PROJECT REPORT

MEASURING CONSUMER'S ERCEPTIONS OF QUALITY OFFERED BY COFFEE BARS

MACHINE LEARNING AND ITS APPLICATIONS IN CYBER SECURITY
ORGANIZED BY:
DEPARTMENT OF INFORMATION TECHNOLOGY
INDIRA GANDHI DELHI TECHNICAL UNIVERSITY FOR WOMEN

SUBMITTED BY:

DIPTI (01604092021)

NISHA SINHA (03604092021)

PRANSHU YADAV (04204092021)

DIVYA AWASTHI (07104092021)

ACKNOWLEDGEMENT

We would like to express a deep sense of gratitude to our mentor Ms. Pooja Gambhir and course instructor Dr. Santanoo Pattnaik for teaching and guiding us immensely through the duration of our course. We have been able to understand and grasp many new and interesting concepts of Machine Learning under his guidance. His constructive advice & constant motivation has been responsible for the successful completion of this project. We are deeply grateful to Honourable Vice Chancellor Dr. Amita Dev ma'am for providing me this wonderful opportunity which has helped us in growing and developing our skills.

INDEX

ABSTRACT

This Project is about “Measuring Consumer’s Perceptions of Quality Offered By Coffee Bars” that allows to predict the quality of coffee of different coffee bars on the basis of different parameters. A set of 14 parameters was selected based on which the quality of coffee could be predicted accurately. The training dataset was cleaned and feature extraction was performed using python libraries to give the most accurate result. We have used Machine Learning Algorithms, Gaussian algorithm and linear regression to test our model for any shortcomings. This model can be applied in real time to help the coffee Industry to rate their coffee in systematic and mathematical manner.

INTRODUCTION

Colombia is the third-largest producer of coffee in the world and the main producer of Arabian coffee, recognized by its coffee quality, which goes through rigorous selection processes and a final qualification that allows its definition for exportation. Coffee exports in Colombia for 2017 were 710,440 Metric tons and 710,836 for 2018, representing 7% and 5% of total exports for their respective years. Colombian coffee is produced by more than 560,000 small coffee producers, who are grouped in the National Federation of Coffee Growers (FNC) and follow standards to guarantee coffee quality.

To assess the quality of coffee, professional tasters are required, specialized in recognizing the different cup coffee features, who transfer their sensory experience to a score. His knowledge is based on empirical models of interpretation of coffee quality, however, human errors can occur in some of the measurements made which require standardized statistical methods, forms, and analyses, in addition to the experience of the tasters in interpreting the results. It is necessary to find statistical methods based on numerical measurements of physicochemical properties that can be performed by a wider range of professionals, with reproducible results and with less variation. Alternatively, ML techniques emerge as a novel alternative given their ability to emulate human activities and operations.

OBJECTIVE

Coffee is one of the most popular beverages in the world. According to the National Coffee Association (NCA), 58% of American consumers surveyed for its 2011 “National Coffee Drinking Trends” report said they had drunk coffee the previous day. It is available in numerous traditional and gourmet varieties and blends and in many forms, including whole beans, ground, instant, and ready-to-drink beverages.

Coffee Production & Processing of about 60 species of coffee trees, two dominate world trade: Coffee Arabica (referred to as Arabica), which constitutes 75% of production, and Coffee canephora (called Robusta). Coffees from the three main growing regions—Latin America, Southeast Asia, and Africa—have distinctively different flavour characteristics. We are focusing on the major specie of coffee, which is Arabica for our model to assess the quality.

GENERAL INFORMATION ABOUT QUALITY PARAMETERS

- **Aroma:** When we talk about the “flavour” of coffee, it’s easy to think we’re referring solely to the way it tastes. However, with more than 40 aromatic compounds in every roasted coffee bean, aroma not only affects perceptions of flavour, it can also tell us a lot about the growing conditions, roast profile, and processing methods of the coffee beans themselves.
- **Flavour:** Flavour in coffee is made by enhancing the coffee beans with a particular flavour. People can find many different flavoured coffee blends, such as cinnamon, vanilla, caramel, coconut, etc. After the coffee beans have been roasted, the flavour compounds will be added to give the beans a special flavour.
- **Aftertaste:** Aftertaste is the collection of flavours that linger on the palate after you’ve swallowed. Flavour is not a static experience, so flavour can’t be summed up in one word. Flavour changes, from the smell of the coffee, to the first sip, to the aftertaste that remains. As such, it’s essential to consider the ways flavour changes as we drink our coffee.
- **Acidity:** When we talk about the “flavour” of coffee, it’s easy to think we’re referring solely to the way it tastes. However, with more than 40 aromatic compounds in every roasted coffee bean, aroma not only affects perceptions of flavour, it can also tell us a lot about the growing conditions, roast profile, and processing methods of the coffee beans themselves. Acidity is generally very noticeable and can be described as being sweet, crisp and/or tart, somewhat like a dry wine. This enhances other qualities in the coffee.
- **Body:** Body is a measure of the coffee's viscosity (thickness), which contributes to a sensation of the coffee's richness, including its aroma and flavour. A coffee's body is largely created by the coffee beans' oils and organic acids which are extracted during the brewing process.
- **Balance:** The balance of coffee determines the aftertaste that it leaves in the mouth of the drinker. This aftertaste is determined by the correct balance of acidity and moisture content. Hence, the balance of coffee makes impact, because if the different composites are not balanced or the ingredients are not in a correct proportion, the balance is toppled which leads to a bitter or too sweet aftertaste.
- **Uniformity:** The roasting of coffee under correct temperature and technique ensures that it is not too raw or charred for drinking. Therefore, a uniform roasting and infusion time leads to a uniform and well balanced coffee.
- **Serving Hygiene (Cup-cleanliness):** Serving hygiene plays an important role in the overall satisfaction of the customer of a coffee bar as drinking out of a tidy and clean cup is always better and more hygienic than a dirty or spoiled coffee. An untidy cup might contain bacteria and thus further lead to health issues for the customer and thus lead to a decline in number of customers.
- **Sweetness:** In coffee the sweetness is produced by solutions of sugars, glycols, and alcohols as well as some amino acids that together create a variety of sweet aroma descriptors. Generally, lighter roasts will have more of a fruity sweetness. If the coffee is too sweet or too bitter, the quality is compromised and thus achieving a balance between both is of high importance.

- **Moisture:** Being hygroscopic, instant coffee particles are susceptible to the action of moisture – that is, they absorb moisture from the air. If the moisture content increases to 7–8%, the powder or granules can become a paste or solid mass. Hence, the moisture content should be in the limit and not make the coffee stale.
- **Defects:** Defects in coffee refers to the shortcomings that a cup of coffee has at a coffee bar. It could be the lack of aftertaste or the lack of balance between the ingredients. Any defect in the parameters mentioned is a back step for the quality and thus any defect is deducted from the rating of the quality in order to make it more precise and apt.

