**Wine Quality Predictor**

Submitted for the Summer Internship

On

**AI/ML**

(from $15^{th}$ May to $15^{th}$ July 2023)

**Organized by**

**Skillish**

By Akshita Pal

**ECE-AI (06601182021)**

**INDIRA GANDHI DELHI TECHNICAL UNIVERSITY FOR WOMEN**

(Established by Govt. of Delhi vide Act 09 of 2012)

Kashmere Gate, Delhi - 110006

## INDEX

**CERTIFICATE**


Skillish

We Enlighten your Skills

# CERTIFICATE

OF EXCELLENCE

PRESENTED TO :

Akshita Pal

has completed his/her internship of 2 months . The candidate has gone through several project modules during his/her internship and successfully completed all the tasks given to him/her . His/her performance has been marked "Good " during the internship period .

Pushpender Kaushik
Manager

Prateek Raj
Founder

## UNDERTAKING REGARDING ANTI-PLAGIARISM

I hereby declare that the material/content presented in the report is free from plagiarism and is properly cited and written in my own words. In case, plagiarism is detected at any stage, I shall be responsible for it.

**Akshita Pal**
**(06601182021)**

# ACKNOWLEDGEMENT

I would like to thank Ms. Ritika, counselor at Skillish, for reaching out to me with this wonderful course in Machine Learning and guiding me through all the steps from registration to project submission. I would also like to thank Skillish for creating this really easy-to-understand course, which helped me to clarify my knowledge regarding Machine Learning.
I would also like to thank my college administration, faculty, and peers for motivating me throughout the journey.

**Akshita Pal**
**(06601182021)**

# DECLARATION

I, Akshita Pal, solemnly declare that the project report, coding, and internship are based on my own words carried out during my study under the supervision of Skillish . I assert the statements made and conclusions drawn are an outcome of my research work. I further certify that:

I. The work contained in the report is original and has been done by me under the supervision of my supervisor.

II. The work has not been submitted to any other Institute for any other degree/diploma/certificate in this university or any other University of India or abroad.

III. We have followed the guidelines provided by the university in writing the report.

IV. Whenever we have used materials (text, data, theoretical analysis/equations, code/programs, etc.) from other sources, we have given due credit to them in the report and have given their details in the references.

**Akshita Pal**
**(06601182021)**

# ABSTRACT/SUMMARY


## Skillish - Learning Machine Learning

I was always interested in exploring the field of Data Science and AI. The first step towards it would include learning Machine Learning. The course guided me through all the basic skills and prerequisites required to build an efficient program. To begin with, I learned about the required in-built modules of Python that I would require followed by all the essential information regarding ML. This course was a really enriching experience.

## Wine Quality Predictor

Wine is the finest drink of all time and requires very minute detailing for it to not become hazardous. Many companies don't keep track of all the materials and their usage in their wine products. This program is made to help people check if the wine they are consuming is hazardous or not, simply by inputting the details of the ingredients, given on the package body.

# CHAPTER - 1 SKILLISH - LEARNING MACHINE LEARNING

Skillish is an India-based company which provide students with various courses in the field of Computer Science across India at a nominal price. The organization carefully curates and develops its courses meticulously, keeping the end-user in mind the whole time.

## 1.1 Getting started with ML

The course very smoothly started with the introduction to Machine Learning, stating its importance in different fields. It went on with how the common day-to-day apps like Spotify, Amazon, Flipkart, etc. use Machine Learning so that their customers' engagement doesn't go down. It basically defines Machine Learning as the ability of the program to learn the trends and predict the outcome of the further trend.
For example,
Amazon's most common use of Machine Learning can be seen under the link of "People buy this together". The app learns what maximum people like to buy together and then suggests the same to all its customers, to increase sales.

## 1.2 Python for Data Analysis

The most common languages used for Machine Learning are R and Python. I opted for the Python language.
The first step for Machine Learning is Data Analysis. In Python, Data Analysis is done by using the inbuilt modules NumPy and Pandas. NumPy and Pandas sort the data into different groups in the form of arrays so that it is easier for the program to understand it. Pandas also enable the user to open a CSV file that includes the dataset required for the Data Analysis.

## 1.3 Python for Data Visualization

Python includes another inbuilt module PyPlot whose submodule is named Matplotlib, which is used for Data Visualization in Machine Learning.
Data Visualization in ML refers to forming graphs against different factors that will affect the end results. This helps the programmer to build insight into the data and the trend.
Matplotlib is used along with NumPy to provide an environment that is an effective open-source alternative to Matlab.
Matplotlib toolkits that I learned in the course:

1. Basemap: map plotting with various map projections, coastlines, and political boundaries
2. Cartopy: a mapping library featuring object-oriented map projection definitions, and arbitrary point, line, polygon, and image transformation capabilities.
3. Excel tools: utilities for exchanging data with Microsoft Excel
4. Mplot3d: 3-D plots

5. Seaborn: provides an API on top of Matplotlib that offers sane choices for plot style and color defaults, defines simple high-level functions for common statistical plot types, and integrates with the functionality provided by Pandas

## 1.4 Linear Regression

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.

Steps for Linear Regression:

1. Importing Libraries
2. Importing Datasets
3. Dividing dataset into Training and Testing Data
4. Fitting Model to training dataset
5. Prediction of the test set results

## 1.5 K Nearest Neighbors

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

As a dataset grows, KNN becomes increasingly inefficient, compromising overall model performance. It is commonly used for simple recommendation systems, pattern recognition, data mining, financial market predictions, intrusion detection, and more.

