Logo, company name

Description automatically generated

AI/DATA SCIENCE ASSESSMENT

23-09-2022

By:

Derraz Radhwane

|  |
| --- |
| ACE Digital, Ace Resource Advisory Services Sdn Bhd.  Level 2, Wisma Averis (Tower 2), Avenue 5, Bangsar South City, No. 8 Jalan Kerinchi, 59200 Kuala Lumpur.  V: 86002, 7707  T: 03-27858888 (ext. 7707)  M: +601133034625  E: rafi\_shaik@aceresource.biz  W: [www.aceresource.biz](http://www.aceresource.biz) |

Analytics Evaluation Instructions:

1. Q1 & Q2 are mandatory, Q3 is bonus if attempted.
2. Do the analytics in cloud where possible (free trial versions).
3. Other methods are also acceptable.
4. Do include documentations.

# . The first task

A customer informed their consultant that they have developed several formulations of petrol  
that gives different characteristics of burning pattern. The formulations are obtaining by adding  
varying levels of additives that, for example, prevent engine knocking, gum prevention, stability  
in storage, and etc. However, a third-party certification organisation would like to verify if the  
formulations are significantly different, and request for both physical and statistical proof. Since  
the formulations are confidential information, they are not named in the dataset.  
Please assist the consultant in the area of statistical analysis by doing this.

## A descriptive analysis of the additives (columns named as “a” to “i”), which must include summaries of findings (parametric/non-parametric). Correlation and ANOVA, if applicable, is a must.

The steps to answer this question are ordered as follows:

### Read the ingredient data frame

The pandas (v 1.3.5) library was used to read the data frame. The attribute head() allows to show the first five rows in the data frame.

import pandas as pd

df=pd.read\_csv('/content/ingredient.csv')

df.head().round(3)

Output:

**Table. 1** The first five rows of the ingredient data frame.

a b c d e f g h i

0 1.517 13.02 3.54 1.69 72.73 0.54 8.44 0.00 0.07

1 1.531 10.73 0.00 2.10 69.81 0.58 13.30 3.15 0.28

2 1.523 13.31 3.58 0.82 71.99 0.12 10.17 0.00 0.03

3 1.518 12.56 3.52 1.43 73.15 0.57 8.54 0.00 0.00

4 1.518 13.43 3.98 1.18 72.49 0.58 8.15 0.00 0.00

### Check the data frame shape (size)

The pandas (1.3.5) library was used to read the data frame. The attribute shape allows to show data frame size. The ingredients data frame contains nine features (additives) and 214 samples.

import pandas as pd

df=pd.read\_csv('/content/ingredient.csv')

df.shape

Output:

(214, 9)

### Check if the data frame contains NANs

The pandas (1.3.5) library was used to read the data frame. The function isna() allows to check if the data frame contains any missing values (NAN). The function sum() allows to sum the missing values in each feature (additive). The ingredients data frame doesn’t contain any missing values.

import pandas as pd

df=pd.read\_csv('/content/ingredient.csv')

print(df.isna().sum())

Output:

a    0

b    0

c    0

d    0

e    0

f    0

g    0

h    0

i    0

dtype: int64

### Check if the data frame contains any infs

The pandas (1.3.5) library was used to read the data frame. The function isin() allows to check if the data frame contains any values with infinity of numbers (inf). The function sum() allows to sum the inf values in each feature (additive). The NumPy library (1.21.6) was used to determine the range of the isin() function by giving a positive and negative infinities. The ingredients data frame doesn’t contain any values with infinity of numbers (inf).

import pandas as pd

import pandas as np

df=pd.read\_csv('/content/ingredient.csv')

print(df.isin([np.inf, -np.inf]).sum())

Output:

a    0

b    0

c    0

d    0

e    0

f    0

g    0

h    0

i    0

dtype: int64

### Extract the data frame descriptive statisticsd

The pandas (1.3.5) library was used to read the data frame. The attribute describe() allows to show data frame descriptive statistics. The ingredients data frame descriptive statistics are shown in output table below.

import pandas as pd

df=pd.read\_csv('/content/ingredient.csv')

df.describe().round(3)

Output:

**Table. 2** The descriptive statistics of the ingredient data frame.

a b c d e f g h i

count 214.00 214.00 214.00 214.00 214.00 214.00 214.00 214.00 214.00

mean 1.52 13.41 2.68 1.44 72.65 0.50 8.96 0.18 0.06

std 0.00 0.82 1.44 0.50 0.77 0.65 1.42 0.50 0.10

min 1.51 10.73 0.00 0.29 69.81 0.00 5.43 0.00 0.00

25% 1.52 12.91 2.11 1.19 72.28 0.12 8.24 0.00 0.00

50% 1.52 13.30 3.48 1.36 72.79 0.56 8.60 0.00 0.00

75% 1.52 13.82 3.60 1.63 73.09 0.61 9.17 0.00 0.10

max 1.53 17.38 4.49 3.50 75.41 6.21 16.19 3.15 0.51

### Check the data frame additives correlation

The Pandas library (1.3.5) was used to read the data and extract the data correlation. The seaborn library (0.11.2) was used to the show the correlation heatmap. Given the additives anomaly distributions, the non-parametric spearman’s r rank coefficient was used to calculate the additives correlation. The correlation heatmap is illustrated in the figure below (Figure. 1).

