

A PROJECT REPORT ON

**Wine Quality analysis Using Regression Method**

by

# Miss.Priya R.Umate

**B.E(CE)**

Batch: 2021-5127

Center: pune(Chinchwad)

## Under the Guidance of,

Mr. Mathivanan Balakrishnan

Technical Trainer

### EduBridge

(School of Coding)

#### **Introduction**

**What is Regression Analysis?**

Regression analysis is a set of statistical methods used for the estimation of relationships between a dependent variable and one or more [independent variables](https://corporatefinanceinstitute.com/resources/knowledge/modeling/independent-variable/). It can be utilized to assess the strength of the relationship between variables and for modeling the future relationship between them.Regression analysis includes several variations, such as linear, multiple linear, and nonlinear.

Regression analysis is primarily used for two conceptually distinct purposes. First, regression analysis is widely used for [prediction](https://en.wikipedia.org/wiki/Prediction) and [forecasting](https://en.wikipedia.org/wiki/Forecasting), where its use has substantial overlap with the field of [machine learning](https://en.wikipedia.org/wiki/Machine_learning). Second, in some situations regression analysis can be used to infer [causal relationships](https://en.wikipedia.org/wiki/Causality) between the independent and dependent variables. Importantly, regressions by themselves only reveal relationships between a dependent variable and a collection of independent variables in a fixed dataset. To use regressions for prediction or to infer causal relationships, respectively, a researcher must carefully justify why existing relationships have predictive power for a new context or why a relationship between two variables has a causal interpretation. The latter is especially important when researchers hope to estimate causal relationships using [observational data](https://en.wikipedia.org/wiki/Observational_study).

## Terminologies related to regression analysis

1. Outliers

Suppose there is an observation in the dataset which is having a very high or very low value as compared to the other observations in the data, i.e. it does not belong to the population, such an observation is called an outlier. In simple words, it is extreme value. An outlier is a problem because many times it hampers the results we get.

2.Multicollinearity

When the independent variables are highly correlated to each other then the variables are said to be multicollinear. Many types of regression techniques assumes multicollinearity should not be present in the dataset. It is because it causes problems in ranking variables based on its importance. Or it makes job difficult in selecting the most important independent variable (factor).

3. Heteroscedasticity

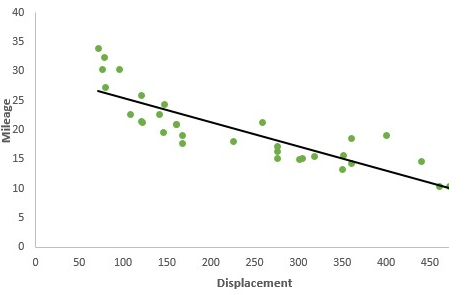
When dependent variable's variability is not equal across values of an independent variable, it is called heteroscedasticity. Example -As one's income increases, the variability of food consumption will increase. A poorer person will spend a rather constant amount by always eating inexpensive food; a wealthier person may occasionally buy inexpensive food and at other times eat expensive meals. Those with higher incomes display a greater variability of food consumption.

## Types of Regression

### Every regression technique has some assumptions attached to it which we need to meet before running analysis. These techniques differ in terms of type of dependent and independent variables and distribution.

### 1. Linear Regression

It is the simplest form of regression. It is a technique in which the dependent variable is continuous in nature. The relationship between the dependent variable and independent variables is assumed to be linear in nature.We can observe that the given plot represents a somehow linear relationship between the mileage and displacement of cars. The green points are the actual observations while the black line fitted is the line of regression



### 2. Polynomial Regression

It is a technique to fit a nonlinear equation by taking polynomial functions of independent variable. In the figure given below, you can see the red curve fits the data better than the green curve. Hence in the situations where the relation between the dependent and independent variable seems to be non-linear we can deploy Polynomial Regression Models.

### 3. Logistic Regression

In logistic regression, the dependent variable is binary in nature (having two categories). Independent variables can be continuous or binary. In multinomial logistic regression, you can have more than two categories in your dependent variable.

### 4. Lasso Regression

Lasso stands for Least Absolute Shrinkage and Selection Operator. It makes use of L1 regularization technique in the objective function. Thus the objective function in LASSO regression becomes For the estimates we don't have any specific mathematical formula but we can obtain the estimates using some statistical software.