The k value in the k-NN algorithm defines how many neighbors will be checked to determine the classification of a specific query point. For example, if k=1, the instance will be assigned to the same class as its single nearest neighbor. The default value of k is 5.

## 1.6 Comparing Different Classification Models

There are two types of models:

1. Classification
2. Regression

Classification:

Classification problems occur when the output is a category or a group.

Under Classification Models, there are different methods. Some are:

1. K Nearest Neighbor
2. Decision Tree
3. Naive Bayes
4. Support Vector Machine
5. Random Forest

Regression:

Regression problems occur when the output is a real value

Under Regression Models, there are different methods. Some are:

1. Linear Regression
2. Logistic Regression
3. Polynomial Regression
4. Ridge Regression
5. Lasso Regression
6. Elastic Net Regression

## 1.7 K Means Clustering

K-means clustering aims to partition data into k clusters in a way that data points in the same cluster are similar and data points in the different clusters are farther apart.

K-means clustering tries to minimize distances within a cluster and maximize the distance between different clusters.

Pros and Cons

Pros:

1. Easy to interpret
2. Relatively fast
3. Scalable for large data sets
4. Able to choose the positions of initial centroids in a smart way that speeds up the convergence
5. Guarantees convergence

Cons:

1. The number of clusters must be pre-determined. The K-means algorithm is not able to guess how many clusters exist in the data. Determining the number of clusters may well be a challenging task.

2. Can only draw linear boundaries. If there is a non-linear structure separating groups in the data, k-means will not be a good choice.
3. Slows down as the number of samples increases because, at each step, the k-means algorithm accesses all data points and calculates distances. An alternative way is to use a subset of data points to update the location of centroids (i.e. sklearn.cluster.MiniBatchKMeans)
4. Sensitive to outliers

## 1.8 Unsupervised Learning

Unsupervised Learning:

Training of an ML algorithm using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. The data given to unsupervised algorithms are not labeled, which means only the input variables are given with no corresponding output variables.

Types of Unsupervised Learning:

1. Clustering:
   Clustering problem is where you want to discover the inherent grouping in the data, such as grouping customers by purchasing behaviors.
2. Association:
   An association rule learning is where you want to discover rules that describe large portions of your data, such as ‘People that buy X tend to buy Y’.

Types of Clustering Methods:

1. K Means Clustering
2. Mean Shift Clustering
3. EM Clustering
4. Agglomerating Hierarchical Clustering

Types of Association Methods:

1. Apriori Algorithm
2. Eclat Algorithm
3. F P Growth Algorithm

# CHAPTER - 2 WINE QUALITY PREDICTOR

Wine is an alcoholic drink that is made up of fermented grapes.
Link to my project:
https://github.com/Akshita41/Wine-Quality-Predictor-

## 2.1 Dataset

Dataset is found from Kaggle.
Link: https://www.kaggle.com/datasets/rajyellow46/wine-quality?resource=download
To classify the quality of wine, we need to understand about different chemicals:

- volatile acidity: Volatile acidity is the gaseous acids present in wine.
- fixed acidity: Primarily fixed acids found in wine are tartaric, succinic, citric, and malic
- residual sugar: Amount of sugar left after fermentation.
- citric acid: It is a weak organic acid, found in citrus fruits naturally.
- chlorides: Amount of salt present in wine.
- free sulfur dioxide: So2 is used for the prevention of wine by oxidation and microbial spoilage.
- total sulfur dioxide
- pH: In wine, pH is used for checking acidity
- density
- sulphates: Added sulphates preserve freshness and protect wine from oxidation, and bacteria.
- alcohol: Percent present in wine.

## 2.2 Points to be Noted while detecting the Quality

While making the program, a few of the important points to be noted and a few important graphs to be made are:

1. Check for the null values in the dataset, and either remove them or fill them with mean values
2. Graph of each chemical vs quality graph
3. According to the dataset, the quality of wine with 7 or more is termed good wine.
4. Construct a heatmap to understand the correlation between each component of the wine.
5. Check the accuracy score of the model. Accuracy of 90% and above is useful.

## 2.3 Code and working of my program

Copy of Wine testing ML.ipynb

File Edit View Insert Runtime Tools Help

Comment Share

+ Code + Text

RAM Disk Editing

Importing the libraries

[1]
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

Data Collection

[2]
```
# loading the dataset to pandas dataframe
wine_dataset = pd.read_csv('/content/drive/MyDrive/winequalityN.csv')
```

```
# Number of rows and columns in the dataset
wine_dataset.shape
```

```
(6497, 13)
```

Copy of Wine testing ML.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

[4]
```
wine_dataset.head()
```

| | type | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcohol | quality |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | white | 7.0 | 0.27 | 0.36 | 20.7 | 0.045 | 45.0 | 170.0 | 1.0010 | 3.00 | 0.45 | 8.8 | 6 |
| 1 | white | 6.3 | 0.30 | 0.34 | 1.6 | 0.049 | 14.0 | 132.0 | 0.9940 | 3.30 | 0.49 | 9.5 | 6 |
| 2 | white | 8.1 | 0.28 | 0.40 | 6.9 | 0.050 | 30.0 | 97.0 | 0.9951 | 3.26 | 0.44 | 10.1 | 6 |
| 3 | white | 7.2 | 0.23 | 0.32 | 8.5 | 0.058 | 47.0 | 186.0 | 0.9956 | 3.19 | 0.40 | 9.9 | 6 |
| 4 | white | 7.2 | 0.23 | 0.32 | 8.5 | 0.058 | 47.0 | 186.0 | 0.9956 | 3.19 | 0.40 | 9.9 | 6 |

```
# checking for null values
wine_dataset.isnull().sum()
```

```
type                0
fixed acidity      10
volatile acidity    8
citric acid         3
residual sugar      2
chlorides           2
```