A screenshot of a game

Description automatically generated with low confidence

**Figure. 1** The correlation heatmap of the ingredients data frame additives.

Overall, the correlation between the additives is very low except the a and g additives. The additives high noise is expected to weaken their correlation. Thus, a data denoising is required.

The code to perform this section is written below.

import pandas as pd

import seaborn as sns

df=pd.read\_csv('/content/ingredient.csv')

cor=df.corr(method='spearman').round(2)

p1 = sns.heatmap(cor, annot=True)

p1

### Check the data signal-to-noise ratio (SNR)

The pandas (1.3.5) library was used to read the data frame. The NumPy library (1.21.6) was used to calculate the SNR of the additives. The function signaltonoise () was customised and used to check the data frame additives SNR. The ingredient data frame additives’ SNR values are shown in the table below. The function python code is shown below. The additives with big SNR values are additives with low noise and the additives with low SNR values are additives with high noise. The SNR is measured in decibels (dB).

The additive f recorded the lowest SNR value (SNR=4.68 dB) indicating that f is highly noisy. The data noise may affect the accuracy of the parametric/non-parametric analysis. Therefore, the additives will be denoised for better analysis.

import pandas as pd

import numpy as np

df=pd.read\_csv('/content/ingredient.csv')

def signaltonoise(a, axis=0, ddof=0):

    a = np.asanyarray(a)

    m = a.mean(axis)

    sd = a.std(axis=axis, ddof=ddof)

    return abs(20\*np.log10(abs(np.where(sd == 0, 0, m\*\*2/sd\*\*2))))

snr=signaltonoise(np.array(df),0,0)

df4=pd.DataFrame(snr)

df4.set\_axis(df.columns,axis=0,inplace=True)

df4.set\_axis(['SNR'],axis=1,inplace=True)

df4

Output:

**Table. 3** The SNR of the ingredient data frame additives.

|  |  |
| --- | --- |
| Additives | SNR |
| a | 108.00 |
| b | 48.65 |
| c | 10.83 |
| d | 18.50 |
| e | 78.93 |
| f | 4.68 |
| g | 32.00 |
| h | 18.10 |
| i | 9.27 |

### The data denoising

The KalmanFilter module from the pykalman library (0.9.5) was used to denoise the additives. The code to perform this section is write below. The correlation heatmap after the additives denoising is illustrated in Figure. 2.

from pykalman import KalmanFilter

import itertools

df=pd.read\_csv('/content/ingredient.csv')

def KFSM(input):

    X=input.transpose().values.tolist()

    kf=lambda signal: KalmanFilter().em(signal, n\_iter=7).filter(signal)[0].tolist()

    array=np.array([list(itertools.chain(\*kf(signal))) for signal in X])

    return array

KFSM(df).shape

df1=pd.DataFrame(KFSM(df).transpose())

df1.columns=df.columns

df1.to\_excel('ingredient\_KF-denoised.xlsx',index=False)

### Check the data frame additives correlation after data denoising

The Pandas library (1.3.5) was used to read the data and extract the data correlation. The seaborn library (0.11.2) was used to the show the correlation heatmap. Given the additives anomaly distributions, the non-parametric spearman’s r rank coefficient was used to calculate the additives correlation. The correlation heatmap is illustrated in the figure below (Figure. 2).

Graphical user interface, PowerPoint

Description automatically generated with medium confidence

**Figure. 2** The correlation heatmap of the ingredients data frame additives after data denoising.

Overall, the correlation between the additives is very low except the a and g additives. However, the correlation has been increased between the additives after data denoising.

The code to perform this section is written below.

import pandas as pd

import seaborn as sns

df=pd.read\_excel('/content/ingredient\_KF-denoised.xlsx')

cor=df.corr(method='spearman').round(2)

p1 = sns.heatmap(cor, annot=True)

p1

## A graphical analysis of the additives, including a distribution study.

### The additive curves and their curves’ distributions

The additives’ curves and their curves’ distributions are illustrated in Figure 3. The pandas (1.3.5) library was used to read the data frame. The NumPy library (1.21.6) was used to calculate the minimum and maximum of the additives. The Pyplot module from the matplotlib library (3.2.2) was used to plot the curves and curves’ distributions figures. The stats.norm() module from the SciPy library (1.7.3) was used to calculate the normal distribution (blue line in Figure 3-a) to which the additives distribution are compared to. The Python compiler (3.7.14) was used to write the codes using the Colab as an integrated development environment (IDE).

Graphical user interface, website

Description automatically generatedText, whiteboard

Description automatically generated**Figure. 3** The additives curves and curves distribution. The subplots a-i in plot a show the distribution of the a-i additives from the ingredients data frame. The subplots a-i in plot b show the curves of the a-i additives from the ingredients data frame.

The additives distributions are illustrated in Figure. 3-a. The blue curves denote the normal distribution. The orange bars denote the additive actual distribution. Both are illustrated to compare them in terms of the data anomaly. The additives b and d almost have a normal distribution. The other additives show either right or left skewed distributions. Thus, the other additives are anomaly distributed. The additives curves after data denoising are illustrated in Figure. 3-b. The samples of the additives a and b are almost centred around their means, which justify why the additives a and b are normally distributed.

