.

Both RMSE values are again comparable, but this time, the model obtained from Backward elimination, combination of backward & forward as well as Mallows Cp beats the other forward selection model.

Hence, we choose the model from Backward Elimination as our final model.

The final regression equation is

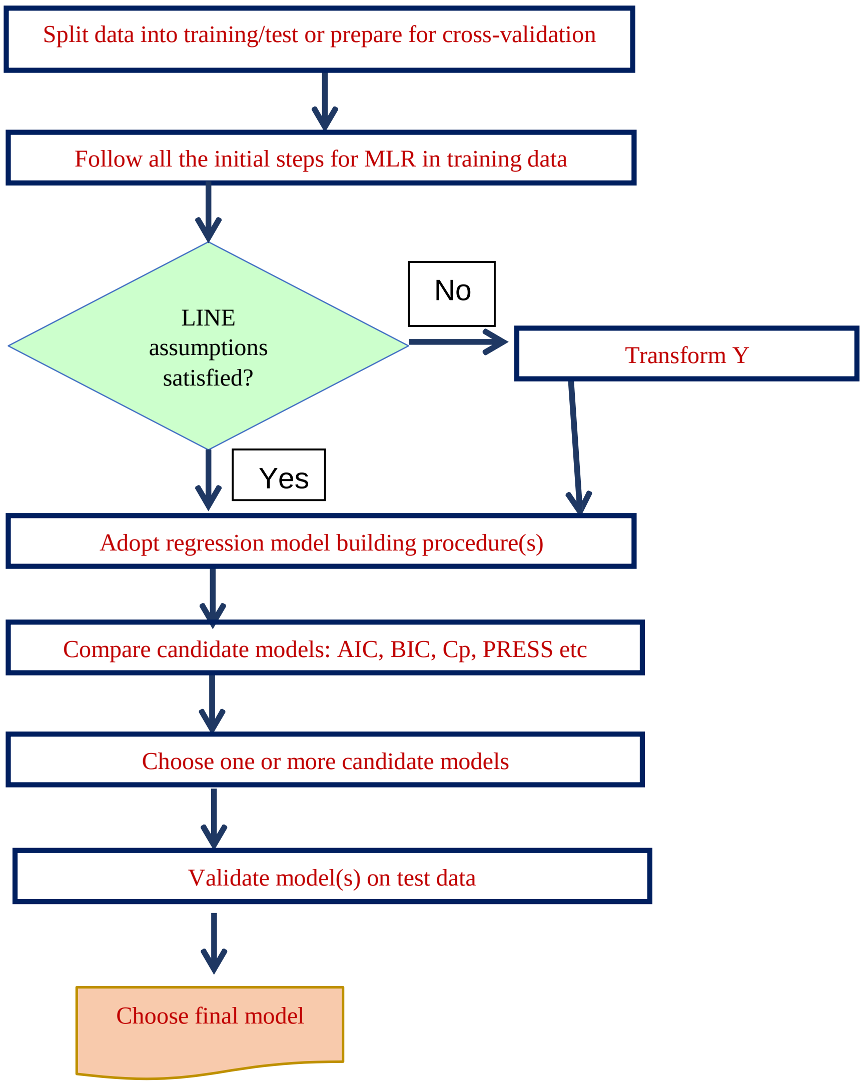
$1/Alcohol = -4.65 - 0.0042 \times FA - 0.0041 \times VA - 0.0078 \times CA - 0.0022\times RS + 0.0165 \times chloride + 0.000024 \times TSD + 4.924 \times density - 0.0097 \times sulphate - 0.0309 \times pH + 0.0022 \times I(Brand = Seagram)$

Note that the RMSE for this model is 0.00531 on the test data, and the RMSE for the same model on the train data is approximately 0.00537.

```
y_train_pred = MLR_wine_be.predict(X)
print("Train MSE :",np.sqrt(mean_squared_error(y,y_train_pred)))
Train MSE : 0.005367713537929523
```

The comparability of these two values suggests that our final model is unbiased and hence a good fit.

Flowchart for Regression Model Selection


Split data into training/test or prepare for cross-validation
Follow all the initial steps for MLR in training data
LINE assumptions satisfied?
No
Transform Y
Yes
Adopt regression model building procedure(s)
Compare candidate models: AIC, BIC, Cp, PRESS etc
Choose one or more candidate models
Validate model(s) on test data
Choose final model

References

Draper, N. R., Smith, H. (1998). Applied Regression Analysis. Wiley Series in Probability and Statistics.

Neter, J., Wasserman, W., Kutner, M. H. (1983). Applied Linear Regression Models. Richard D. Irwin, Inc.

Seber, G. A. F., Lee, A. J. (2003). Linear Regression Analysis. Wiley Series in Probability and Statistics.

https://newonlinecourses.science.psu.edu/stat501/node/2/

https://online-learning.harvard.edu/course/data-science-linear-regression