Copy of Wine testing ML.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
[6] # filling the null values with the meaqn of the column
wine_dataset.update(wine_dataset.fillna(wine_dataset.mean()))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is
```

Data Analysis and Visualization

```
# desciption of the data set
wine_dataset.describe()
```

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcohol | quality |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 6497.000000 | 6497.000000 | 6497.000000 | 6497.000000 | 6497.000000 | 6497.000000 | 6497.000000 | 6497.000000 | 6497.000000 | 6497.000000 | 6497.000000 | 6497.000000 |
| **mean** | 7.216579 | 0.339691 | 0.318722 | 5.444326 | 0.056042 | 30.525319 | 115.744574 | 0.994697 | 3.218395 | 0.531215 | 10.491801 | 5.818378 |
| **std** | 1.295751 | 0.164548 | 0.145231 | 4.757392 | 0.035031 | 17.749400 | 56.521855 | 0.002999 | 0.160637 | 0.148768 | 1.192712 | 0.873255 |
| **min** | 3.800000 | 0.080000 | 0.000000 | 0.600000 | 0.009000 | 1.000000 | 6.000000 | 0.987110 | 2.720000 | 0.220000 | 8.000000 | 3.000000 |
| **25%** | 6.400000 | 0.230000 | 0.250000 | 1.800000 | 0.038000 | 17.000000 | 77.000000 | 0.992340 | 3.110000 | 0.430000 | 9.500000 | 5.000000 |

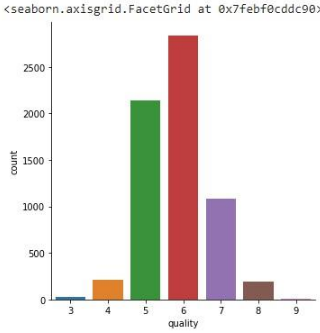
✓ 0s completed at 9:48 PM

Copy of Wine testing ML.ipynb

File Edit View Insert Runtime Tools Help

```
# number of values for each quality
sns.catplot(x = 'quality', data = wine_dataset, kind = 'count')
```

```
<seaborn.axisgrid.FacetGrid at 0x7febf0cddc90>
```

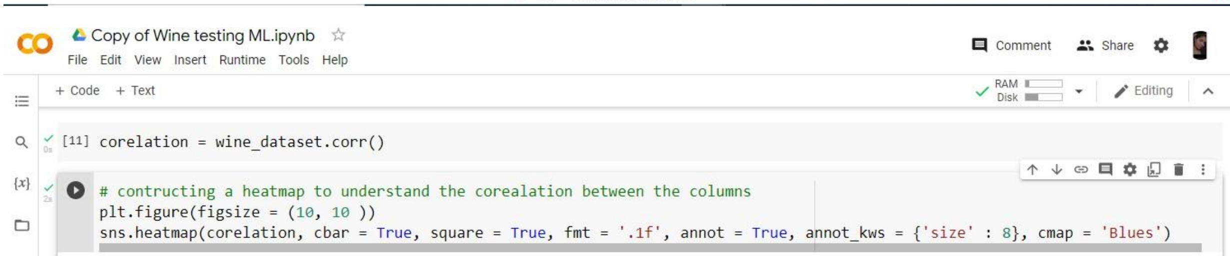



```
[ ] # volatile acidity vs quality graph
```

✓ 1s completed at 9:49 PM


Copy of Wine testing ML.ipynb
File Edit View Insert Runtime Tools Help
Comment Share
+ Code + Text
RAM
Disk
Editing
# volatile acidity vs quality graph
plot = plt.figure(figsize=(6,5))
sns.barplot(x = 'quality', y = 'volatile acidity', data = wine_dataset )
<matplotlib.axes._subplots.AxesSubplot at 0x7febee39aa90>
volatile acidity
quality
0s completed at 9:49 PM


Copy of Wine testing ML.ipynb
File Edit View Insert Runtime Tools Help
Comment Share
+ Code + Text
RAM
Disk
Editing
# citric acid vs quality graph
plot = plt.figure(figsize=(6,5))
sns.barplot(x = 'quality', y = 'citric acid', data = wine_dataset )
<matplotlib.axes._subplots.AxesSubplot at 0x7febee336fd0>
citric acid
quality
1s completed at 9:50 PM


Copy of Wine testing ML.ipynb
File Edit View Insert Runtime Tools Help
Comment Share
+ Code + Text
RAM
Disk
Editing
[11] corelation = wine_dataset.corr()
# contructing a heatmap to understand the corealation between the columns
plt.figure(figsize = (10, 10 ))
sns.heatmap(corelation, cbar = True, square = True, fmt = '.1f', annot = True, annot_kws = {'size' : 8}, cmap = 'Blues')


Data Preprocessing

```
#separation of data
x = wine_dataset.drop(['quality', 'type'], axis = 1)
print(x)
```

```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0               7.0             0.270         0.36            20.7      0.045
1               6.3             0.300         0.34             1.6      0.049
2               8.1             0.280         0.40             6.9      0.050
3               7.2             0.230         0.32             8.5      0.058
4               7.2             0.230         0.32             8.5      0.058
...             ...               ...          ...             ...        ...
6492            6.2             0.600         0.08             2.0      0.090
6493            5.9             0.550         0.10             2.2      0.062
6494            6.3             0.510         0.13             2.3      0.076
6495            5.9             0.645         0.12             2.0      0.075
6496            6.0             0.310         0.47             3.6      0.067