TOOLS AND LIBRARIES

Language

- Python

Platform

- Google Colab

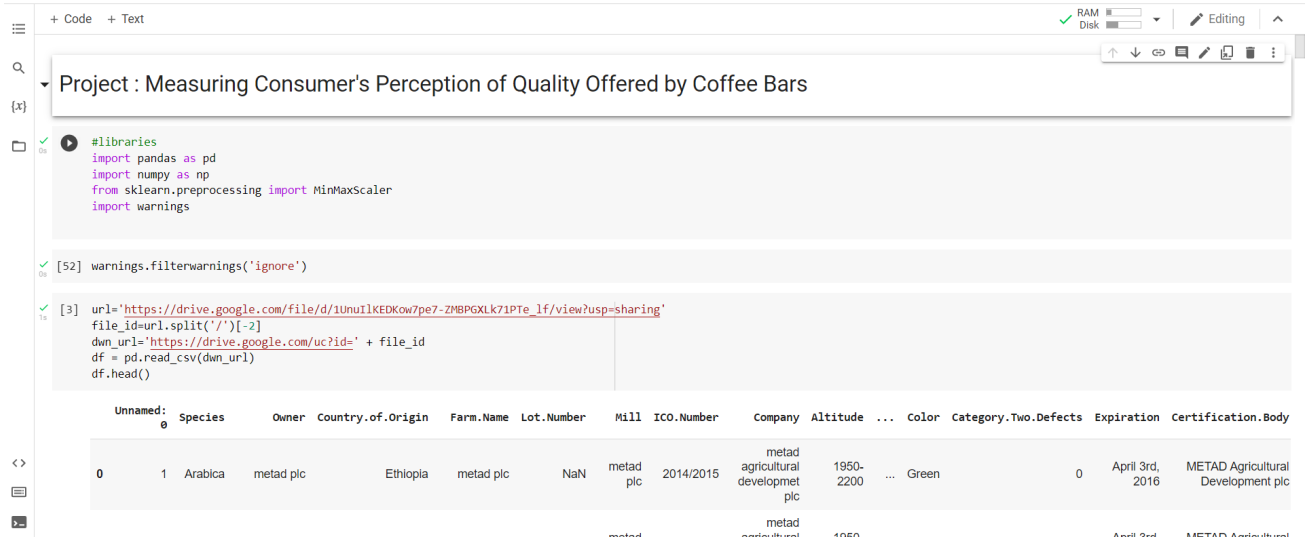
Libraries

- Pandas: Pandas is an open-source library that is made mainly for working with relational or labelled data both easily and intuitively.
- Seaborn: Seaborn is an open-source Python library built on top of matplotlib. It is used for data visualization and exploratory data analysis. Seaborn works easily with data frames and the Pandas library.
- Numpy: NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.
- Matplotlib: Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications.
- Sklearn: Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, and clustering and dimensionality reduction via a consistence interface in Python.

PROCESS

1. Data collection
2. Data pre-processing
3. Data analysis and visualisation
4. Model selection, training and testing
5. Model Evaluation

CODE AND OUTPUT



+ Code + Text

RAM Disk

Editing

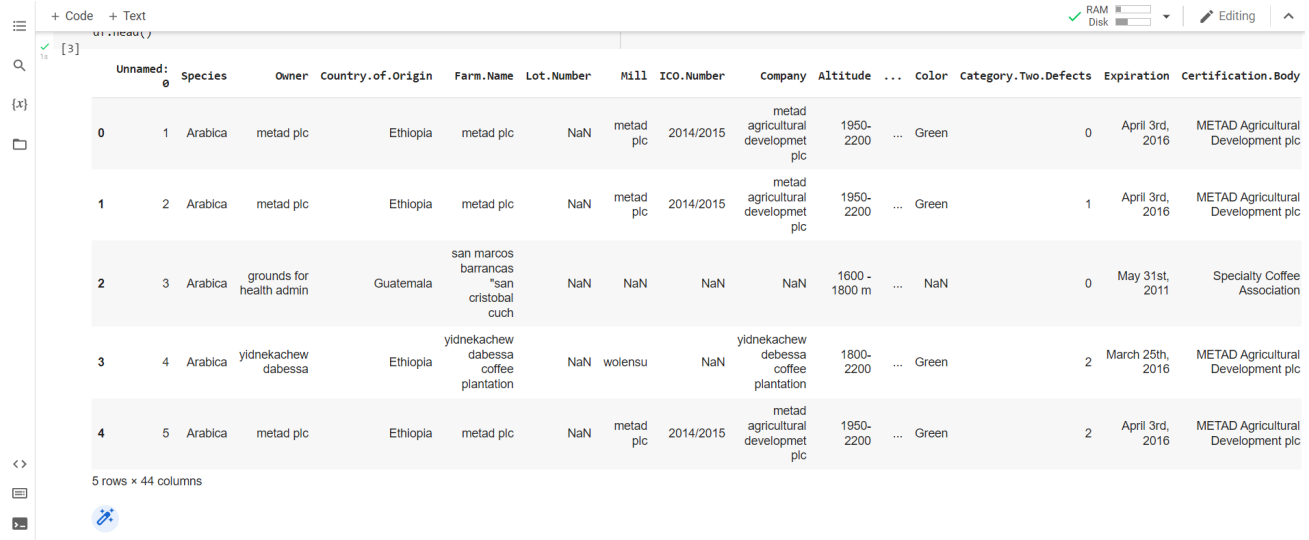
Project : Measuring Consumer's Perception of Quality Offered by Coffee Bars

```
#libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
import warnings
```

```
[52] warnings.filterwarnings('ignore')
```

```
[3] url='https://drive.google.com/file/d/1UnuIlKEDKow7pe7-ZMBPGXLk71PTe_lf/view?usp=sharing'
    file_id=url.split('/')[-2]
    dwn_url='https://drive.google.com/uc?id=' + file_id
    df = pd.read_csv(dwn_url)
    df.head()
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Category.Two.Defects	Expiration	Certification.Body
0	1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	0	April 3rd, 2016	METAD Agricultural Development plc



+ Code + Text

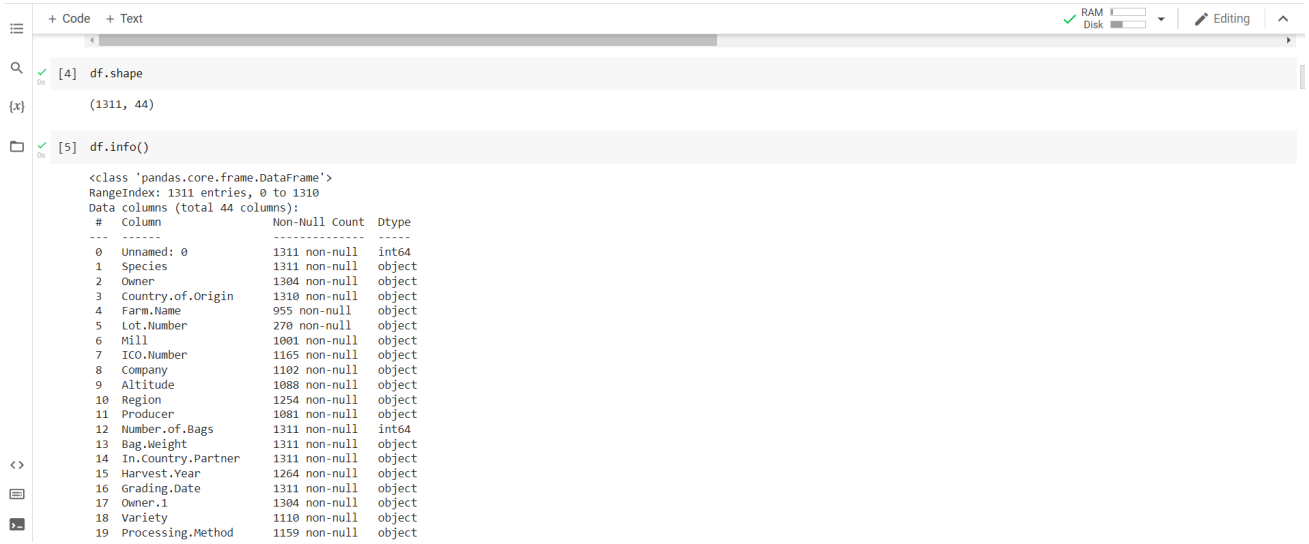
RAM Disk

Editing

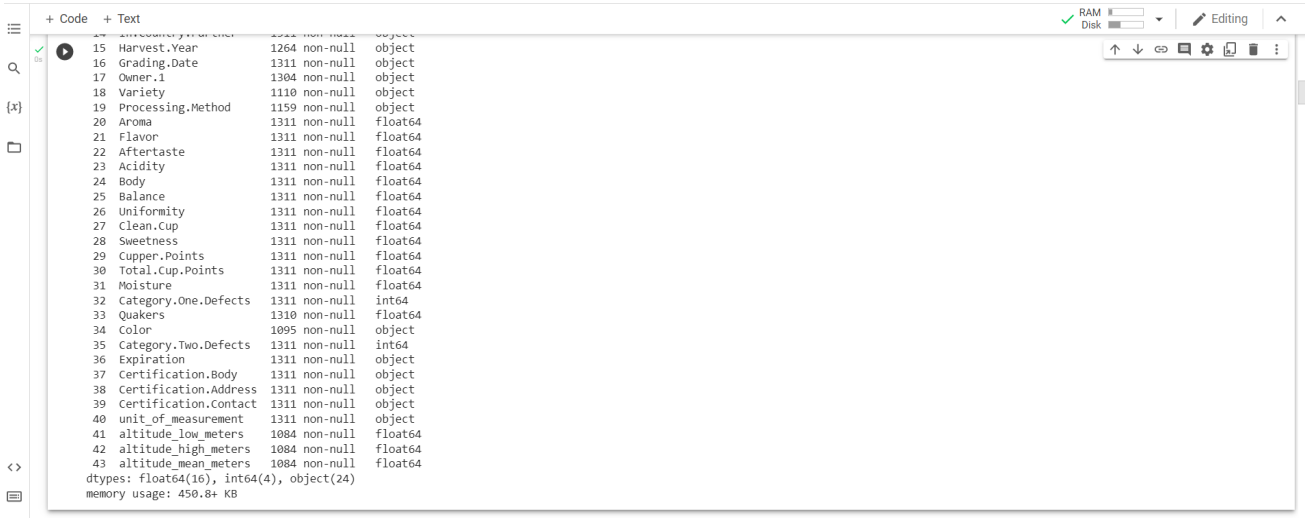
```
[3]
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Category.Two.Defects	Expiration	Certification.Body
0	1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	0	April 3rd, 2016	METAD Agricultural Development plc
1	2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	1	April 3rd, 2016	METAD Agricultural Development plc
2	3	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN	NaN	NaN	1600 - 1800 m	...	NaN	0	May 31st, 2011	Specialty Coffee Association
3	4	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolensu	NaN	yidnekachew debessa coffee plantation	1800-2200	...	Green	2	March 25th, 2016	METAD Agricultural Development plc
4	5	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	2	April 3rd, 2016	METAD Agricultural Development plc