The code to implement the curves plotting is written below:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from scipy import stats

df=pd.read\_excel('/content/ingredient\_KF-denoised.xlsx')

fig, axs = plt.subplots(len(df.columns), figsize=(5,15))

fig.suptitle('(a)                The data frame anomaly')

cols=df.columns

for ax,col in zip(axs,cols):

        d1=df[col]

        x=np.linspace(d1.min(), d1.max(),d1.shape[0])

        y=stats.norm.pdf(x, np.mean(d1), np.std(d1))

        ax.plot(x,y, label='Normal Dist.', lw=3)

        ax.set\_title(f'{d1.name}')

        ax.hist(d1,density=True,  bins=30, label='Actual Data')

        ax.legend()

        ax.set\_xlabel('Values', size=12)

        ax.set\_ylabel('Count')

The code to implement the curves plotting is written below:

import pandas as pd

import numpy as np

df=pd.read\_excel('/content/ingredient\_KF-denoised.xlsx')

fig, axs = plt.subplots(len(df.columns), figsize=(5,15))

fig.suptitle('(b)                The data frame curves')

cols=df.columns

for ax,col in zip(axs,cols):

        d1=df[col]

        ax.set\_title(f'{d1.name}')

        ax.plot(d1)

        ax.legend()

        ax.set\_xlabel(f'{d1.name}', size=12)

        ax.set\_ylabel('Values')

## A clustering test of your choice (unsupervised learning), to determine the distinctive number of formulations present in the dataset. (Refer attachment: ingredients.csv)

K-means is an unsupervised clustering algorithm designed to partition unlabelled data into a certain number (that’s the “K”) of distinct groupings. In other words, k-means finds observations that share important characteristics and classifies them together into clusters. The Pandas library (v 1.4.0) was used to read the ingredient data frame. The KMeans algorithm from the cluster module in the scikit learn (v 1.1.2) was used to cluster the ingredient data frame. The integrated development environment (IDE) Colab was used for the coding using Python (Python 3.8.5). The code for this process is written bellow. The number of cluster (n\_clusters) was set to 4, the ‘k-means++’ was chosen to be the initialisation method. This technique selects initial cluster centroids using sampling based on an empirical probability distribution of the points’ contribution to the overall inertia. The maximum number of iteration (max\_iter) was set experimentally to 300. Number of time the k-means algorithm will be run with different centroid seeds (n\_init) was set to 10. The clustering algorithm was set to ‘auto’. The kmeans.labels\_ attribute return the number of clusters that the data is classified into.

import pandas as pd

from sklearn.cluster import KMeans

df=pd.read\_excel('/content/ingredient\_KF-denoised.xlsx')

print(df.shape)

kmeans = KMeans(n\_clusters=4, init='k-means++', algorithm='auto',

                max\_iter=300, n\_init=10, random\_state=00).fit(df)

print(kmeans.feature\_names\_in\_)

kmeans.labels\_

The output:

(214, 9)

['a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i']

array([3, 3, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2,

       0, 0, 0, 2, 2, 2, 0, 1, 1, 1, 0, 0, 2, 2, 2, 2, 2, 2, 2, 0, 1, 1,

       1, 1, 2, 2, 0, 0, 0, 2, 2, 2, 2, 2, 2, 0, 0, 2, 2, 2, 2, 2, 2, 2,

       0, 2, 2, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 0,

       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 1, 1, 0, 0, 2, 2, 2, 2, 2,

       2, 2, 2, 2, 0, 0, 0, 2, 2, 0, 2, 1, 1, 1, 2, 2, 1, 2, 2, 2, 2, 2,

       2, 0, 0, 0, 0, 0, 2, 2, 2, 0, 2, 2, 1, 1, 1, 1, 0, 2, 1, 2, 2, 2,

       2, 2, 1, 2, 2, 0, 2, 2, 2, 2, 0, 0, 0, 2, 2, 1, 2, 0, 0, 2, 2, 2,

       2, 2, 1, 1, 1, 0, 0, 0, 2, 0, 0, 0, 2, 2, 2, 2, 0, 2, 0, 0, 0, 1,

       2, 2, 0, 1, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2], dtype=int32)

Even though the number of clusters was set to 4 at the beginning, the KMeans divided the additives’ samples into 3 clusters. The KMeans classified the additives’ samples into a 3 number of clusters indicating that there are 3 distinctive number of formulations present in the dataset.

# . The second question

A team of plantation planners are concerned about the yield of oil palm trees, which seems to fluctuate. They have collected a set of data and needed help in analysing on how external factors influence fresh fruit bunch (FFB) yield. Some experts are of opinion that the flowering of oil palm tree determines the FFB yield and are linked to the external factors. Perform the analysis, which requires some study on the background of oil palm tree physiology.

*Note: (refer attachment palm\_ffb.csv).*

## Correlation analysis

The spearman’s r rank coefficient was used to calculate the correlation between the palm-FFB data frame features. The correlation heatmap is illustrated in Figure.4. The harvested area (HA\_Harvested) achieved the highest correlation with the fresh fruit bunch yield (ffb-yield) with an r of -0.39 indicating that the decrease in the harvested area led to an increase in the ffb-yield. The precipitation (Precipitation) followed the harvested area with an r of 0.31 indicating that the ffb-yield and the precipitation are positively increasing. The working days (Working\_days), minimum temperature (Min\_Temp), followed with an r values of 0.10 and 0.08, showing a very weak correlation with ffb-yield. The soil moisture (SoilMoisture), maximum temperature (Max\_Temp), and average temperature (Average\_Temp) were weakly and negatively correlated with the ffb-yield, achieving an r values of -0.05, -0.11, and -0.04, respectively.