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                    45.0                 170.0  1.00100  3.00   0.450000
1                    14.0                 132.0  0.99400  3.30   0.490000
2                    30.0                  97.0  0.99510  3.26   0.440000
3                    47.0                 186.0  0.99560  3.19   0.400000
4                    47.0                 186.0  0.99560  3.19   0.400000
...                   ...                   ...      ...   ...        ...
```

```
      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                    45.0                 170.0  1.00100  3.00   0.450000
1                    14.0                 132.0  0.99400  3.30   0.490000
2                    30.0                  97.0  0.99510  3.26   0.440000
3                    47.0                 186.0  0.99560  3.19   0.400000
4                    47.0                 186.0  0.99560  3.19   0.400000
...                   ...                   ...      ...   ...        ...
6492                 32.0                  44.0  0.99490  3.45   0.580000
6493                 39.0                  51.0  0.99512  3.52   0.531215
6494                 29.0                  40.0  0.99574  3.42   0.750000
6495                 32.0                  44.0  0.99547  3.57   0.710000
6496                 18.0                  42.0  0.99549  3.39   0.660000

      alcohol
0         8.8
1         9.5
2        10.1
3         9.9
4         9.9
...       ...
6492     10.5
6493     11.2
6494     11.0
6495     10.2
6496     11.0

[6497 rows x 11 columns]
```

Copy of Wine testing ML.ipynb

```
[14] y = wine_dataset['quality'].apply(lambda y_value: 1 if y_value >= 7 else 0)
     print(y)
```

```
0       0
1       0
2       0
3       0
4       0
       ..
6492    0
6493    0
6494    0
6495    0
6496    0
Name: quality, Length: 6497, dtype: int64
```

Splitting data into train adn test data

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 2)
print(y.shape, y_train.shape, y_test.shape)
```

```
(6497,) (5197,) (1300,)
```

Copy of Wine testing ML.ipynb

Training the model

```
[16] model = RandomForestClassifier()
```

```
[17] model.fit(x_train, y_train)
```

```
RandomForestClassifier()
```

Accuracy score

```
# accuracy on test data
x_test_prediction = model.predict(x_test)
test_data_accuracy = accuracy_score(x_test_prediction, y_test)
print('The accuracy is : ', test_data_accuracy)
```

```
The accuracy is :  0.8992307692307693
```

Building a predictive system

0s completed at 9:53 PM

Copy of Wine testing ML.ipynb

Building a predictive system

```
[19] input = (6.2, 0.66, 0.48, 1.2, 0.029, 29, 75, 0.9892, 3.33, 0.39, 12.8)
     # changing the input data into numpy array
     input_data = np.asarray(input)

     #reshaping the data
     input_data_reshape = input_data.reshape(1, -1)

     prediction = model.predict(input_data_reshape)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature r
  "X does not have valid feature names, but"
```

```
if (prediction[0] == 1):
    print('The wine quality is good')
else:
    print('The wine quality is bad')
```

```
The wine quality is good
```

0s completed at 9:54 PM

# CHAPTER - 3 CONCLUSION/FUTURE SCOPE

## Conclusion

The wine quality prediction system is based on the concept of Machine Learning using different classification and regression methods. Skillish course can serve a great purpose of uplifting my coding skills and thus upscale my overall skill set. Thanks to this course I'm fairly confident while doing Machine Learning. It helped me make projects, which is one of the things that companies look for in a resume of a candidate. Overall, I’m very satisfied with the domain knowledge I gained from this course.

## Future scope

As social drinking has increased, the wine business has recently experienced exponential expansion. Industry participants now use product quality certifications to market their goods. This is a time-consuming process that also costs a lot of money because it needs to be evaluated by human experts. Additionally, the cost of wine is determined by tasters' opinions, which can vary greatly and are based on an abstract concept of wine appreciation. Physicochemical tests, which are laboratory-based and take into account elements like acidity, pH level, sugar, and other chemical qualities, are an additional crucial component in the certification of red wine and the evaluation of its quality. If the human ability to taste can be connected to the chemical characteristics of wine, the wine industry may be of interest.

Aside from this, wine consumers can also detect the quality of wine using this system, to check whether the wine is worth of money or not.

# BIBLIOGRAPHY


- https://Skillish.in/courses/machine-learning/
- https://towardsdatascience.com/red-wine-quality-prediction-using-regression-modeling-and-machine-learning-7a3e2c3e1f46
- https://www.kaggle.com/datasets/rajyellow46/wine-quality?resource=download
- https://www.hackerrank.com/
- https://www.quora.com/
- https://www.geeksforgeeks.org/
- https://stackoverflow.com/