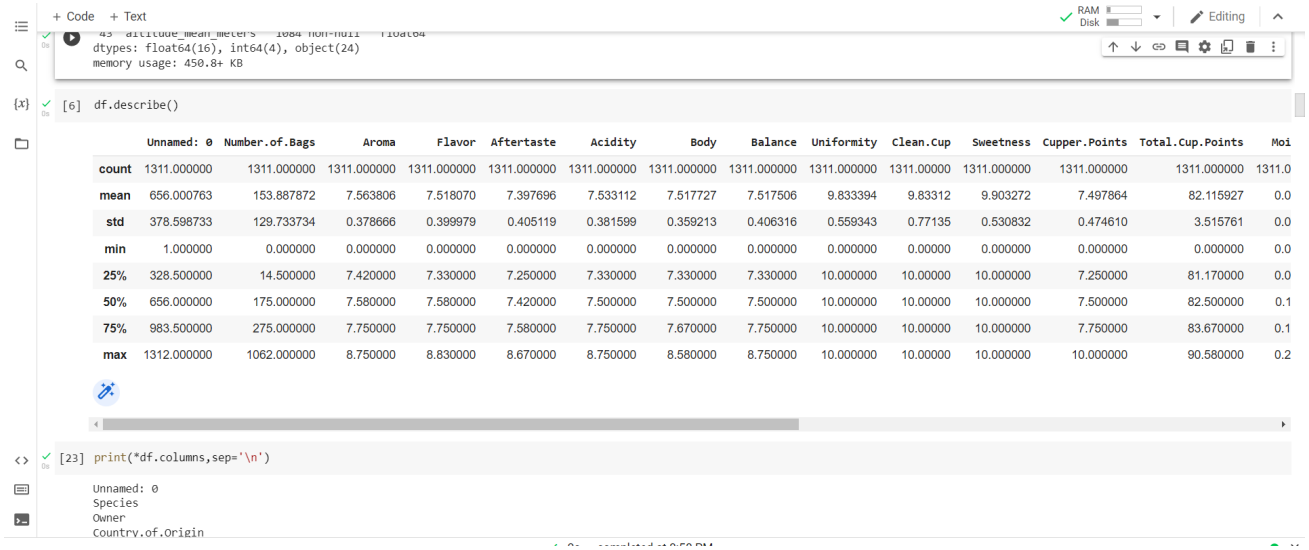
5 rows × 44 columns



```
[4] df.shape

(1311, 44)

[5] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1311 entries, 0 to 1310
Data columns (total 44 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             1311 non-null   int64
 1   Species                1311 non-null   object
 2   Owner                  1304 non-null   object
 3   Country.of.Origin      1310 non-null   object
 4   Farm.Name              955 non-null    object
 5   Lot.Number             270 non-null    object
 6   Mill                   1001 non-null   object
 7   ICO.Number             1165 non-null   object
 8   Company                1102 non-null   object
 9   Altitude               1088 non-null   object
 10  Region                 1254 non-null   object
 11  Producer               1081 non-null   object
 12  Number.of.Bags         1311 non-null   int64
 13  Bag.Weight             1311 non-null   object
 14  In.Country.Partner     1311 non-null   object
 15  Harvest.Year           1264 non-null   object
 16  Grading.Date           1311 non-null   object
 17  Owner.1                1304 non-null   object
 18  Variety                1110 non-null   object
 19  Processing.Method      1159 non-null   object
```