### 5. Ordinal Regression

Ordinal Regression is used to predict ranked values. In simple words, this type of regression is suitable when dependent variable is ordinal in nature. Example of ordinal variables - Survey responses (1 to 6 scale), patient reaction to drug dose (none, mild, severe).

**Five Applications of Regression Analysis**

The regression analysis method of forecasting generally involves five basic applications. There are more, but businesses that believe in the advantages of regression analysis generally use the following:

1. **Predictive analytics:** This application, which involves forecasting future opportunities and risks, is the most widely used application of regression analysis in business
2. **Operation efficiency: Companies use this application to optimize the business process. For example, a factory manager might use regression analysis to see what the impact of oven temperature will be on loaves of bread baked in those ovens, such as how long their shelf life might be.**
3. **Supporting decisions: Many companies and their top managers today are using regression analysis (and other kinds of data analytics) to make an informed business decision and eliminate guesswork and gut intuition.**
4. **Correcting errors: Even the most informed and careful managers do make mistakes in judgment. Regression analysis helps managers, and businesses in general, recognize and correct errors.**
5. **New Insights: Looking at the data can provide new and fresh insights. Many businesses gather lots of data about their customers**.

**Python**:**Introduction**

Python is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). Python's design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with its notable use of [significant indentation](https://en.wikipedia.org/wiki/Off-side_rule). Its [language constructs](https://en.wikipedia.org/wiki/Language_construct) as well as its [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help [programmers](https://en.wikipedia.org/wiki/Programmers) write clear, logical code for small and large-scale projects.

Pythonis [dynamically-typed](https://en.wikipedia.org/wiki/Dynamic_programming_language) and [garbage-collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)).

[Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) began working on Python in the late 1980s, as a successor to the [ABC programming language](https://en.wikipedia.org/wiki/ABC_(programming_language)), and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as [list comprehensions](https://en.wikipedia.org/wiki/List_comprehension) and a garbage collection system using [reference counting](https://en.wikipedia.org/wiki/Reference_counting) and was discontinued with version 2.7.18 in 2020. Python 3.0 was released in 2008 and was a major revision of the language that is not completely [backward-compatible](https://en.wikipedia.org/wiki/Backward_compatibility) and much Python 2 code does not run unmodified on Python 3.

###### **Data Processing**

Itis the process of performing data operations to collect, convert and indentify data to generate useful informations. In this lab session we are going to handle noisy, inconsistent, intentional data using python library which can handle various type of encoding such as comma-separated values (CSV), eXtensible Markup Language (XML), Hyper Text Markup Langauge (HTML), Structured Query Language (SQL), JavaScript Object Notation (JSON) etc. For this encoding process different types of modules should be imported. Python's has different data pre-processing library where Pandas is one of them langauge package used for data processing.

**Python Data Preprocessing methods.**

* Importing the libraries
* Importing the dataset
* Handling the Missing data
* Split the dataset into training and testing datasets.
* Feature Scaling

**Important Libraries need to be imported for projects:**

**Pandas** : Pandas is the most popular and favourite data science library written in the Python Programming Language for data manipulation and analysis also pandas provides fast analysis as well as data cleaning and preparation. The best is that Pandas can work with variety of data such as: Excel Sheet, csv file, SQL file or even a webpage. Some of the features using Python Pandas library are listed below: 1. Pandas Data Frame makes manipulating data easy, we are able to select, replace columns and rows and even reshape our data. 2. Pandas allows to perform conditional selecting conditional selection using bracket notation []. 3. Pandas allow index of a Data Frame 4. Pandas allow setting the index of a Data Frame. 5. Pandas will automatic fill in those missing points with a NaN or Null value, also we can replace our missing values using .fillna() method. 6. Pandas has a .groupby () method which is used to group together rows based off a column so that we can perform aggregate functions (sum, mean. median, standard deviation) 7. Pandas allows to get the number of times occurs in a DataFrame. 8. Pandas has .describe() method is used to get an overview of DataFrame. 9.Similarly , we can concatenate, merge and join multiple DataFrame.

Import Pandas as pd

**Numpy** : Numpy is a package used for scientific calculating and perform various operations. Numpy Array is a multidimensional array whose indexed is similar to Sequences which is start with zero that is used to store values of same datatype. Numpy in the python uses less memory to store data as compared with python list. Numpy provides multiple functions they are nonzero and count\_non-zero for finding the element. Using Numpy Library we can speed up our workflow and interface with other package in python ecosystem.