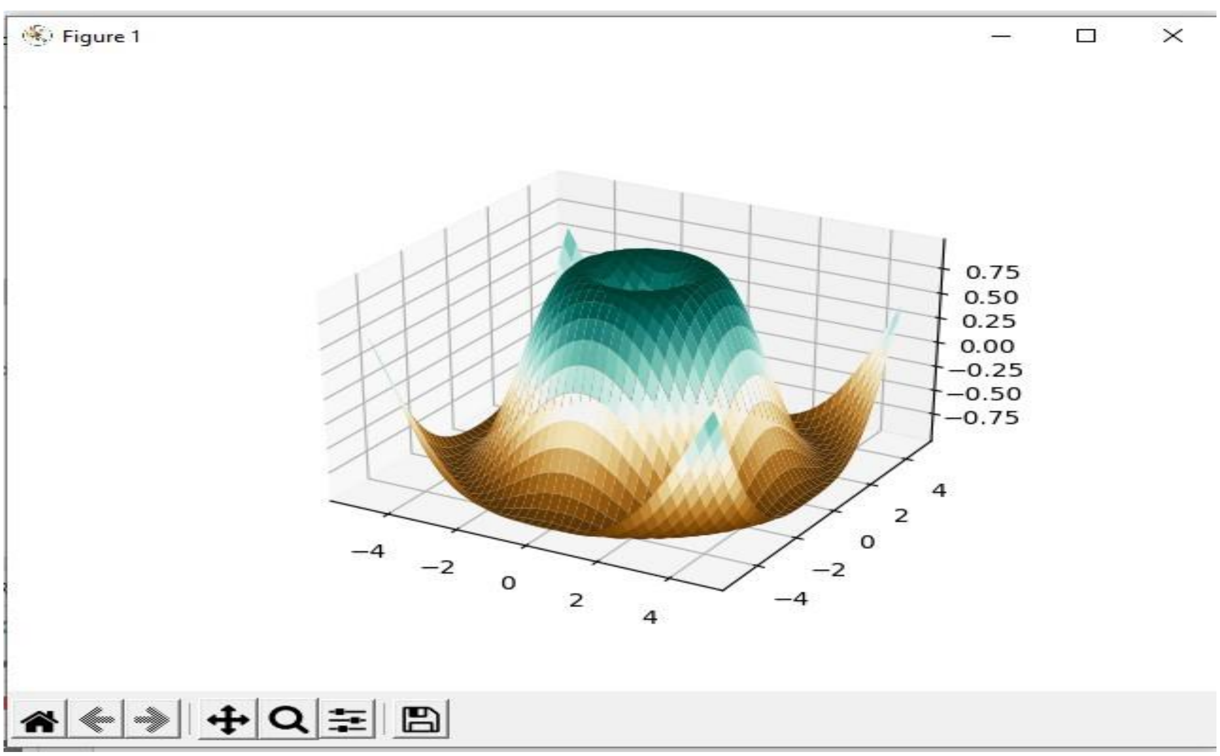
# APPENDIX

These are some codes written by me while pursuing Machine Learning Course

## 1. Matplotlib

```python
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import cm
f = plt.figure()
ax = f.gca(projection='3d')
x = np.arange(-5, 5, 0.25)
y = np.arange(-5, 5, 0.25)
x, y = np.meshgrid(x, y)
r = np.sqrt(x**2 + y**2)
z = np.sin(r)
surf = ax.plot_surface(x, y, z, rstride=1, cstride=1, cmap=cm.BrBG)
plt.show()

```

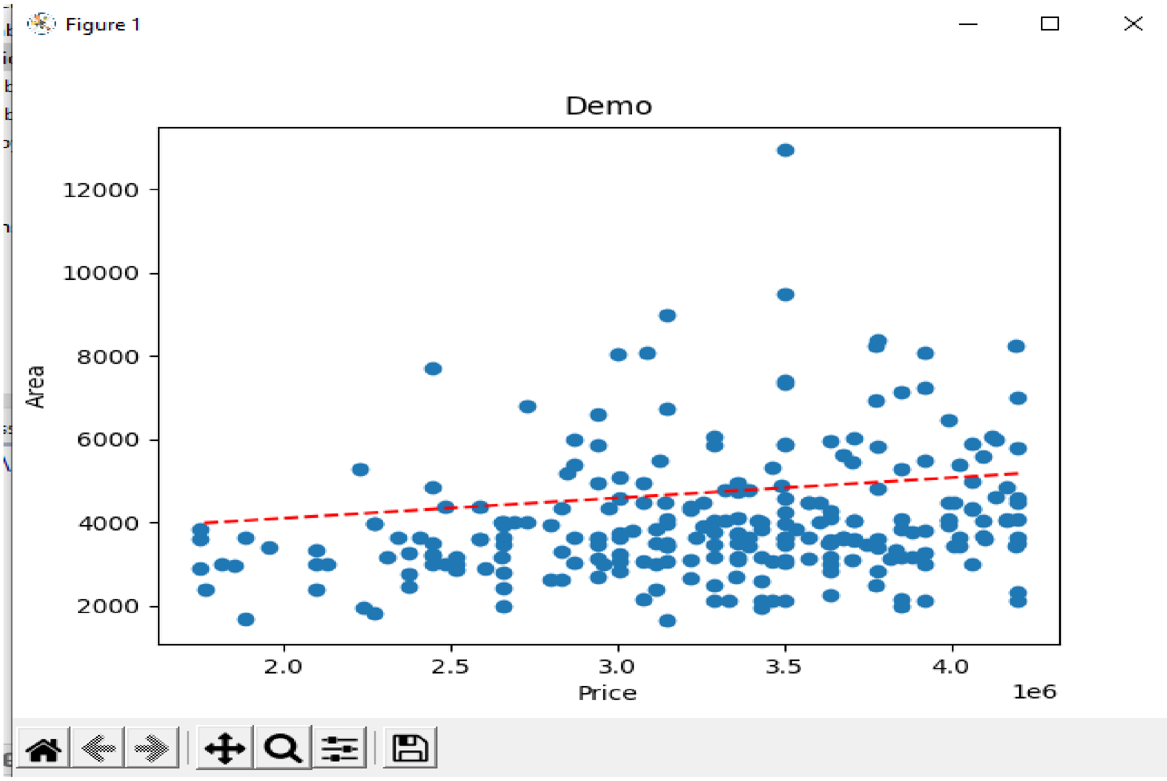
The graph made:



## 2. Linear Regression on Housing price dataset

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn import linear_model

# gathering of data
data = pd.read_csv("C:/Users/ANKITAGHOSH/Desktop/Housing.csv")
x = np.array(data["price"])
y = np.array(data["area"])
X = x.reshape(len(x), 1)
Y = y.reshape(len(y), 1)