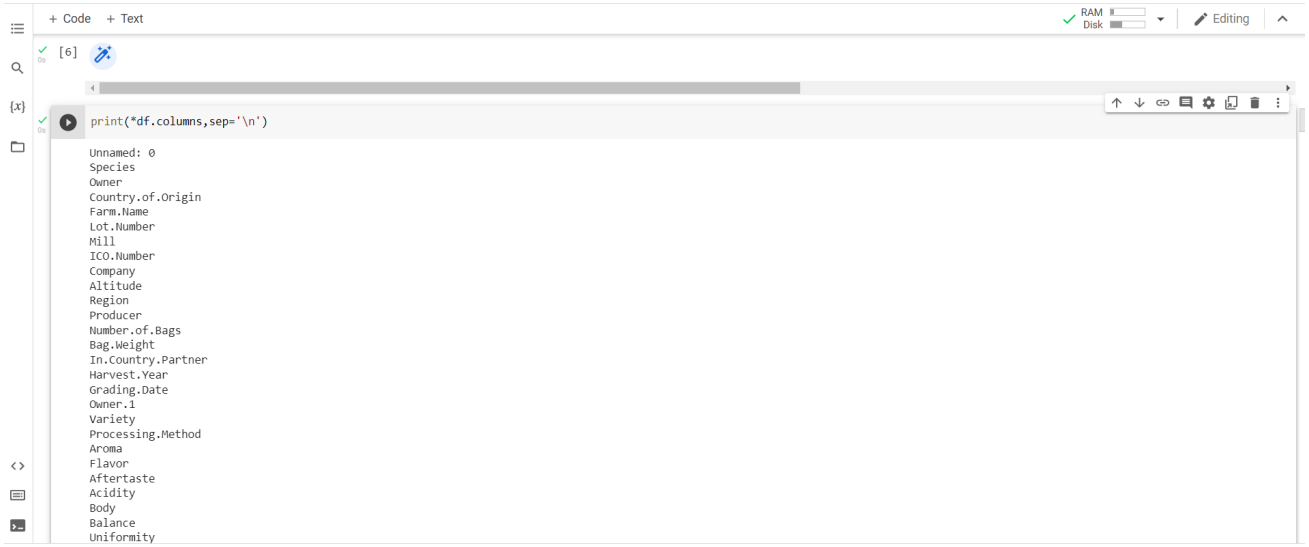
```
 15  Harvest.Year           1264 non-null   object
 16  Grading.Date           1311 non-null   object
 17  Owner.1                1304 non-null   object
 18  Variety                1110 non-null   object
 19  Processing.Method      1159 non-null   object
 20  Aroma                  1311 non-null   float64
 21  Flavor                 1311 non-null   float64
 22  Aftertaste             1311 non-null   float64
 23  Acidity                1311 non-null   float64
 24  Body                   1311 non-null   float64
 25  Balance                1311 non-null   float64
 26  Uniformity             1311 non-null   float64
 27  Clean.Cup              1311 non-null   float64
 28  Sweetness              1311 non-null   float64
 29  Cupper.Points          1311 non-null   float64
 30  Total.Cup.Points       1311 non-null   float64
 31  Moisture               1311 non-null   float64
 32  Category.One.Defects   1311 non-null   int64
 33  Quakers                1310 non-null   float64
 34  Color                  1095 non-null   object
 35  Category.Two.Defects   1311 non-null   int64
 36  Expiration             1311 non-null   object
 37  Certification.Body     1311 non-null   object
 38  Certification.Address  1311 non-null   object
 39  Certification.Contact  1311 non-null   object
 40  unit_of_measurement    1311 non-null   object
 41  altitude_low_meters    1084 non-null   float64
 42  altitude_high_meters   1084 non-null   float64
 43  altitude_mean_meters   1084 non-null   float64
dtypes: float64(16), int64(4), object(24)
memory usage: 450.8+ KB
```



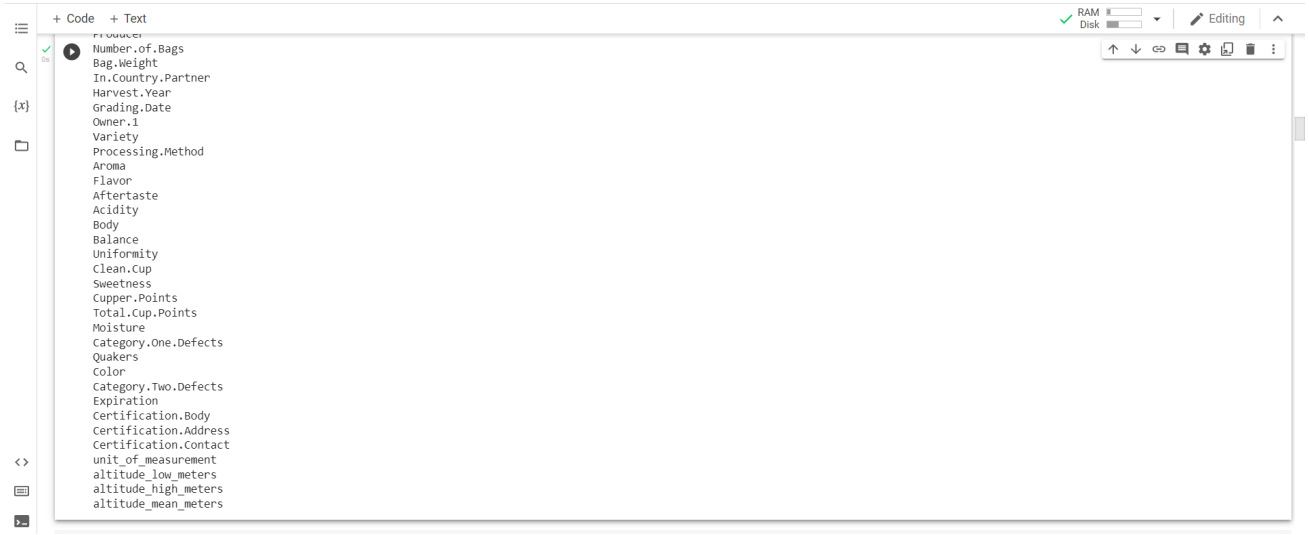
```
dtypes: float64(16), int64(4), object(24)
memory usage: 450.8+ KB

[6] df.describe()
```

	Unnamed: 0	Number.of.Bags	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Clean.Cup	Sweetness	Cupper.Points	Total.Cup.Points	Moi
count	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000	1311.00000	1311.000000	1311.000000	1311.000000	1311.0
mean	656.000763	153.887872	7.563806	7.518070	7.397696	7.533112	7.517727	7.517506	9.833394	9.83312	9.903272	7.497864	82.115927	0.0
std	378.598733	129.733734	0.378666	0.399979	0.405119	0.381599	0.359213	0.406316	0.559343	0.77135	0.530832	0.474610	3.515761	0.0
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.0
25%	328.500000	14.500000	7.420000	7.330000	7.250000	7.330000	7.330000	7.330000	10.000000	10.00000	10.000000	7.250000	81.170000	0.0
50%	656.000000	175.000000	7.580000	7.580000	7.420000	7.500000	7.500000	7.500000	10.000000	10.00000	10.000000	7.500000	82.500000	0.1
75%	983.500000	275.000000	7.750000	7.750000	7.580000	7.750000	7.670000	7.750000	10.000000	10.00000	10.000000	7.750000	83.670000	0.1
max	1312.000000	1062.000000	8.750000	8.830000	8.670000	8.750000	8.580000	8.750000	10.000000	10.00000	10.000000	10.000000	90.580000	0.2

```
[23] print(*df.columns,sep='\n')

Unnamed: 0
Species
Owner
Country.of.Origin
```



```
[6]
print(*df.columns,sep='\n')

Unnamed: 0
Species
Owner
Country.of.Origin
Farm.Name
Lot.Number
Mill
ICO.Number
Company
Altitude
Region
Producer
Number.of.Bags
Bag.Weight
In.Country.Partner
Harvest.Year
Grading.Date
Owner.1
Variety
Processing.Method
Aroma
Flavor
Aftertaste
Acidity
Body
Balance
Uniformity
```



```
Number.of.Bags
Bag.Weight
In.Country.Partner
Harvest.Year
Grading.Date
Owner.1
Variety
Processing.Method
Aroma
Flavor
Aftertaste
Acidity
Body
Balance
Uniformity
Clean.Cup
Sweetness
Cupper.Points
Total.Cup.Points
Moisture
Category.One.Defects
Quakers
Color
Category.Two.Defects
Expiration
Certification.Body
Certification.Address
Certification.Contact
unit_of_measurement
altitude_low_meters
altitude_high_meters
altitude_mean_meters
```

```
[8] data = df.copy(deep = True)
```