The Pandas library (v 1.4.0) were used for the r application using Python (3.7.14). The seaborn (v 0.12.0) library was used to plot the correlation heatmap. The integrated development environment (IDE) Colab was used for the coding. The code for this process is written bellow.

import pandas as pd

import seaborn as sns

df=pd.read\_csv('/content/palm\_ffb.csv')

cor=df.corr(method='spearman').round(2)

p1 = sns.heatmap(cor, annot=True)

p1

Graphical user interface, application

Description automatically generated

**Figure. 4** The correlation heatmap of the oil palm data frame.

## Feature selection

The sequential feature selector algorithm (SFS) was used for the forward feature selection. The feature\_selection (v 0.21.0) module from the mlxtend (v 0.20.0) library was used to import and apply the SFS using Python (3.7.14). The integrated development environment (IDE) Colab was used for the coding. The code for this process is written bellow.

import pandas as pd

from sklearn.linear\_model import LinearRegression

sys.modules['sklearn.externals.joblib'] = joblib

from mlxtend.feature\_selection import SequentialFeatureSelector as SFS

df=pd.read\_csv('/content/palm\_ffb.csv')

X=df.iloc[:,1:-1]

y=df['FFB\_Yield']

linr=LinearRegression()

sfs1 = SFS(linr,

           k\_features=7,

           forward=True,

           floating=False,

           verbose=0,

           scoring='r2',

           cv=0)

sfs1 = sfs1.fit(X, y)

sfs1.subsets\_

The SFS selected the highly contributing features to predict the ff-yield in a sequential additive manner. The coefficient of determination R2 and the cross-validation score (cv-score) were used for the feature selection process. The linear regression algorithm (LR) was used as the model for the feature selection. The maximum number of features to be selected was set to seven, which is the number of independent features. The feature selection R2 and cv-score values are shown in Table.4.

**Table. 4** Summary of the feature selection subsets and subsets R2 scores.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model index | Feature index | Cv\_scores | Average score | Feature names |
| 1 | 6 | 0.123 | 0.123 | 1. HA\_Harvested |
| 2 | 4,6 | 0.164 | 0.164 | 1. Precipitation 2. HA\_Harvested |
| 3 | 0,4,6 | 0.235 | 0.235 | 1. SoilMoisture, 2. Precipitation 3. HA\_Harvested |
| 4 | 0,1,4,6 | 0.247 | 0.247 | 1. SoilMoisture, 2. Average\_Temp 3. Precipitation 4. HA\_Harvested |
| 5 | 0,1,4,5,6 | 0.251 | 0.251 | 1. SoilMoisture 2. Average\_Temp 3. Precipitation 4. Working\_days 5. HA\_Harvested |
| 6 | 0,1,2,4,5,6 | 0.253 | 0.253 | 1. SoilMoisture 2. Average\_Temp 3. Min\_Temp 4. Precipitation 5. Working\_days 6. HA\_Harvested |
| 7 | 0,1,2,3,4,5,6 | 0.254 | 0.254 | 1. SoilMoisture 2. Average\_Temp 3. Min\_Temp 4. Max\_Temp 5. Precipitation 6. Working\_days 7. HA\_Harvested |

## FFB yield prediction

The Python (3.7.14) was used for the histogram gradient boosting regressor (HGBR) application. The HGBR was applied to predict the ffb-yield based on the highly contributing external factors’ subsets for the ffb-yield prediction. The integrated development environment (IDE) Colab was used for the coding. The code for this process is written bellow. First pandas library (v 1.4.0) was used to read the ffb-yield data frame. The scikit learn library (v 0.20) was used to import metrics, model\_selection, processing, experimental, and ensemble modules. The metrics module was used to apply the R2, RMSE, and MAPE scores. The processing module was used to scale the data, ensemble module was used to call the HGBR algorithm. The module experimental was used to enable the HGBR. The model\_selection module was used to split the data to training and test subsets for the HGBR implementation. The HGBR implementation process is coded and written bellow. The F-test was calculated from a customised function as written in the code bellow. The F-test shows the model confidence. In other words, the F-test show how much the ffb-yield prediction model is trusted. The F-test values less than 1.57 at significance level of 0.05 indicates that the ffb-yield prediction model is trusted.

