Introduction

This assignment is done to find the impact that Rainfall has on the price of food products. India is a location where both rainfall and price of food products drastically changes. India has the largest sector in terms of agriculture but is still largely dependent on rainfall for irrigation. Therefore India is the best suitable choice for this analysis, with its large dependency on rain along with its unpredictable and rapidly changing weather and fluctuation in the price of food items. The location of India was selected for finding the relation between rainfall and the price of food items. As the effect of rain on the food price does not take place immediately the time it takes for the crop to grow has to be taken into account. The crop that every Indian use and the one whose price changes very rapidly is Onion. Taking the Onion price and mean rainfall would give a rough estimate of the relation between food price and rain. Along with this analysis, the relation between the price of food products with long shelf life and rainfall will be evaluated.

Objective

The objective of this report is to find the statistical relation between the price of food items and mean rainfall. To do this a few suitable data set from a valid source was looked through. The data set will be cross-checked with different sources to see its validity. After making sure the data is valid basic visualization analysis will be done on the data set to find anomalies. After performing data cleaning and merging, the relevant statistical tests will be performed. Furthermore, to check the relation between food product price and different categories of rainfall, data wrangling will be performed. The effect rain category and mean rainfall has on the price of food will be tested using different statistical tools. All the tests will be done using linear and other statistical models. Analysis test will be performed on the data set from the year 2000 to 2017. After linear regression tests, a decision tree will be plotted to predict the cost of food products with respect to mean rainfall. A final analysis will be done to check for the strengths and limitations of my analysis.

Data

The data set for rainfall in India was found in the open government data platform. The data set contained the rainfall in millimetres of all regions of India. The amount of rainfall was divided into each month from the year 1901 to 2017. The data set for the price of food items of different years was found from the data worlds repository. This data set contained the price of food products from 2000 February to 2020 January. Another dataset from the research gate was used for the categorisation of rainfall into light, medium, heavy and very heavy rainfall. In order to perform the analysis data from the year, 2000 February to 2020 January is used. After a simple data wrangling and missing data analysis, the following tables were created indicating the presence of missing data. Table 1 shows the presence of missing data in the price of food products data set. The percentage level of missing data cannot be overlooked and has to be dealt with.

Table 1: missing values for each variable in the dataset food price

Variable	Onions	Mustard_oil	Date	Chickpeas	Potato	Rice	Sugar	Wheat
No of missing value	24	15	0	0	0	0	0	0
Percent missing	10	6.25	0	0	0	0	0	0

Analysing the dataset on rain shows the missing values represented in Table 2. The number of missing values is small but relevant. These missing values have to be handled

Table 2: missing values for each variable in the dataset food price

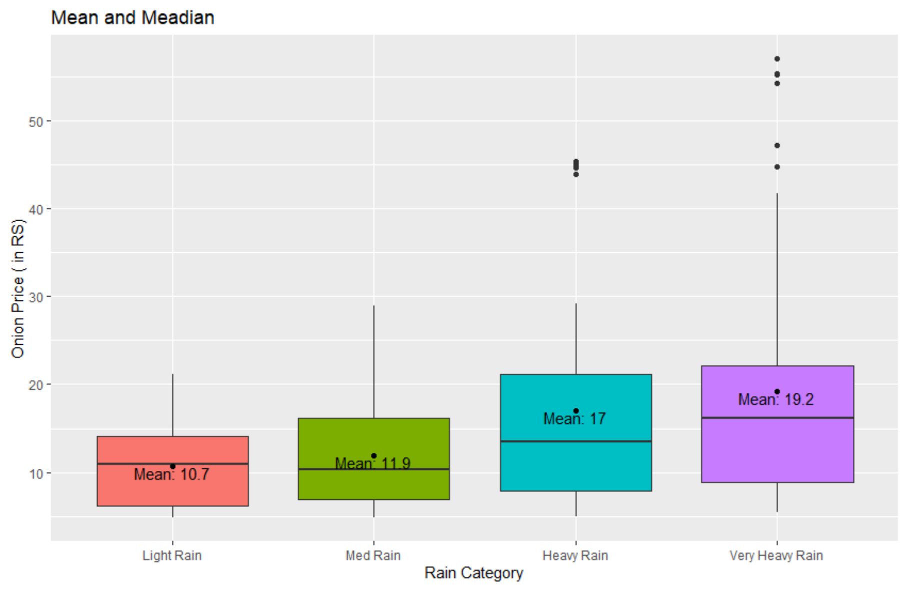
Variable	Region	YEAR	NOV	DEC	JUL	OCT	MAR	SEP	JUN	JAN	APR	AUG	FEB	MAY
No of miss	0	0	11	10	7	7	6	6	5	4	4	4	3	3
% Missing	0	0	0.26	0.23	0.16	0.16	0.14	0.14	0.11	0.09	0.09	0.09	0.07	0.07