Import numpy as np

**Matplotlib** : Malplotlib which was introduced by John Hunter is a multiplatform data visualization library built on Numpy arrays and designed to work with broader SciPy stack. Similarly, matplotlib.pyplot is a collection of command style functions that make matplotlib work like MATLAB. Pyplot is mainly intented for interative plots and simple cases of programmatic plot generations.

import matplotlib.pyplot as plt

**ScikitLearn** : Scikit-learn (formerly scikits.learn and also known as sklearn) is a [free software](https://en.wikipedia.org/wiki/Free_software) [machine learning](https://en.wikipedia.org/wiki/Machine_learning) [library](https://en.wikipedia.org/wiki/Library_(computing)) for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) [programming language](https://en.wikipedia.org/wiki/Programming_language). It features various [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and [clustering](https://en.wikipedia.org/wiki/Cluster_analysis) algorithms including [support vector machines](https://en.wikipedia.org/wiki/Support_vector_machine), [random forests](https://en.wikipedia.org/wiki/Random_forests), [gradient boosting](https://en.wikipedia.org/wiki/Gradient_boosting), [*k*-means](https://en.wikipedia.org/wiki/K-means_clustering) and [DBSCAN](https://en.wikipedia.org/wiki/DBSCAN), and is designed to interoperate with the Python numerical and scientific libraries [NumPy](https://en.wikipedia.org/wiki/NumPy) and [SciPy](https://en.wikipedia.org/wiki/SciPy).

**Seaborn**: One of the best but also more challenging ways to get your insights across is to visualize them: that way, you can more easily identify patterns, grasp difficult concepts or draw the attention to key elements. When you’re using Python for data science, you’ll most probably will have already used [Matplotlib](https://matplotlib.org/), a 2D plotting library that allows you to create publication-quality figures. Another complimentary package that is based on this data visualization library is [Seaborn](http://seaborn.pydata.org/), which provides a high-level interface to draw statistical graphics.

Import seaborn as sns

**Process involved:**

* Importing the dataset
* To know whether any cell value is empty or not
* If empty, Filling missing value
* Preparing our data- defining attributes and labels
* The next step is to split this data into training and test sets
* Training the algorithm with one of the ML concept
* Import regression using Scikit-Learn library
* Making prediction based on attributes(inputs)
* Evaluating the model

**Languages** **Used**: Python

**Libraries Used:** Pandas, Numpy, Matplotlib, ScikitLearn, Seaborn.

**IDE Used:** Jupyter Notebook

**coding and explaination**

1. **Linear regression**

def basic\_linear(wine\_set):

# recode quality into 2 groups: 0:{3,4,5,6}, 1:{7,8,9}

recode = {3: 0, 4: 0, 5:0, 6:0, 7:1, 8:1, 9:1}

wine\_set['quality\_c'] = wine\_set['quality'].map(recode)

scat0 = seaborn.regplot(x="volatile\_acidity", y="quality\_c", fit\_reg=True, data=wine\_set)

plt.xlabel("Amount of volatile acidity in wine")

plt.ylabel("Quality level of wine (0-10 scale)")

plt.title("Association between the amount of volatile acidity in wine and the quality of wine")

plt.show()

# centering the explanatory variable by subrtacting the mean

f\_acidity\_mean = wine\_set["volatile\_acidity"].mean()

print("mean of the volatile acidity variable = ", f\_acidity\_mean)

wine\_set["volatile\_acidity"] = wine\_set["volatile\_acidity"] - f\_acidity\_mean

print("mean of the volatile acidity variable after normalization = ", wine\_set["volatile\_acidity"].mean())

print ("\nOLS regression model for the association between the amount of volatile acidity in wine and the quality of wine:")

model1 = smf.ols(formula="quality\_c ~ volatile\_acidity", data=wine\_set)

results1 = model1.fit()

print(results1.summary())

call(basic\_linear)

**Explaination**:

1. In Linear Regression algorithm first we importing the CSV file which is pd.read\_csv(“path/filename.csv”)
2. After that training set ,test set splits into 2 groups and splits the plots.
3. After that centering the explanatory variable by subrtacting the mean and linear model with training sets .
4. Then call the method that is predicted model.
5. Comparison- as compare to other regression model linear regression is easy to understand and less time consume.