Quality Measures

- Aroma
- Flavor
- Aftertaste
- Acidity
- Body
- Balance
- Uniformity
- Cup Cleanliness
- Sweetness
- Moisture
- Defects

```
[9] data = data.drop(columns = ['Unnamed: 0','Species', 'Owner', 'Country.of.Origin', 'Farm.Name','Lot.Number', 'Mill', 'ICO.Number', 'Company', 'Altitude', 'Region',
                                'Producer', 'Number.of.Bags', 'Bag.Weight', 'In.Country.Partner','Harvest.Year', 'Grading.Date', 'Owner.1', 'Variety', 'Processing.Method',
                                'Expiration', 'Certification.Body', 'Certification.Address', 'Certification.Contact', 'unit_of_measurement', 'altitude_low_meters',
                                'altitude_high_meters', 'altitude_mean_meters','Quakers','Color'])

    print(data.columns)
    print(data.head(5))
```

+ Code + Text

RAM Disk | Editing

```
[9]
print(data.columns)
print(data.head(5))
```

```
Index(['Aroma', 'Flavor', 'Aftertaste', 'Acidity', 'Body', 'Balance',
       'Uniformity', 'Clean.Cup', 'Sweetness', 'Cupper.Points',
       'Total.Cup.Points', 'Moisture', 'Category.One.Defects',
       'Category.Two.Defects'],
      dtype='object')
   Aroma  Flavor  Aftertaste  Acidity  Body  Balance  Uniformity  Clean.Cup  \
0   8.67    8.83        8.67     8.75  8.50     8.42        10.0       10.0
1   8.75    8.67        8.50     8.58  8.42     8.42        10.0       10.0
2   8.42    8.50        8.42     8.42  8.33     8.42        10.0       10.0
3   8.17    8.58        8.42     8.42  8.50     8.25        10.0       10.0
4   8.25    8.50        8.25     8.50  8.42     8.33        10.0       10.0

   Sweetness  Cupper.Points  Total.Cup.Points  Moisture  Category.One.Defects  \
0       10.0           8.75             90.58      0.12                     0
1       10.0           8.58             89.92      0.12                     0
2       10.0           9.25             89.75      0.00                     0
3       10.0           8.67             89.00      0.11                     0
4       10.0           8.58             88.83      0.12                     0

   Category.Two.Defects
0                     0
1                     1
2                     0
3                     2
4                     2
```

```
[10] data.describe()
```

+ Code + Text

RAM Disk | Editing

```
[10] data.describe()
```

	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Clean.Cup	Sweetness	Cupper.Points	Total.Cup.Points	Moisture	Category.One.Defects
count	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000	1311.00000	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000
mean	7.563806	7.518070	7.397696	7.533112	7.517727	7.517506	9.833394	9.83312	9.903272	7.497864	82.115927	0.088863	0.426392
std	0.378666	0.399979	0.405119	0.381599	0.359213	0.406316	0.559343	0.77135	0.530832	0.474610	3.515761	0.047957	1.832415
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	7.420000	7.330000	7.250000	7.330000	7.330000	7.330000	10.000000	10.00000	10.000000	7.250000	81.170000	0.090000	0.000000
50%	7.580000	7.580000	7.420000	7.500000	7.500000	7.500000	10.000000	10.00000	10.000000	7.500000	82.500000	0.110000	0.000000
75%	7.750000	7.750000	7.580000	7.750000	7.670000	7.750000	10.000000	10.00000	10.000000	7.750000	83.670000	0.120000	0.000000
max	8.750000	8.830000	8.670000	8.750000	8.580000	8.750000	10.000000	10.00000	10.000000	10.000000	90.580000	0.280000	31.000000

```
[26] col = list(data.columns)
print(*col,sep='\n')
```

```
Aroma
Flavor
Aftertaste
Acidity
Body
```

+ Code + Text

RAM Disk | Editing

```
[26] col = list(data.columns)
print(*col,sep='\n')
```

```
Aroma
Flavor
Aftertaste
Acidity
Body
Balance
Uniformity
Clean.Cup
Sweetness
Cupper.Points
Total.Cup.Points
Moisture
Category.One.Defects
Category.Two.Defects
Rating
```

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer = imputer.fit(data)

data = pd.DataFrame(imputer.transform(data))
```

```
[13] data.shape
```

```
(1311, 14)
```

```
[14] data.columns = col
     data
```

	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Clean.Cup	Sweetness	Cupper.Points	Total.Cup.Points	Moisture	Category.One.Defects	Category.Two.Defects
0	8.67	8.83	8.67	8.75	8.50	8.42	10.00	10.00	10.00	8.75	90.58	0.12	0.0	0.0
1	8.75	8.67	8.50	8.58	8.42	8.42	10.00	10.00	10.00	8.58	89.92	0.12	0.0	1.0
2	8.42	8.50	8.42	8.42	8.33	8.42	10.00	10.00	10.00	9.25	89.75	0.00	0.0	0.0
3	8.17	8.58	8.42	8.42	8.50	8.25	10.00	10.00	10.00	8.67	89.00	0.11	0.0	2.0
4	8.25	8.50	8.25	8.50	8.42	8.33	10.00	10.00	10.00	8.58	88.83	0.12	0.0	2.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1306	7.08	6.83	6.25	7.42	7.25	6.75	10.00	0.00	10.00	6.75	68.33	0.11	0.0	20.0
1307	6.75	6.58	6.42	6.67	7.08	6.67	9.33	6.00	6.00	6.42	67.92	0.14	8.0	16.0
1308	7.25	6.58	6.33	6.25	6.42	6.08	6.00	6.00	6.00	6.17	63.08	0.13	1.0	5.0
1309	7.50	6.67	6.67	7.67	7.33	6.67	8.00	1.33	1.33	6.67	59.83	0.10	0.0	4.0
1310	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.12	0.0	2.0

1311 rows × 14 columns

```
[15] data['Rating'] = data[data.columns[:12]].mean(numeric_only=True, axis=1) - data[data.columns[12:]].mean(numeric_only=True,axis=1)

[16] data['Rating'] = 10*(data['Rating'] - data['Rating'].min())/(data['Rating'].max() - data['Rating'].min())
```

```
[17] data['Rating']

0       10.000000
1        9.851016
2        9.963821
3        9.691667
4        9.684756
           ...
1306     6.656098
1307     6.152236
1308     8.150407
1309     8.261789
1310     6.073780
Name: Rating, Length: 1311, dtype: float64

[18] data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1311 entries, 0 to 1310
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Aroma                 1311 non-null   float64
 1   Flavor                1311 non-null   float64
 2   Aftertaste            1311 non-null   float64
 3   Acidity               1311 non-null   float64
 4   Body                  1311 non-null   float64
 5   Balance               1311 non-null   float64
 6   Uniformity            1311 non-null   float64
 7   Clean.Cup             1311 non-null   float64
 8   Sweetness             1311 non-null   float64
 9   Cupper.Points         1311 non-null   float64
 10  Total.Cup.Points      1311 non-null   float64
```

```
[18]  6   Uniformity            1311 non-null   float64
      7   Clean.Cup             1311 non-null   float64
      8   Sweetness             1311 non-null   float64
      9   Cupper.Points         1311 non-null   float64
      10  Total.Cup.Points      1311 non-null   float64
      11  Moisture              1311 non-null   float64
      12  Category.One.Defects  1311 non-null   float64
      13  Category.Two.Defects  1311 non-null   float64
      14  Rating                1311 non-null   float64
     dtypes: float64(15)
     memory usage: 153.8 KB