Naniar package is used for creating these tables for representing the missing values. During the initial dataset analysis, each data set was searched to find if there are any hidden missing data called implicit data. To check this a tidyr command was run. Implicit missing data occurs when an entire row goes missing. Implicit missing data cannot be found using normal missing data visualisation or summarisation. Therefore, it goes unnoticed in many circumstances. There were chunks of implicit missing data in the rainfall data set. The rainfall dataset had data on rainfall from different regions of India. To get the mean rainfall of India for the different time period a new data set showing only the total mean rainfall in India was created. This was calculated using simple mean.

After the initial wrangling, I combined the two datasets to create a single dataset that has data from the years 2000 to 20017. A new column representing the different categories of rainfall based on mean rainfall in millimetres was created. Therefore, the final dataset had the price of food items, the mean rainfall and different categories of rain. One problem that has to be considered when analysing the rainfall and price of food products is the time it takes for the effect of rain to come into play. rainfall does not affect the price of food products immediately as crops take time to grow. This delay in time can be called as a lag time. To compute this lag while doing the analysis the dataset is mutated. This is achieved by creating a lag in the mean rainfall column of the dataset. The amount of lag depends on the crop being evaluated. In the case of onion, the time it takes for it to grow is 3 months therefore a lag of 3 months is put in the rainfall column by 3 months. This should account for some of the problem caused by the lag time.

Before dealing with the missing data a simple hypothesis test is done to see the relationship between the category of rain and the price of food products. For the test to take place we consider a null hypothesis (H_0), which is a statement prior to the experiment, that the mean of the price of onion is the same for all categories is considered. This in turn means that H_0 considers that there is no varying relation between rainfall category and price of onion. The alternative hypothesis(H_1) is that at least one of the categories has a different mean. A significance level of 5% is taken and when the value is lower than that the null hypothesis will be rejected. The significance level or alpha is also the probability of a type 1 error. To do this hypothesis test a one-way ANOVA test is used. ANOVA was chosen because it tests the relationship between a variable with categories or factors against a variable with numeric values. Before conducting a hypothesis test, A boxplot is plotted to visualise the relationship between each category and the rainfall.

Fig 1: Box plot with mean



Tab 3: Standard deviation

	Standard Deviation
Light Rain	4.71
Medium Rain	6.42
Heavy Rain	11.3
V Heavy Rain	13.1

From the boxplot there is a clear relationship between the category of rain and the price of Onion. The mean of each category is different and changes as the category of rainfall changes. From the boxplot and mean we may consider the possibility that as rainfall increases the price of onion increases, but looking at the standard deviation as the rainfall gets heavier the standard deviation also increases. Therefore, a proper test has to be performed to check the relation.

ANOVA test is run with the assumption that the null hypothesis is true. From the simple ANOVA test, we get table 4 as shown below. In the table below we see an F and P-value. F- value is a value that considers the variation from the overall mean. F-value accounts for variations that are expected and unexpected. To find the significance of these variations we are using the level of P-value and F-value we get from the test.