# preparing/splitting the data
X_train = X[:-250]
X_test = X[-250:]
Y_train = Y[:-250]
Y_test = Y[-250:]

# creating a model
reg = linear_model.LinearRegression()

# training the model
reg.fit(X_train, Y_train)

# testing the model
plt.scatter(X_test, Y_test)
plt.plot(X_test, reg.predict(X_test), '--r')
plt.title("Demo")
plt.xlabel("Price")
plt.ylabel("Area")
plt.show()

```

The Graph Made




### 3. K Nearest Neighbors

```python
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier

# preparing/gathering data
iris = load_iris()

# describing the data in iris
print(iris.DESCR)
print(iris.data)
print(iris.feature_names)
print(iris.target)
print(iris.target_names)

knn = KNeighborsClassifier(n_neighbors=3)

# training model
knn.fit(iris.data, iris.target)

# prediction
print(knn.predict([[4, 6, 8, 10], ]))
a = knn.predict([[1, 1, 1, 1], ])
print(iris.target_names[a])
```

Results

```
C:\Users\ANKITAGHOSH\AppData\Local\Microsoft\WindowsApps\python3.9.exe "C:/Users/ANKITAGHOSH/PycharmProjects/MachineLearning/k nearest neghbours.py"
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
```

```
  Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
  Mathematical Statistics" (John Wiley, NY, 1950).
- Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
  (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
- Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
  Structure and Classification Rule for Recognition in Partially Exposed
  Environments".  IEEE Transactions on Pattern Analysis and Machine
  Intelligence, Vol. PAMI-2, No. 1, 67-71.
- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
  on Information Theory, May 1972, 431-433.
- See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
  conceptual clustering system finds 3 classes in the data.
- Many, many more ...

['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
['setosa' 'versicolor' 'virginica']
[2]
['setosa']

Process finished with exit code 0
```

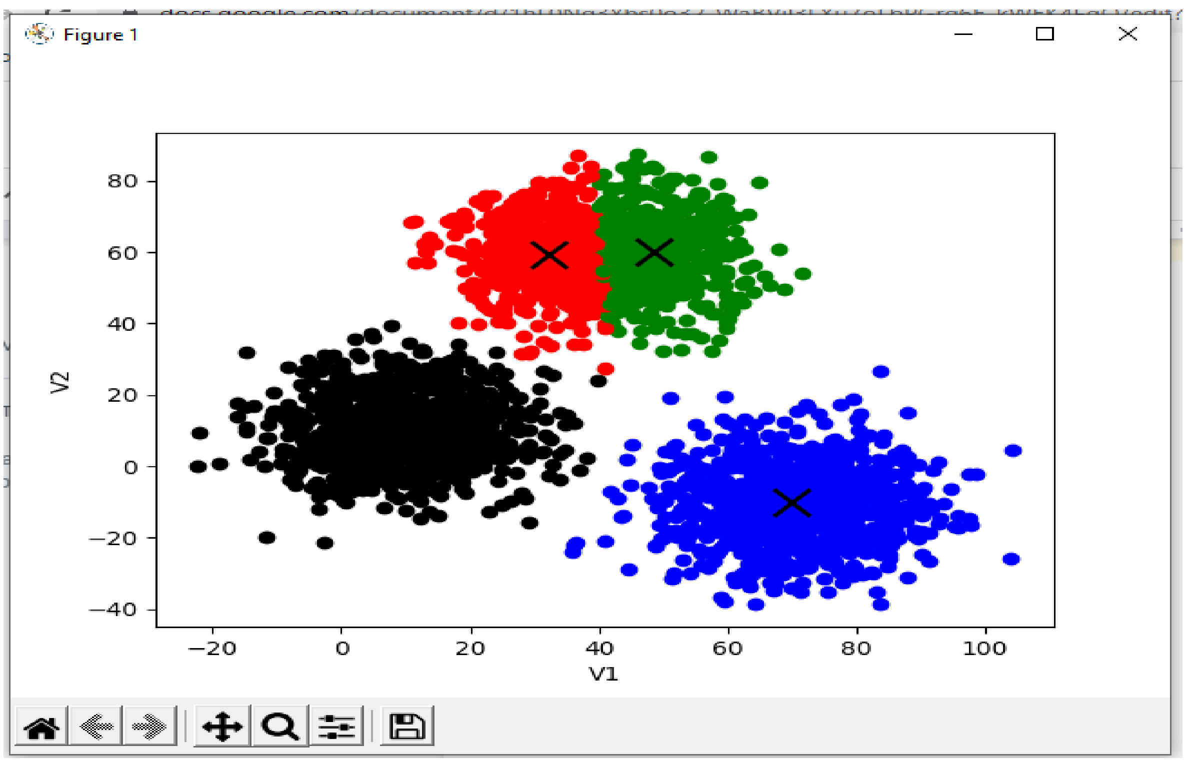
## 4. K Means Clustering

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

data = pd.read_csv('C:/Users/ANKITAGHOSH/Desktop/xclara.csv')
print(data.head())
k = 4
k_mean = KMeans(n_clusters=k)

# training of model
k_mean.fit(data)

labels = k_mean.labels_
centroids = k_mean.cluster_centers_

# testing data
x_test = [[4.6, 67], [2.88, -60], [4.65, 98], [3.4, 56], [-1.33, 5.6], [4.555, -1.22]]

prediction = k_mean.predict(x_test)
print(prediction)

colours = ['Blue', 'Red', 'Green', 'Black']
y = 0
for x in labels:
    plt.scatter(data.iloc[y, 0], data.iloc[y, 1], color=colours[x])
    y += 1
```

```python
for x in range(k):
    lines = plt.plot(centroids[x, 0], centroids[x, 1], 'kx')
    plt.setp(lines, ms=15.0)
    plt.setp(lines, mew=2.0)

title = 'Number of clusters (k) = {}'.format(k)
plt.xlabel('V1')
plt.ylabel('V2')
plt.show()
```