### 2. Logistic Regression

def log\_regression(wine\_set):

# local variable to identify if the wine\_set red or white

w = wine\_set

# recode quality (response variable) into 2 groups: 0:{3,4,5}, 1:{6,7,8,9}

recode = {3: 0, 4: 0, 5: 0, 6: 1, 7: 1, 8: 1, 9: 1}

wine\_set['quality\_c'] = wine\_set['quality'].map(recode)

#split into training and testing sets

predictors = wine\_set[["sulphates", 'alcohol']]

targets = wine\_set.quality\_c

pred\_train, pred\_test, tar\_train, tar\_test = train\_test\_split(predictors, targets, test\_size=.4)

classifier = LogisticRegression()

classifier = classifier.fit(pred\_train, tar\_train)

predictions = classifier.predict(pred\_test)

print('Confusion Matrix:\n',sklearn.metrics.confusion\_matrix(tar\_test, predictions))

print('Accuracy:',sklearn.metrics.accuracy\_score(tar\_test, predictions))

print ('Score:', classifier.score(pred\_test, tar\_test))

print ('RMSE:', mean\_squared\_error(predictions, tar\_test) \*\* 0.5)

print('----------------Logistic Regression------------------------')

call(log\_regression)

**Explaination**:

1 . in logistic regression algorithm first importing csv file .

1. then local variable to identify if the wine\_set red or white assign w= wine\_set
2. after that split into training and testing sets test set split and plot the points
3. and after that predict the model wine set which is sulphate and alcohol, and print the logistic regression and also call it.
4. Comparison: logistic regression algorithm is used for solving classification problem.
5. In logistic Regression, we predict the values of categorical variables.

### 3. Lasso Regression:

def lasso\_regr(wine\_set):

pred = wine\_set[["density", 'alcohol', 'sulphates', 'pH', 'volatile\_acidity', 'chlorides', 'fixed\_acidity',

'citric\_acid', 'residual\_sugar', 'free\_sulfur\_dioxide', 'total\_sulfur\_dioxide']]

predictors = pred.copy()

targets = wine\_set.quality

# standardize predictors to have mean=0 and sd=1

predictors = pd.DataFrame(preprocessing.scale(predictors))

predictors.columns = pred.columns

# print(predictors.head())

# split into training and testing sets

pred\_train, pred\_test, tar\_train, tar\_test = train\_test\_split(predictors, targets, test\_size=.3, random\_state=123)

# specify the lasso regression model

model = LassoLarsCV(cv=10, precompute=False).fit(pred\_train, tar\_train)

print('Predictors and their regression coefficients:')

d = dict(zip(predictors.columns, model.coef\_))

for k in d:

print(k, ':', d[k])

# plot coefficient progression

m\_log\_alphas = -np.log10(model.alphas\_)

# ax = plt.gca()

plt.plot(m\_log\_alphas, model.coef\_path\_.T)

print('\nAlpha:', model.alpha\_)

plt.axvline(-np.log10(model.alpha\_), linestyle="dashed", color='k', label='alpha CV')

plt.ylabel("Regression coefficients")

plt.xlabel("-log(alpha)")

plt.title('Regression coefficients progression for Lasso paths')

plt.show()

# plot mean squared error for each fold

m\_log\_alphascv = -np.log10(model.cv\_alphas\_)

plt.plot(m\_log\_alphascv, model.mse\_path\_, ':')

plt.plot(m\_log\_alphascv, model.mse\_path\_.mean(axis=-1), 'k', label='Average across the folds', linewidth=2)

plt.legend()

plt.xlabel('-log(alpha)')

plt.ylabel('Mean squared error')

plt.title('Mean squared error on each fold')

plt.show()

# Mean squared error from training and test data

train\_error = mean\_squared\_error(tar\_train, model.predict(pred\_train))

test\_error = mean\_squared\_error(tar\_test, model.predict(pred\_test))

print('\nMean squared error for training data:', train\_error)

print('Mean squared error for test data:', test\_error)

rsquared\_train = model.score(pred\_train, tar\_train)

rsquared\_test = model.score(pred\_test, tar\_test)

print('\nR-square for training data:', rsquared\_train)

print('R-square for test data:', rsquared\_test)

#

print('----------------Lasso Regression------------------------')

call(lasso\_regr)

**Explaination**:--

1. First importing csv file which is pd.read\_csv(“path/filename.csv”)
2. After that we have to predict by using standardize predictors to have mean=0 and sd=1
3. Then after that split into training and testing sets
4. Then we have to specify the lasso regression model and then print the 'Predictors and their regression coefficients
5. After that plot the coefficient of progression
6. After that plot the mean squared error for each fields
7. Next step is to find Mean squared error from training and test data and predict the model
8. Comparison: Lasso is a modification of linear regression, where the model is penalized for the sum of absolute values of the weights. Thus, the absolute values of weight will be (in general) reduced, and many will tend to be zeros. During training, the objective function become