In a situation where the null hypothesis is true, we expect the F value to be close to 1. which indicate that unexpected variation and expected variation is close to each other. P-value gives the probability of the observed test statistic or the extreme if null is true. From the table, it is clear that the probability of getting an F-value greater than 8.5 when it should be 1 (ie null) is much lower than our significance level. The significance level we set was 0.05 and the P-value from the table is less than 0.05. Therefore we have reason to reject the null hypothesis. This means the mean price of each category does vary for at least one category of rain.

Table 4: Summary of ANOVA test

	df	Mean sq	F value	P-value
Rain category	3	870.7	8.583	2.13e-05 ***
Residuals	207	101.4		

An analysis of which category differs from another can be done by comparing all possible pairs of means. There are 6 possible pairs as shown in table 5. Multiple comparisons of each possible pair will increase the probability of getting at least one type 1 error. To avoid getting a high probability of getting at least one type 1 error we use a correction method. The correction method used here is the Bonferroni correction method. This adjusts the alpha or significance level which returns an adjusted p-value. We do this to reduce the number of false positives but this will make more false negatives.

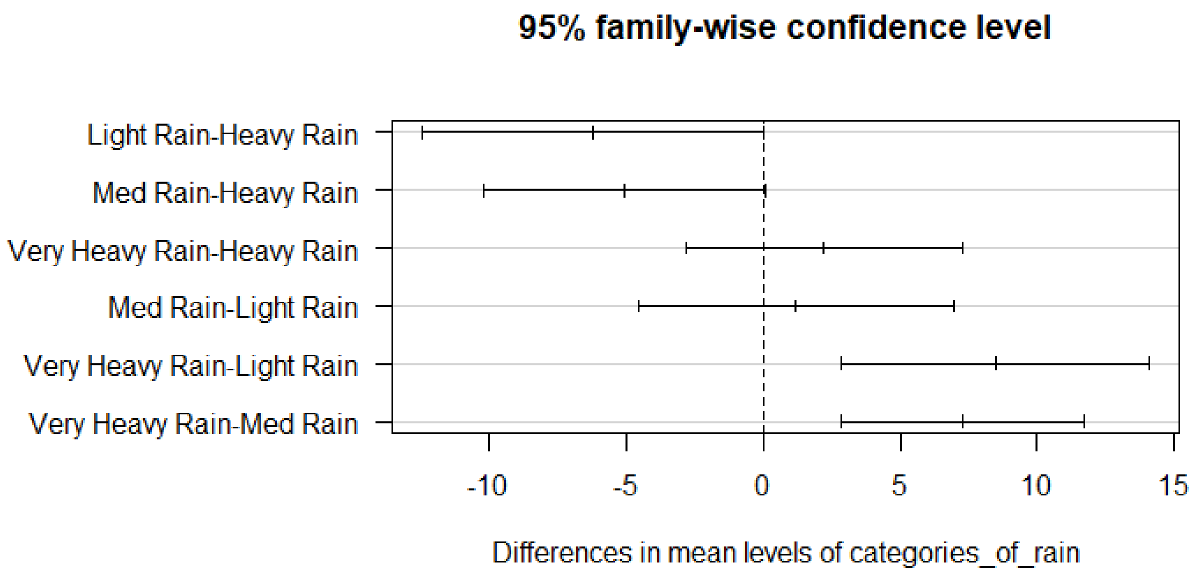
Table 5: Table shows the adjusted P value of all possible pairs

	Heavy Rain P-adj	Light Rain P-adj	Med Rain p-adj
Light Rain	0.06	-	-
Med Rain	0.07	1.0	-
Very Heavy Rain	1.0	0.00084	0.00019

From table 5 we can see that pairs of light rain and very heavy rain are statistically different as the adjusted p-value is lower than 0.05 which is the significance level. The pair med rain and very heavy rain is also significantly different all other pairs are not significantly different.