The Graph made

```
          V1         V2
0   2.072345  -3.241693
1  17.936710  15.784810
2   1.083576   7.319176
3  11.120670  14.406780
4  23.711550   2.557729
[1 3 1 1 3 3]
```



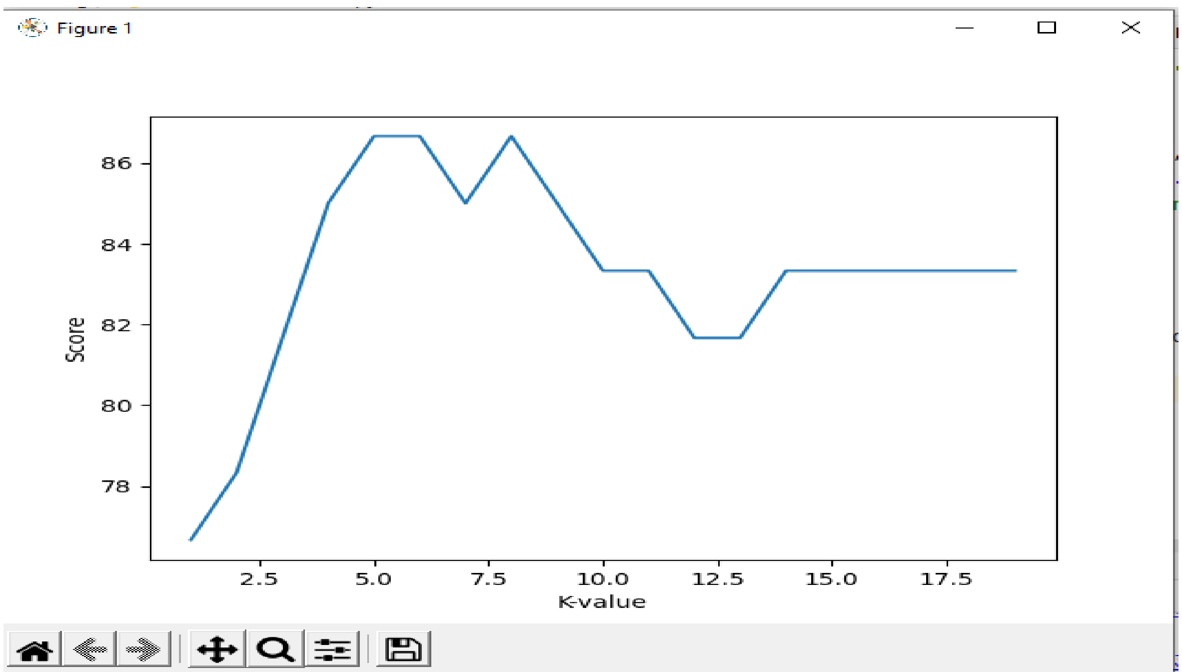
## 5. Using Various models on Cleveland Heart Disease Dataset

C:\Users\ANKITAGHOSH\PycharmProjects\MachineLearning\matplotlibbbbbb p2.py

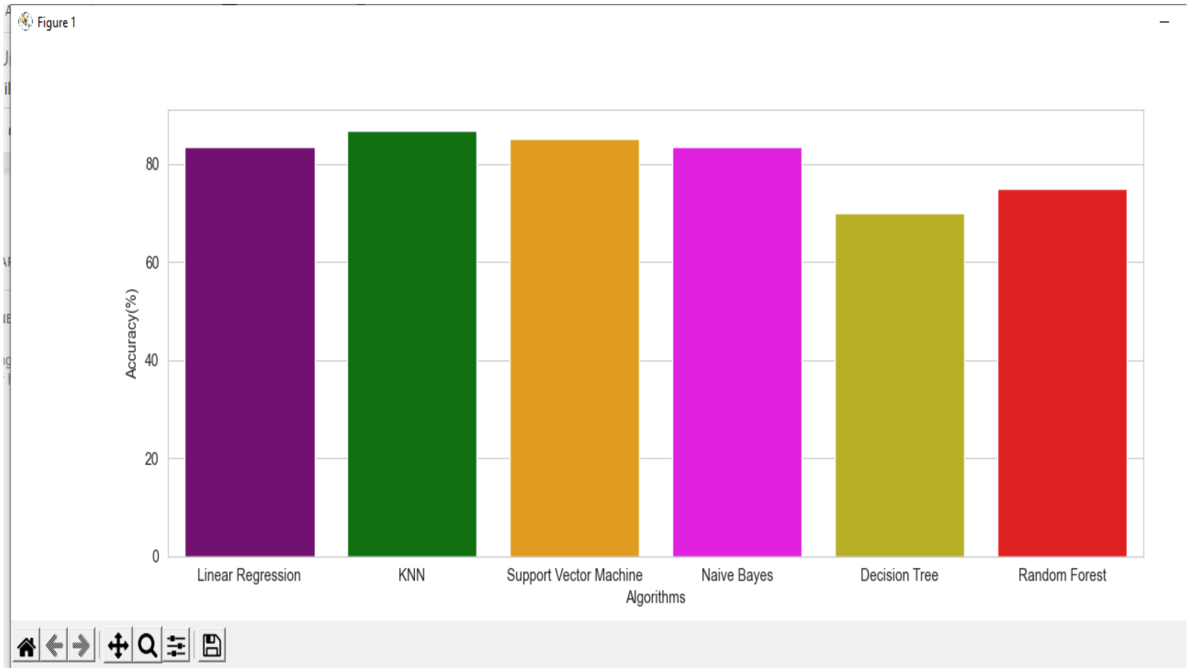
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import seaborn as sb

data = pd.read_csv('C:/Users/ANKITAGHOSH/Desktop/heart_cleveland_upload.csv')

y = data.condition.values
x_data = data.drop(['condition'], axis=1)