[19] data.describe()
```

	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Clean.Cup	Sweetness	Cupper.Points	Total.Cup.Points	Moisture	Category.One.Defects
count	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000	1311.00000	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000
mean	7.563806	7.518070	7.397696	7.533112	7.517727	7.517506	9.833394	9.83312	9.903272	7.497864	82.115927	0.088863	0.426392
std	0.378666	0.399979	0.405119	0.381599	0.359213	0.406316	0.559343	0.77135	0.530832	0.474610	3.515761	0.047957	1.832415
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	7.420000	7.330000	7.250000	7.330000	7.330000	7.330000	10.000000	10.00000	10.000000	7.250000	81.170000	0.090000	0.000000
50%	7.580000	7.580000	7.420000	7.500000	7.500000	7.500000	10.000000	10.00000	10.000000	7.500000	82.500000	0.110000	0.000000
75%	7.750000	7.750000	7.580000	7.750000	7.670000	7.750000	10.000000	10.00000	10.000000	7.750000	83.670000	0.120000	0.000000
max	8.750000	8.830000	8.670000	8.750000	8.580000	8.750000	10.000000	10.00000	10.000000	10.000000	90.580000	0.280000	31.000000

+ Code + Text

[19]														
	75%	7.750000	7.750000	7.580000	7.750000	7.670000	7.750000	10.000000	10.00000	10.000000	7.750000	83.670000	0.120000	0.000000
	max	8.750000	8.830000	8.670000	8.750000	8.580000	8.750000	10.000000	10.00000	10.000000	10.000000	90.580000	0.280000	31.000000

[49]
```
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm
```

[33]
```
corr_mat = data.corr()
corr_list = corr_mat.Rating.abs().sort_values(ascending=False).index[0:]
corr_list
```

```
Index(['Rating', 'Category.Two.Defects', 'Category.One.Defects',
       'Total.Cup.Points', 'Clean.Cup', 'Aftertaste', 'Flavor',
       'Cupper.Points', 'Balance', 'Aroma', 'Acidity', 'Uniformity', 'Body',
       'Sweetness', 'Moisture'],
      dtype='object')
```

[37]
```
corr_list.size
```

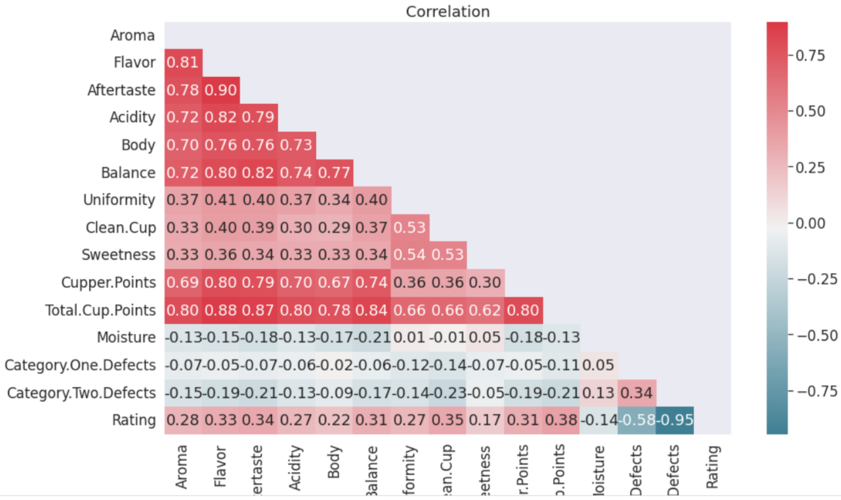
```
15
```

[42]
```
plt.figure(figsize=(15,9))
plt.title('Correlation')
dropSelf = np.zeros_like(corr_mat)
dropSelf[np.triu_indices_from(dropSelf)] = True
```

✓ 0s completed at 8:58 PM

+ Code + Text

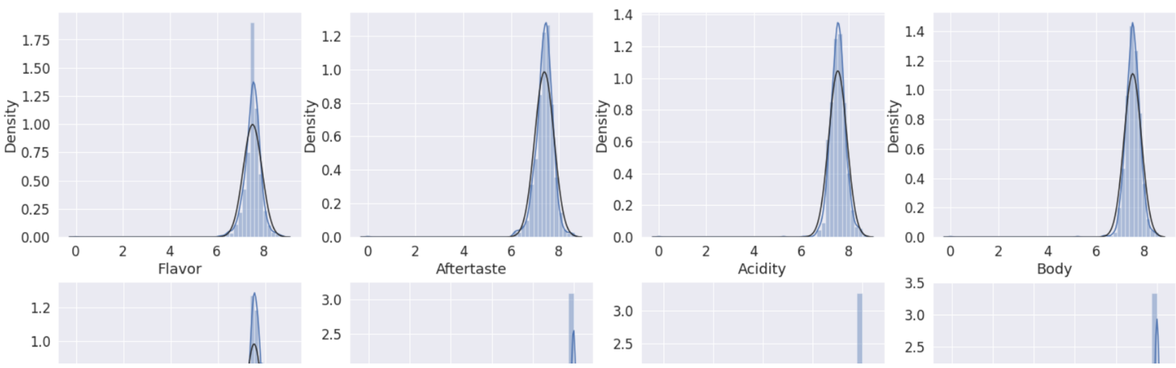
[42]

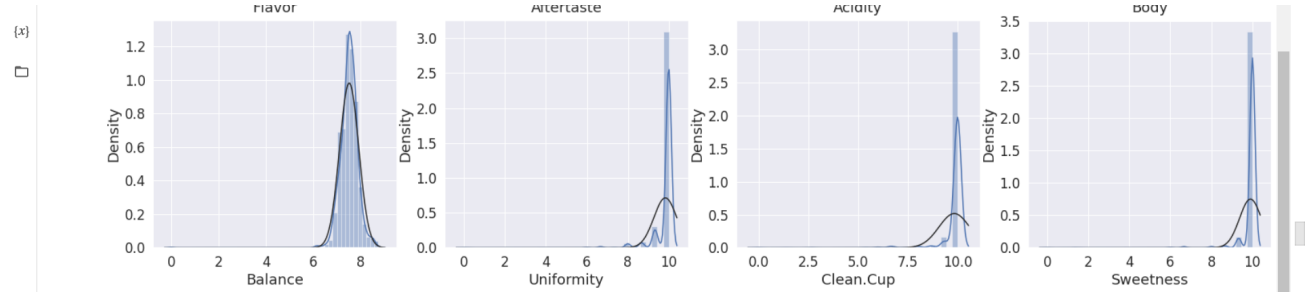


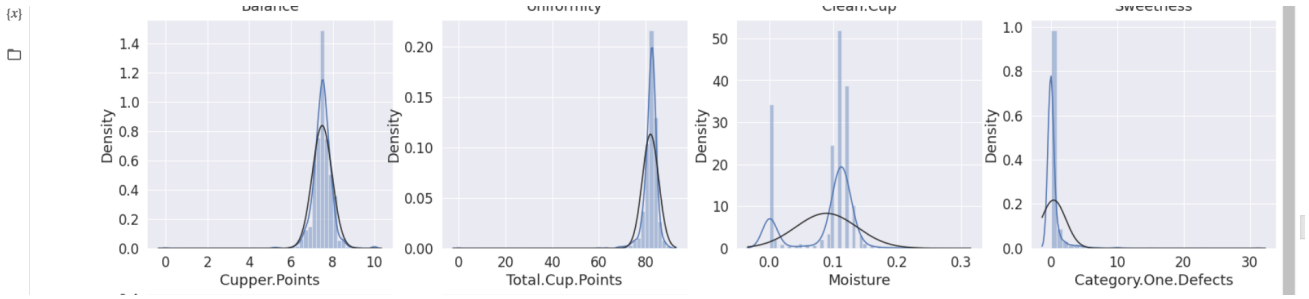
+ Code + Text

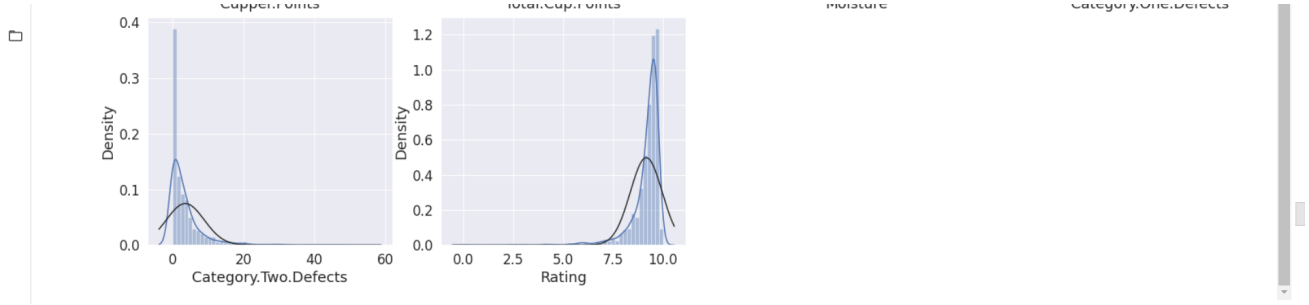
Distribution Of Variables

[59]
```
plt.figure(figsize = (25,30))
for i in range(1,15):
  plt.subplot(5,4,i)
  sns.distplot(data[data.columns[i]],fit=norm)
```









+ Code + Text

RAM Disk | Editing