**4. knn regression**

def knn(wine\_set):

# recode quality (response variable) into 2 groups: 0:{3,4,5}, 1:{6,7,8,9}

recode = {3: 0, 4: 0, 5: 0, 6: 1, 7: 1, 8: 1, 9: 1}

wine\_set['quality\_c'] = wine\_set['quality'].map(recode)

# split into training and testing sets

predictors = wine\_set[["residual\_sugar", 'alcohol']]

targets = wine\_set.quality\_c

pred\_train, pred\_test, tar\_train, tar\_test = train\_test\_split(predictors, targets, test\_size=.4)

# build model on training data

classifier = KNeighborsClassifier()

classifier = classifier.fit(pred\_train, tar\_train)

predictions = classifier.predict(pred\_test)

# print the confusion matrix and accuracy of the model

print('Confusion Matrix:\n',sklearn.metrics.confusion\_matrix(tar\_test, predictions))

print('Accuracy:',sklearn.metrics.accuracy\_score(tar\_test, predictions))

print ('Score:', classifier.score(pred\_test, tar\_test))

print ('RMSE:', mean\_squared\_error(predictions, tar\_test) \*\* 0.5)

print('----------------KNN------------------------')

call(knn)

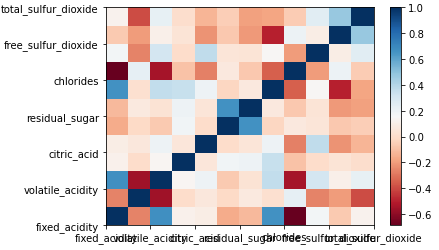
**Explaination:---**

**1** KNN regression is a **non-parametric method** that, in an intuitive manner, approximates the association between independent variables and the continuous outcome by averaging the observations in the same neighbourhood.

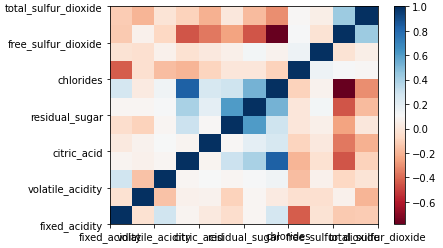
1. Importing csv file with(.csv )extension.
2. After that split into training and testing sets
3. Then build model on training data
4. Then print the confusion matrix and accuracy of the model
5. Comparison:- KNN is a non-parametric model, where LR is a parametric model.
6. KNN is comparatively slower than Logistic Regression.
7. KNN supports non-linear solutions where LR supports only linear solutions.
8. LR can derive confidence level (about its prediction), whereas KNN can

only output the labels.

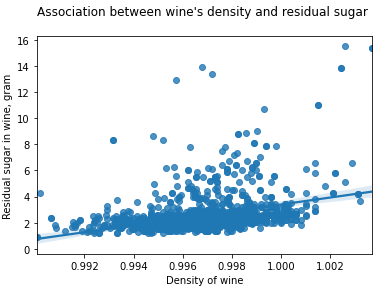
**Screenshots**

****

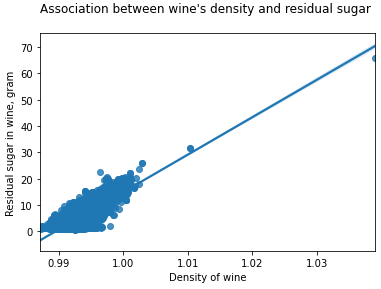
**Src: 1: covariance matric of Red wine quality dataset**

****

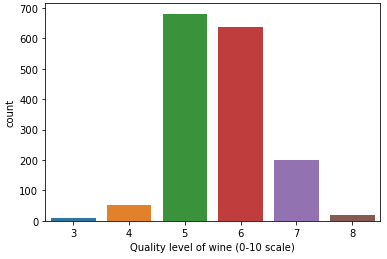
**Src: 2: covariance matric of white wine quality dataset**