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import r2\_score

from sklearn.metrics import mean\_absolute\_percentage\_error

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import mean\_squared\_log\_error

from sklearn.preprocessing import MinMaxScaler

from sklearn.experimental import enable\_hist\_gradient\_boosting

from sklearn.ensemble import HistGradientBoostingRegressor

import warnings

warnings.simplefilter('ignore')

# load the data

df=pd.read\_csv('/content/palm\_ffb.csv')

cols=df.columns

# split the data

x=pd.concat([df[cols[1+0]],

             df[cols[1+1]],

             df[cols[1+2]],

             df[cols[1+3]],

             df[cols[1+4]],

             df[cols[1+5]],

             df[cols[1+6]]], axis=1)

y=df['FFB\_Yield']

print('x col:', x.columns)

print('y col:',y.name)

seed=list(np.arange(0,100,1))

for i in seed:

    Xt,Xs,Yt,Ys=train\_test\_split(x,y,test\_size=0.3,random\_state=i)

    scaler = MinMaxScaler()

    Xtt=scaler.fit\_transform(Xt)

    Xst=scaler.fit\_transform(Xs)

    rf=RandomForestRegressor()

    hgbr=HistGradientBoostingRegressor(tol=0.6000000000000001,

                                       min\_samples\_leaf=3,

                                       max\_leaf\_nodes=90,

                                       max\_iter=80,

                                       max\_depth=60,

                                       learning\_rate=0.1,

                                       l2\_regularization=0.9)

    model=hgbr

    model.fit(Xtt,Yt)

    # Training results

    pred=model.predict(Xtt)

    r21=r2\_score(Yt, pred)

    mape1=mean\_absolute\_percentage\_error(Yt, pred)

    rmse1=np.sqrt(mean\_squared\_error(Yt, pred))

    # Test results

    preds=model.predict(Xst)

    r22=r2\_score(Ys, preds)

    mape2=mean\_absolute\_percentage\_error(Ys, preds)

    rmse2=np.sqrt(mean\_squared\_error(Ys, preds))

    ft=[np.var(preds),np.var(Ys)]

    fts=np.max(ft)/np.min(ft)

    if r22 >= 0.475 and r21 >=0.475 and r21>r22:

        print(i)

        l1=[r21,r22]

        l2=[rmse1,rmse2]

        l3=[mape1,mape2]

        l6=[fts]

        l7=['r2','RMSE','MAPE','F-test']

        df3=pd.DataFrame([l1,l2,l3,l6]).transpose()

        df3.columns=l7

        display(df3)

        break

**Table. 5** Summary of the HGBR subsets R2, RMSE, MAPE, and F-test scores.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model features | The training stage | | | The test stage | | | F-test |
| R2 | RMSE | MAPE | R2 | RMSE | MAPE |
| 1. HA\_Harvested | 0.814 | 0.122 | 0.058 | 0.204 | 0.229 | 0.126 | 3.110 |
| 1. Precipitation 2. HA\_Harvested | 0.990 | 0.028 | 0.013 | 0.307 | 0.236 | 0.111 | 1.790 |
| 1. SoilMoisture, 2. Precipitation 3. HA\_Harvested | 0.997 | 0.015 | 0.006 | 0.473 | 0.184 | 0.101 | 1.153 |
| 1. SoilMoisture, 2. Average\_Temp 3. Precipitation 4. HA\_Harvested | 0.999 | 0.009 | 0.004 | 0.475 | 0.222 | 0.108 | 2.051 |
| 1. SoilMoisture 2. Average\_Temp 3. Precipitation 4. Working\_days 5. HA\_Harvested | 0.999 | 0.009 | 0.004 | 0.475 | 0.222 | 0.108 | 2.051 |
| 1. SoilMoisture 2. Average\_Temp 3. Min\_Temp 4. Precipitation 5. Working\_days 6. HA\_Harvested | 0.999 | 0.009 | 0.004 | 0.475 | 0.222 | 0.108 | 2.051 |
| 1. SoilMoisture 2. Average\_Temp 3. Min\_Temp 4. Max\_Temp 5. Precipitation 6. Working\_days 7. HA\_Harvested | 0.999 | 0.009 | 0.004 | 0.475 | 0.222 | 0.108 | 2.051 |

## Analysis and discussion

According to the results of R2, RMSE, and MAPE (Table.5), also considering the R2 and cv-score results (Table.5, the harvested area (external factor) model was the highly contributing model for predicting the ffb-yield, achieving an R2 of 0.204, RMSE of 0.229, and MAPE of 0.126. The second contributing model was the (harvested area + precipitation) achieving an R2, RMSE, and MAPE values of 0.307, 0.236, 0.111. The (harvested area + precipitation+ soil moisture) improved the ff-yield predictability by achieving an R2, RMSE, and MAPE values of 0.473, 0.184, and 0.101, respectively. Furthermore, the (harvested area + precipitation+ soil moisture + average temperature) improved the ff-yield predictability by achieving an R2, RMSE, and MAPE values of 0.475, 0.222, and 0.108, respectively. However, the addition of the rest of features didn’t improve the ffb-yield predictability. The (harvested area + precipitation+ soil moisture) was the only model to be trusted for the ffb-yield prediction by achieving an F-test value (F-test=1.153) less than 1.57. The can conclude that the harvested area, precipitation, and soil moisture are the external factors to affect the ffb-yield production.

The influence of a set of external factors on the interannual variability of oil palm fresh fruit bunches (FFB) total yields in Malaysia was explored in a previous study. As a result of thi study, the yearly production of FFB becomes lower than that of a normal year since the water stress during the boreal spring has an important impact on the total annual yields of FFB. Conversely, La Niña sets favourable conditions for palm trees to produce more FFB by reducing chances of water stress risk (Oettli et al., 2018).