# normalizing data means converting all the data in x variable into 0 or 1
x = (x_data - np.min(x_data))/(np.max(x_data) - np.min(x_data)).values

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, train_size=0.8, random_state=0)

# logistic regression
lr = LogisticRegression()
lr.fit(x_train, y_train)
print('Test accuracy of logistic regression : {}'.format(lr.score(x_test, y_test)*100))

# K nearest neighbours
score_list = []
for i in range(1, 20):
    knn = KNeighborsClassifier(i)
    knn.fit(x_train, y_train)
    prediction = knn.predict(x_test)
    score_list.append(knn.score(x_test, y_test)*100)

plt.plot(range(1, 20), score_list)
plt.xlabel('K-value')
plt.ylabel('Score')
plt.show()
print('Test Accuracy of knn is : {}'.format(max(score_list)))

# support vector machine classification
svm = SVC(random_state=1)
svm.fit(x_train, y_train)
print('Test Accuracy of svm is : {}'.format(svm.score(x_test, y_test)*100))

# Naive Bayes Classification
nb = GaussianNB()
nb.fit(x_train, y_train)
print('Test Accuracy of nb is : {}'.format(nb.score(x_test, y_test)*100))

# decision tree classification
```

```python
# decision tree classification
dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)
print('Test Accuracy of dt is : {}'.format(dt.score(x_test, y_test)*100))

# random forest classifier
rf = RandomForestClassifier(n_estimators=1000, random_state=1)
rf.fit(x_train, y_train)
print('Test Accuracy of rf is : {}'.format(rf.score(x_test, y_test)*100))


classifiers = ['Linear Regression', 'KNN', 'Support Vector Machine', 'Naive Bayes', 'Decision Tree', 'Random Forest']
Accuracy = [83.33, 86.66, 85.0, 83.33, 70.0, 75.0]
colours = ['Purple', 'Green', 'Orange', 'Magenta', '#CFC60E', 'Red']
sb.set_style('whitegrid')
plt.figure(figsize=(16, 5))
plt.ylabel("Accuracy(%)")
plt.xlabel("Algorithms")
sb.barplot(x=classifiers, y=Accuracy, palette=colours)
plt.show()
```

The result and graph




```
Test accuracy of logistic regression : 83.33333333333334
Test Accuracy of knn is : 86.66666666666667
Test Accuracy of svm is : 85.0
Test Accuracy of nb is : 83.33333333333334
Test Accuracy of dt is : 66.66666666666666
Test Accuracy of rf is : 75.0
```




## 6. Apriori Algorithm

```python
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
import pandas as pd
dataset = [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
           ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
           ['Milk', 'Apples', 'Kidney Beans', 'Eggs'],
           ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
           ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice Cream', 'Eggs']]
te = TransactionEncoder()
Trans_array = te.fit(dataset).transform(dataset)
df = pd.DataFrame(Trans_array, columns=te.columns_)
print(df)
ap = apriori(df, min_support=0.6, use_colnames=True)
print(ap)
ap['Length'] = ap['itemsets'].apply(lambda x: len(x))
print(ap)
print(ap[(ap['Length'] == 2) & (ap['support'] >= 0.8)])
```

The result

```
C:\Users\ANKITAGHOSH\AppData\Local\Microsoft\WindowsApps\python3.7.exe "C:/Users/ANKITAGHOSH/PycharmProjects/MachineLearning/Apriori Algorithm.py
   Apples   Corn   Dill   Eggs  ...  Nutmeg  Onion  Unicorn  Yogurt
0   False  False  False   True  ...    True   True    False    True
1   False  False   True   True  ...    True   True    False    True
2    True  False  False   True  ...   False  False    False   False
3   False   True  False  False  ...   False  False     True    True
4   False   True  False   True  ...   False   True    False   False

[5 rows x 11 columns]
    support                     itemsets
0       0.8                       (Eggs)
1       1.0               (Kidney Beans)
2       0.6                       (Milk)
3       0.6                      (Onion)
4       0.6                     (Yogurt)
5       0.8         (Kidney Beans, Eggs)
6       0.6                (Eggs, Onion)
7       0.6         (Kidney Beans, Milk)
8       0.6        (Kidney Beans, Onion)
9       0.6       (Yogurt, Kidney Beans)
10      0.6  (Kidney Beans, Eggs, Onion)
    support                     itemsets  Length
0       0.8                       (Eggs)       1
1       1.0               (Kidney Beans)       1
2       0.6                       (Milk)       1
3       0.6                      (Onion)       1
4       0.6                     (Yogurt)       1
5       0.8         (Kidney Beans, Eggs)       2
6       0.6                (Eggs, Onion)       2
7       0.6         (Kidney Beans, Milk)       2
8       0.6        (Kidney Beans, Onion)       2
9       0.6       (Yogurt, Kidney Beans)       2
10      0.6  (Kidney Beans, Eggs, Onion)       3
   support              itemsets  Length
5      0.8  (Kidney Beans, Eggs)       2

Process finished with exit code 0
```