****

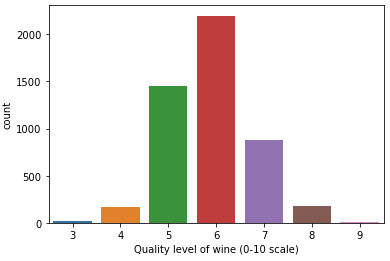
**Src:density of red wine quality**

****

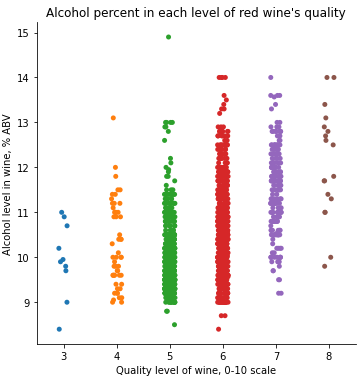
**Src: density of white wine quality**

****

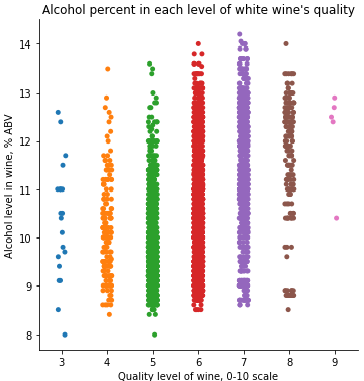
**Src: Visualization with countplots of Red wine quality**

****

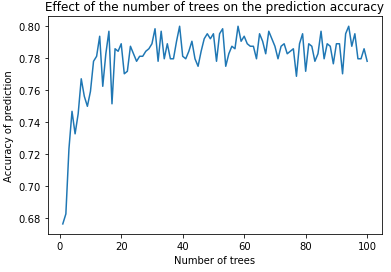
**Src: Visualization with countplots of White wine quality**

****

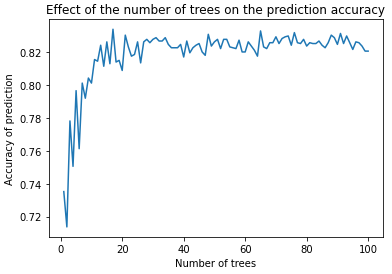
**Src: Visualization with factorplots of red wine quality**

****

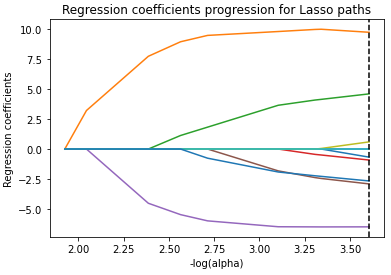
**Src: Visualization with factorplots of White wine quality**

****

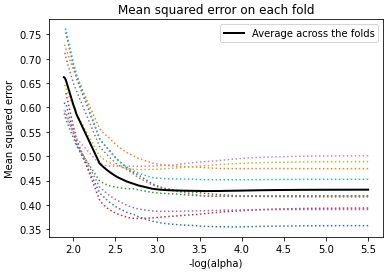
**Src: random forest plots for Red wine quality**

****

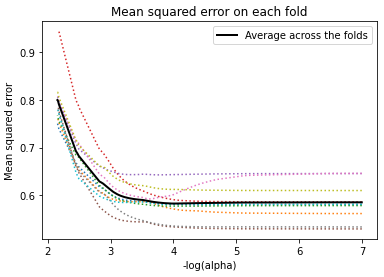
**Src: random forest plots for White wine quality**

****

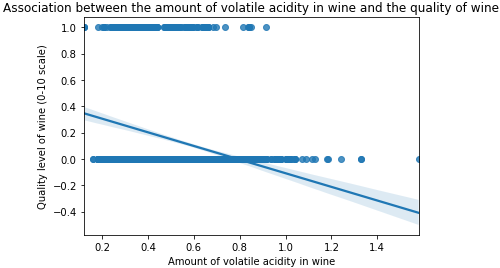
**Src: lasso regression**

****

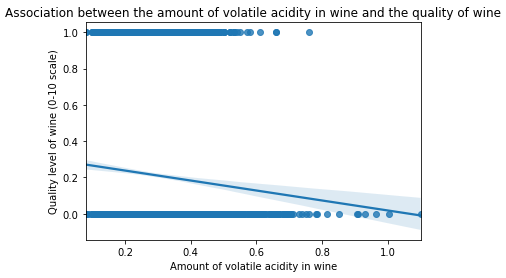
**Src: lasso regression 2**

****

**Src: lasso regression 3**

****

**Src: linear regression for Red wine quality**

****

**Src: linear regression for White wine quality**

# THANK YOU