```
[59]
```



```
[60] from sklearn import linear_model
     from sklearn.model_selection import train_test_split
     from sklearn import metrics
```

```
[61] X = data[data.columns[:-1]]
     Y = data['Rating']

     X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.25, random_state=5)
```

```
[62] # create linear regression object
     reg = linear_model.LinearRegression()

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

     LinearRegression()
```

```
[63] Y_pred = reg.predict(X_test)
     Y_pred[:10]

     array([6.07378049, 9.32560976, 9.75284553, 8.33069106, 8.71260163,
            9.5695122 , 9.67764228, 9.71158537, 9.51199187, 9.34634146])
```

```
[64] print('Accuracy : ',metrics.r2_score(Y_test,Y_pred))
```

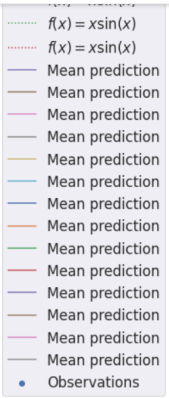
```
[64] print('Accuracy : ',metrics.r2_score(Y_test,Y_pred))

     Accuracy :  1.0

[65] import statsmodels.api as sm

[66] #OLS Model
     regr = linear_model.LinearRegression()
     regr.fit(X_train, Y_train)

     print('Intercept: \n', regr.intercept_)
     print('Coefficients: \n', regr.coef_)

     # with statsmodels
     x_train = sm.add_constant(X_train) # adding a constant
     x_test = sm.add_constant(X_test)
     model = sm.OLS(Y_train, x_train).fit()
     predictions = model.predict(x_test)

     print_model = model.summary()
     print(print_model)

     Intercept:
      6.315243902439025
     Coefficients:
      [ 0.0203252   0.0203252   0.0203252   0.0203252   0.0203252   0.0203252
       0.0203252   0.0203252   0.0203252   0.0203252   0.0203252   0.0203252
      -0.12195122 -0.12195122]
                                 OLS Regression Results
                                 OLS Regression Results
     ==============================================================================
     Dep. Variable:                 Rating   R-squared:                       1.000
     Model:                            OLS   Adj. R-squared:                  1.000
     Method:                 Least Squares   F-statistic:                 7.932e+26
     Date:                Mon, 01 Aug 2022   Prob (F-statistic):               0.00
     Time:                        15:27:47   Log-Likelihood:                 27197.
     No. Observations:                 983   AIC:                        -5.436e+04
     Df Residuals:                     968   BIC:                        -5.429e+04
     Df Model:                          14
     Covariance Type:            nonrobust
     ================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
     --------------------------------------------------------------------------------
     const            6.3152   3.08e-13   2.05e+13      0.000       6.315       6.315
     Aroma            0.0203   4.63e-13   4.39e+10      0.000       0.020       0.020
     Flavor           0.0203   4.66e-13   4.36e+10      0.000       0.020       0.020
     Aftertaste       0.0203   4.66e-13   4.36e+10      0.000       0.020       0.020
     Acidity          0.0203   4.62e-13    4.4e+10      0.000       0.020       0.020
     Body             0.0203   4.67e-13   4.36e+10      0.000       0.020       0.020
     Balance          0.0203   4.65e-13   4.37e+10      0.000       0.020       0.020
     Uniformity       0.0203   4.64e-13   4.38e+10      0.000       0.020       0.020
     Clean.Cup        0.0203   4.63e-13   4.39e+10      0.000       0.020       0.020
     Sweetness        0.0203   4.63e-13   4.39e+10      0.000       0.020       0.020
     Cupper.Points    0.0203   4.62e-13    4.4e+10      0.000       0.020       0.020
     Total.Cup.Points 0.0203   4.63e-13   4.39e+10      0.000       0.020       0.020
     Moisture         0.0203   1.65e-13   1.23e+11      0.000       0.020       0.020
     Category.One.Defects    -0.1220   4.46e-15  -2.74e+13      0.000      -0.122      -0.122
     Category.Two.Defects    -0.1220    1.6e-15   -7.6e+13      0.000      -0.122      -0.122
     ==============================================================================
     Omnibus:                      118.293   Durbin-Watson:                   0.002
     Prob(Omnibus):                  0.000   Jarque-Bera (JB):              382.504
     Skew:                           0.576   Prob(JB):                     8.72e-84
     Kurtosis:                       5.831   Cond. No.                     1.77e+04
     ==============================================================================

     Warnings:
     [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
     [2] The condition number is large, 1.77e+04. This might indicate that there are
     strong multicollinearity or other numerical problems.

[67] # Prediciton using Gaussian Regression Model
     from sklearn.gaussian_process import GaussianProcessRegressor
     from sklearn.gaussian_process.kernels import RBF

     kernel = 1 * RBF(length_scale=1.0, length_scale_bounds=(1e-2, 1e2))
     gaussian_process = GaussianProcessRegressor(kernel=kernel, n_restarts_optimizer=9)
     gaussian_process.fit(X_train, Y_train)
     gaussian_process.kernel_

     3.33**2 * RBF(length_scale=100)
```

```
[68] mean_prediction, std_prediction = gaussian_process.predict(X, return_std=True)

     plt.plot(X, Y, label=r"$f(x) = x \sin(x)$", linestyle="dotted")
     plt.scatter(X_train.iloc[:, 0], Y_train, label="Observations")
     plt.plot(X, mean_prediction, label="Mean prediction")
     plt.legend(bbox_to_anchor=(1.1, 1.05))
     plt.xlabel("$x$")
     plt.ylabel("$f(x)$")
     _ = plt.title("Gaussian process regression on noise-free dataset")
```

+ Code + Text

RAM Disk Editing