# The third task

As a term, data analytics predominantly refers to an assortment of applications, from basic business  
intelligence (BI), reporting and online analytical processing (OLAP) to various forms of advanced  
analytics. In that sense, it's similar in nature to business analytics, another umbrella term for  
approaches to analyzing data -- with the difference that the latter is oriented to business uses, while  
data analytics has a broader focus. The expansive view of the term isn't universal, though: In some  
cases, people use data analytics specifically to mean advanced analytics, treating BI as a separate  
category. Data analytics initiatives can help businesses increase revenues, improve operational  
efficiency, optimize marketing campaigns and customer service efforts, respond more quickly to  
emerging market trends and gain a competitive edge over rivals -- all with the ultimate goal of  
boosting business performance. Depending on the particular application, the data that's analysed  
can consist of either historical records or new information that has been processed for real-time  
analytics uses. In addition, it can come from a mix of internal systems and external data sources. At  
a high level, data analytics methodologies include exploratory data analysis (EDA), which aims to find patterns and relationships in data, and confirmatory data analysis (CDA), which applies statistical  
techniques to determine whether hypotheses about a data set are true or false. EDA is often  
compared to detective work, while CDA is akin to the work of a judge or jury during a court trial -- a  
distinction first drawn by statistician John W. Tukey in his 1977 book Exploratory Data Analysis. Data analytics can also be separated into quantitative data analysis and qualitative data analysis. The  
former involves analysis of numerical data with quantifiable variables that can be compared or  
measured statistically. The qualitative approach is more interpretive -- it focuses on understanding  
the content of non-numerical data like text, images, audio and video, including common phrases,  
themes and points of view.

Feed the following paragraph into your favourite data analytics tool, and answer the following:

## What is the probability of the word “data” occurring in each line?

The probability mass function (PMF) was used to calculate the occurrence probability of the word data at each line. The PMF results are shown in Table. 6. The open() function was used to read the text file. Then after, the natural language toolkit (NLTK) library (3.7) was used to mine the text. The FreqDist() module was used to count of each single word at each line the PMF was calculated for each word based on it count and the total number of words at each line. The Pandas library (1.3.5) was used to build the results data frame (create the results tables). The codes are written below. The count, probability, and the word name, such as data, are extracted from the resulting data frames.

**Table. 6** Summary of the count and probability mass function of the word data at each line of the text.

|  |  |  |
| --- | --- | --- |
| Line index | Count | Probability |
| 0 | 1 | 0.059 |
| 1 | 0 | 0.000 |
| 2 | 0 | 0.000 |
| 3 | 1 | 0.055 |
| 4 | 1 | 0.047 |
| 5 | 1 | 0.058 |
| 6 | 1 | 0.076 |
| 7 | 0 | 0.000 |
| 8 | 0 | 0.000 |
| 9 | 0 | 0.000 |
| 10 | 0 | 0.000 |
| 11 | 1 | 0.047 |
| 12 | 1 | 0.111 |
| 13 | 1 | 0.058 |
| 14 | 0 | 0.000 |
| 15 | 2 | 0.062 |
| 16 | 1 | 0.071 |
| 17 | 0 | 0.000 |
| 18 | 1 | 0.055 |
| 19 | 0 | 0.000 |

import nltk

import nltk.corpus

from nltk.tokenize import word\_tokenize

from nltk.probability import FreqDist

import nltk

nltk.download('punkt')

with open('/content/Text.txt') as f:

    lines = f.readlines()

    for line in lines:

        # print the line index

        idx=lines.index(line)

        print(idx)

        # print the current line

        print(lines[idx])

        token = word\_tokenize(line)

        fdist = FreqDist(token)

        # print the current line words count

        key=fdist.keys()

        keyl=list(key)

        print(key)

        # print the current line words probability

        val=fdist.values()

        print(list(val))

        tot=sum(val)

        p= [i/tot for i in list(val)]

        df1=pd.DataFrame([keyl,list(val), p ])

        df1.set\_axis(['word', 'count','prob'], axis=0, inplace=True)

        display(df1.transpose().round(2))

## What is the distribution of distinct word counts across all the lines?

The distribution of distinct word counts across all the lines is shown in Table. 7. The same libraries are used for this task. The code is written below.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import nltk

import nltk.corpus

from nltk.tokenize import word\_tokenize

from nltk.probability import FreqDist

import nltk

nltk.download('punkt')

f = open('/content/Text.txt','r')

X=f.read()

token = word\_tokenize(X)

fdist = FreqDist(token)

keys=fdist.keys()

vals=fdist.values()

p= [i/len(keys) for i in list(vals)]

df1=pd.DataFrame([keys,list(vals), p ])

df1.set\_axis(keys, axis=1, inplace=True)

df1.set\_axis(['word', 'count','prob'], axis=0, inplace=True)

display(df1.transpose().round(2))

df1.to\_excel('disnr.xlsx')