```
[68]
```



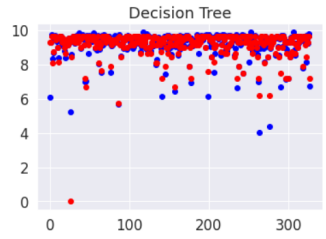

```
[69] # Prediction using Decision Tree Model
     from sklearn.tree import DecisionTreeRegressor
     DT_model = DecisionTreeRegressor(max_depth=5).fit(X_train,Y_train)
     DT_predict = DT_model.predict(X_test) #Predictions on Testing data
     print(DT_predict)
```

+ Code + Text

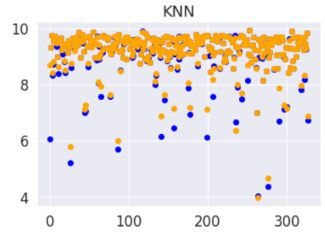
RAM Disk Editing

```
[69] # Prediction using Decision Tree Model
     from sklearn.tree import DecisionTreeRegressor
     DT_model = DecisionTreeRegressor(max_depth=5).fit(X_train,Y_train)
     DT_predict = DT_model.predict(X_test) #Predictions on Testing data
     print(DT_predict)

     [9.30609909 9.30609909 9.6630888  8.10748645 8.73937669 9.6630888
      9.6630888  9.6630888  9.45369467 9.53780604 8.981905   8.12709059
      9.6630888  9.53780604 9.6630888  9.30609909 9.53780604 9.30609909
      9.45369467 8.981905   8.66587575 9.30609909 9.45369467 9.16739185
      9.53780604 9.30609909 0.         8.43297589 9.45369467 9.6630888
      9.45369467 9.30609909 9.30609909 9.53780604 9.6630888  9.53780604
      9.30609909 9.6630888  9.45369467 8.981905   9.53780604 9.6630888
      9.6630888  9.16739185 7.20182927 6.69241192 9.6630888  9.45369467
      9.45369467 9.16739185 9.6630888  9.53780604 9.53780604 9.45369467
      9.6630888  9.6630888  9.45369467 8.981905   9.30609909 9.30609909
      9.6630888  8.10748645 9.30609909 8.981905   9.16739185 7.67085874
      9.6630888  9.53780604 9.6630888  9.6630888  9.30609909 9.6630888
      9.6630888  9.53780604 9.53780604 9.45369467 9.30609909 7.89015679
      9.16739185 9.30609909 9.30609909 9.45369467 9.45369467 9.30609909
      9.45369467 9.53780604 5.76294038 9.6630888  9.16739185 8.43297589
      9.53780604 9.16739185 9.30609909 8.981905   9.53780604 9.45369467
      9.6630888  8.981905   9.6630888  9.53780604 9.6630888  9.16739185
      9.53780604 9.30609909 9.16739185 9.6630888  9.16739185 9.6630888
      9.30609909 9.6630888  9.6630888  9.30609909 9.6630888  9.45369467
      8.66587575 9.16739185 9.45369467 9.6630888  9.16739185 9.45369467
      9.30609909 8.981905   9.6630888  8.66587575 9.6630888  8.981905
      8.981905   8.66587575 9.30609909 9.53780604 9.16739185 9.6630888
      9.53780604 8.10748645 8.10748645 9.16739185 9.53780604 9.45369467
      9.45369467 9.16739185 8.73937669 7.17247387 9.53780604 9.30609909
      9.53780604 7.89015679 8.981905   9.6630888  8.981905   8.66587575
      9.6630888  9.30609909 9.16739185 8.43297589 9.16739185 9.53780604
      9.6630888  6.69241192 9.16739185 9.30609909 9.6630888  8.66587575
      9.45369467 8.981905   9.6630888  9.30609909 9.53780604 8.981905
```

```
[70] print("Accuracy : ",metrics.r2_score(Y_test,DT_predict))

     Accuracy :  0.7509923577898329

[71] if max(Y_test) >= max(DT_predict):
         my_range = int(max(Y_test))
     else:
         my_range = int(max(DT_predict))
     plt.scatter(range(len(Y_test)), Y_test, color='blue')
     plt.scatter(range(len(DT_predict)), DT_predict, color='red')
     plt.title("Decision Tree")
     plt.show()
```




```
[72] # Prediction using KNN Model
     from sklearn.neighbors import KNeighborsRegressor
     KNN_model = KNeighborsRegressor(n_neighbors=3).fit(X_train,Y_train)
     KNN_predict = KNN_model.predict(X_test) #Predictions on Testing data
     print(KNN_predict)

     [8.69905149 9.32920054 9.75345528 8.41233062 8.80311653 9.57079946
      9.67838753 9.70995935 9.51172087 9.4751355  9.00176152 8.62906504
      9.56192412 9.63333333 9.66436314 9.23685637 9.33536585 9.4399729
      9.33821138 8.8403794  8.51856369 9.45250678 9.44234417 9.14952575
      9.56714092 9.31795393 5.78367209 8.7451897  9.57594851 9.66673442
      9.48292683 9.20054201 9.45121951 9.34911924 9.68976965 9.63739837
      9.41944444 9.86253388 9.49342818 8.95074526 9.60616531 9.77073171
      9.52249322 9.16395664 7.12066396 7.26300813 9.69288618 9.35433604
      9.35948509 8.73855014 9.72554201 9.64200542 9.51422764 9.33509485
      9.57533875 9.86253388 9.49329268 8.97093496 9.4350271  8.96544715
      9.68976965 8.08726287 9.40697832 8.96781843 9.07025745 7.93922764
      9.70182927 9.23014905 9.73658537 9.6903794  9.30311653 9.68150407
      9.67289973 9.60697832 9.64017615 9.22398374 9.33028455 7.63075881
      9.09281843 9.22865854 9.26680217 9.43380759 9.71212737 9.30643631
      9.45901084 9.61415989 5.99810298 9.76185637 9.17974255 8.46436314
      9.57256098 9.18448509 9.24884824 8.95088076 9.56382114 9.46402439
      9.67025745 8.90670732 9.6949187  9.52208672 9.6251355  9.26639566
      9.60135501 9.44803523 9.01382114 9.57777778 9.15846883 9.56720867
      9.42174797 9.53733062 9.6298103  9.35101626 9.58895664 9.52676152
      8.75765583 9.08279133 9.47696477 9.86253388 9.25826558 9.49342818
      9.35101626 8.94295393 9.81348238 8.85731707 9.69180217 9.01565041
      8.8153794  8.79390244 9.28882114 9.60609756 9.18150407 9.66551491
      9.5053523  8.11084011 8.39403794 9.39647696 9.25826558 9.5851626
      9.33495935 9.20284553 9.02439024 6.88468835 9.50528455 9.24369919
      9.43658537 9.27825203 9.82093496 6.87073171]

[73] print("Accuracy : ",metrics.r2_score(Y_test,KNN_predict))

     Accuracy :  0.9447626358018536

[74] if max(Y_test) >= max(KNN_predict):
         my_range = int(max(Y_test))
     else:
         my_range = int(max(KNN_predict))
     plt.scatter(range(len(Y_test)), Y_test, color='blue')
     plt.scatter(range(len(KNN_predict)), KNN_predict, color='orange')
     plt.title("KNN")
     plt.show()
```



CONCLUSION

This project uses the Arabica species of coffee to predict the quality of the coffee based on the various parameters. In this project, we checked the general characteristics of the dataset. Data has some NULL values. Instead of dropping missing values we completed them by the mean of data as the percentage of missing values was considerably low. We also looked at quality levels in each variable by using suitable charts for a general understanding. The accuracy of the Linear Regression algorithm came out to be 100% which was in our opinion not full proof. So, we performed Gaussian Regression, Decision Tree and KNN Model whose accuracies were --, 75.09% and 94.47%.

FUTURE SCOPE

In the future, there might be certain additions to the parameters and therefore the dataset that we have used, this will help to improve the accuracy of the model. This project will help the coffee bars to adjust their supply and process of preparing coffee leading to happier and more satisfied customers.