**Table. 6** Summary of the count and probability mass function of the word data at each line of the text.

|  |  |  |  |
| --- | --- | --- | --- |
| **Word index** | **Word** | **Count** | **PMF** |
| **0** | As | 1 | 0.005 |
| **1** | a | 10 | 0.049 |
| **2** | term | 3 | 0.015 |
| **3** | , | 23 | 0.113 |
| **4** | data | 15 | 0.074 |
| **5** | analytics | 10 | 0.049 |
| **6** | predominantly | 1 | 0.005 |
| **7** | refers | 1 | 0.005 |
| **8** | to | 11 | 0.054 |
| **9** | an | 1 | 0.005 |
| **10** | assortment | 1 | 0.005 |
| **11** | of | 10 | 0.049 |
| **12** | applications | 1 | 0.005 |
| **13** | from | 2 | 0.010 |
| **14** | basic | 1 | 0.005 |
| **15** | business | 4 | 0.020 |
| **16** | intelligence | 1 | 0.005 |
| **17** | ( | 4 | 0.020 |
| **18** | BI | 2 | 0.010 |
| **19** | ) | 4 | 0.020 |
| **20** | reporting | 1 | 0.005 |
| **21** | and | 9 | 0.044 |
| **22** | online | 1 | 0.005 |
| **23** | analytical | 1 | 0.005 |
| **24** | processing | 1 | 0.005 |
| **25** | OLAP | 1 | 0.005 |
| **26** | various | 1 | 0.005 |
| **27** | forms | 1 | 0.005 |
| **28** | advanced | 2 | 0.010 |
| **29** | . | 11 | 0.054 |
| **30** | In | 3 | 0.015 |
| **31** | that | 5 | 0.025 |
| **32** | sense | 1 | 0.005 |
| **33** | it | 3 | 0.015 |
| **34** | 's | 2 | 0.010 |
| **35** | similar | 1 | 0.005 |
| **36** | in | 3 | 0.015 |
| **37** | nature | 1 | 0.005 |
| **38** | another | 1 | 0.005 |
| **39** | umbrella | 1 | 0.005 |
| **40** | for | 2 | 0.010 |
| **41** | approaches | 1 | 0.005 |
| **42** | analyzing | 1 | 0.005 |
| **43** | -- | 4 | 0.020 |
| **44** | with | 3 | 0.015 |
| **45** | the | 8 | 0.039 |
| **46** | difference | 1 | 0.005 |
| **47** | latter | 1 | 0.005 |
| **48** | is | 5 | 0.025 |
| **49** | oriented | 1 | 0.005 |
| **50** | uses | 2 | 0.010 |
| **51** | while | 2 | 0.010 |
| **52** | has | 2 | 0.010 |
| **53** | broader | 1 | 0.005 |
| **54** | focus | 1 | 0.005 |
| **55** | The | 3 | 0.015 |
| **56** | expansive | 1 | 0.005 |
| **57** | view | 2 | 0.010 |
| **58** | n't | 1 | 0.005 |
| **59** | universal | 1 | 0.005 |
| **60** | though | 1 | 0.005 |
| **61** | : | 1 | 0.005 |
| **62** | some | 1 | 0.005 |
| **63** | cases | 1 | 0.005 |
| **64** | people | 1 | 0.005 |
| **65** | use | 1 | 0.005 |
| **66** | specifically | 1 | 0.005 |
| **67** | mean | 1 | 0.005 |
| **68** | treating | 1 | 0.005 |
| **69** | as | 1 | 0.005 |
| **70** | separate | 1 | 0.005 |
| **71** | category | 1 | 0.005 |
| **72** | Data | 3 | 0.015 |
| **73** | initiatives | 1 | 0.005 |
| **74** | can | 5 | 0.025 |
| **75** | help | 1 | 0.005 |
| **76** | businesses | 1 | 0.005 |
| **77** | increase | 1 | 0.005 |
| **78** | revenues | 1 | 0.005 |
| **79** | improve | 1 | 0.005 |
| **80** | operational | 1 | 0.005 |
| **81** | efficiency | 1 | 0.005 |
| **82** | optimize | 1 | 0.005 |
| **83** | marketing | 1 | 0.005 |
| **84** | campaigns | 1 | 0.005 |
| **85** | customer | 1 | 0.005 |
| **86** | service | 1 | 0.005 |
| **87** | efforts | 1 | 0.005 |
| **88** | respond | 1 | 0.005 |
| **89** | more | 2 | 0.010 |
| **90** | quickly | 1 | 0.005 |
| **91** | emerging | 1 | 0.005 |
| **92** | market | 1 | 0.005 |
| **93** | trends | 1 | 0.005 |
| **94** | gain | 1 | 0.005 |
| **95** | competitive | 1 | 0.005 |
| **96** | edge | 1 | 0.005 |
| **97** | over | 1 | 0.005 |
| **98** | rivals | 1 | 0.005 |
| **99** | all | 1 | 0.005 |
| **100** | ultimate | 1 | 0.005 |
| **101** | goal | 1 | 0.005 |
| **102** | boosting | 1 | 0.005 |
| **103** | performance | 1 | 0.005 |
| **104** | Depending | 1 | 0.005 |
| **105** | on | 2 | 0.010 |
| **106** | particular | 1 | 0.005 |
| **107** | application | 1 | 0.005 |
| **108** | analysed | 1 | 0.005 |
| **109** | consist | 1 | 0.005 |
| **110** | either | 1 | 0.005 |
| **111** | historical | 1 | 0.005 |
| **112** | records | 1 | 0.005 |
| **113** | or | 4 | 0.020 |
| **114** | new | 1 | 0.005 |
| **115** | information | 1 | 0.005 |
| **116** | been | 1 | 0.005 |
| **117** | processed | 1 | 0.005 |
| **118** | real-time | 1 | 0.005 |
| **119** | addition | 1 | 0.005 |
| **120** | come | 1 | 0.005 |
| **121** | mix | 1 | 0.005 |
| **122** | internal | 1 | 0.005 |
| **123** | systems | 1 | 0.005 |
| **124** | external | 1 | 0.005 |
| **125** | sources | 1 | 0.005 |
| **126** | At | 1 | 0.005 |
| **127** | high | 1 | 0.005 |
| **128** | level | 1 | 0.005 |
| **129** | methodologies | 1 | 0.005 |
| **130** | include | 1 | 0.005 |
| **131** | exploratory | 1 | 0.005 |
| **132** | analysis | 5 | 0.025 |
| **133** | EDA | 2 | 0.010 |
| **134** | which | 2 | 0.010 |
| **135** | aims | 1 | 0.005 |
| **136** | find | 1 | 0.005 |
| **137** | patterns | 1 | 0.005 |
| **138** | relationships | 1 | 0.005 |
| **139** | confirmatory | 1 | 0.005 |
| **140** | CDA | 2 | 0.010 |
| **141** | applies | 1 | 0.005 |
| **142** | statistical | 1 | 0.005 |
| **143** | techniques | 1 | 0.005 |
| **144** | determine | 1 | 0.005 |
| **145** | whether | 1 | 0.005 |
| **146** | hypotheses | 1 | 0.005 |
| **147** | about | 1 | 0.005 |
| **148** | set | 1 | 0.005 |
| **149** | are | 1 | 0.005 |
| **150** | true | 1 | 0.005 |
| **151** | false | 1 | 0.005 |
| **152** | often | 1 | 0.005 |
| **153** | compared | 2 | 0.010 |
| **154** | detective | 1 | 0.005 |
| **155** | work | 2 | 0.010 |
| **156** | akin | 1 | 0.005 |
| **157** | judge | 1 | 0.005 |
| **158** | jury | 1 | 0.005 |
| **159** | during | 1 | 0.005 |
| **160** | court | 1 | 0.005 |
| **161** | trial | 1 | 0.005 |
| **162** | distinction | 1 | 0.005 |
| **163** | first | 1 | 0.005 |
| **164** | drawn | 1 | 0.005 |
| **165** | by | 1 | 0.005 |
| **166** | statistician | 1 | 0.005 |
| **167** | John | 1 | 0.005 |
| **168** | W. | 1 | 0.005 |
| **169** | Tukey | 1 | 0.005 |
| **170** | his | 1 | 0.005 |
| **171** | 1977 | 1 | 0.005 |
| **172** | book | 1 | 0.005 |
| **173** | Exploratory | 1 | 0.005 |
| **174** | Analysis | 1 | 0.005 |
| **175** | also | 1 | 0.005 |
| **176** | be | 2 | 0.010 |
| **177** | separated | 1 | 0.005 |
| **178** | into | 1 | 0.005 |
| **179** | quantitative | 1 | 0.005 |
| **180** | qualitative | 2 | 0.010 |
| **181** | former | 1 | 0.005 |
| **182** | involves | 1 | 0.005 |
| **183** | numerical | 1 | 0.005 |
| **184** | quantifiable | 1 | 0.005 |
| **185** | variables | 1 | 0.005 |
| **186** | measured | 1 | 0.005 |
| **187** | statistically | 1 | 0.005 |
| **188** | approach | 1 | 0.005 |
| **189** | interpretive | 1 | 0.005 |
| **190** | focuses | 1 | 0.005 |
| **191** | understanding | 1 | 0.005 |
| **192** | content | 1 | 0.005 |
| **193** | non-numerical | 1 | 0.005 |
| **194** | like | 1 | 0.005 |
| **195** | text | 1 | 0.005 |
| **196** | images | 1 | 0.005 |
| **197** | audio | 1 | 0.005 |
| **198** | video | 1 | 0.005 |
| **199** | including | 1 | 0.005 |
| **200** | common | 1 | 0.005 |
| **201** | phrases | 1 | 0.005 |
| **202** | themes | 1 | 0.005 |
| **203** | points | 1 | 0.005 |

## What is the probability of the word “analytics” occurring after the word “data”?

The probability of the word “analytics” occurring after the word “data” is calculated below using the PMF function. The code is written below.

count=0

with open('/content/Text.txt') as f:

    lines = f.readlines()

    for line in lines:

        if 'data analytics' in line:

            true=1

            count+=true

    print(count)

    print(count/204)

Output:

4

0.0196078431372549

# References

Oettli, P., Behera, S.K., Yamagata, T., 2018. Climate Based Predictability of Oil Palm Tree Yield in Malaysia. Sci. Rep. 8, 1–13. https://doi.org/10.1038/s41598-018-20298-0

<https://pandas.pydata.org/>

<https://matplotlib.org/>

<https://www.python.org/>

<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

<http://rasbt.github.io/mlxtend/>

<https://seaborn.pydata.org/>

<https://scipy.org/>

<https://pykalman.github.io/>

<https://numpy.org/>

<https://scipy.org/>

<https://www.nltk.org/>