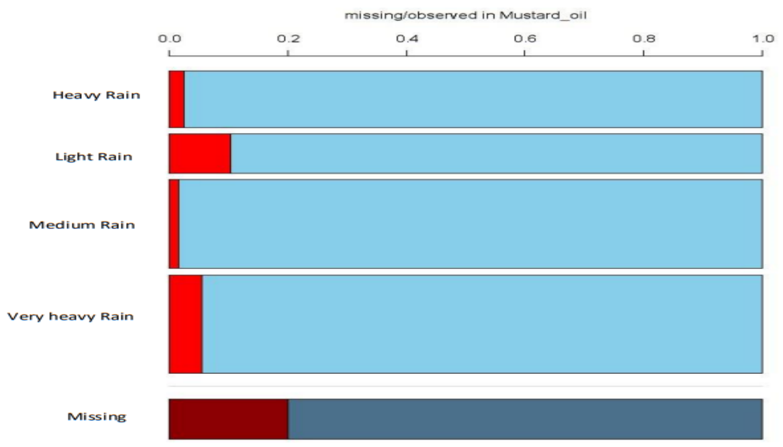
To get a visual representation of the difference between pairs, the TukeyHD correction method was used and the confidence intervals were plotted as in figure 2. It shows that Very Heavy Rain – Light Rain and Very Heavy Rain -Medium Rain looks significant. Also, these confidence intervals do not contain zero which shows a statistically significant difference between these two pairs of groups. No other significance is observed.

Fig 2: Confidence Interval of different pair of categories of rain



One of the problems we face with missing data is that when we fit a model to the missing dataset, we will lose a chunk of data. This happens because the programme deletes the entire row even though only a single data point is missing. To handle missing data, we first have to know what kind of missing data is present in our dataset. The type of missing data that can be handled with ease are data that are missing completely at random (MCAR) and data that are missing at random (MAR). If the data is MAR, the missingness can be dealt by imputing the dataset multiple time and filling the missing values. We do the imputation because we do not want to lose any relevant data and also to handle errors that occur due to missingness. A t-test can be performed to see if the data is missing at random. This is done by creating a dummy variable indicating missingness. Then we split dummy values based on a categorical variable. We then test the mean missingness of each pair of categories and the significance level. From multiple testing we can find out if the data is missing at random with respect to the category. This process of multiple t-test is a cumbersome process therefore a simple visualisation test is done.

Fig 3: spine plot



The figure shows the similar result as a t test without multiple tests being conducted. The above figure is called a spine plot and is created by plotting missing vales in each category against missing values in Mustard oil. From the graph we can see that the proportion of missing data for each category changes based on the missingness in mustard therefore it is MAR.

As the data is missing at random we can fill in the missing values using multiple imputations. Amelia is one of the functions that can be used to impute the data set. Here the dataset is imputed 30 times using the Amelia package. Amelia uses an algorithm that uses expectation maximisation to input values in the missing area. This means it uses an initial set of values to generate a sequence of the most appropriate solutions.

With the imputed data set, analysis can be performed to check the relation between mean rainfall and price of onion. For this, a linear model is used as both the variable to be tested are numeric. The linear model is used to find the effect of mean rainfall on the price of onion and is written as lm(onion price ~ mean rainfall). After getting the list that contains all the imputations, the linear model is applied to each of the 30 imputations using the lapply function. After applying the model we can get the coefficient and standard error of all the imputations. To get a pooled or averaged result we use a function called mi.meld. This function applies Rubin’s rules to the summary we extracted and the result is shown in table.

Table 6(a): Estimate from mi.meld

Izntercept	Estimate of Rain mean
12.8403	0.0311

Table 6(b) : Standard Error from mi.meld

Intercept	Standard error of Rain mean
1.158	0.0076

From table 6(a) we get the estimate of mean rain, which indicates the amount by which the price of onion changes with a unit change in rainfall. As the estimate is positive it proves a positive correlation between rainfall and the price of onion. The correlation is small but is significant. This means that for every 100 mm increase in rainfall it is estimated that the price of onion increases by 3 Rs. The standard error gives the amount by which the estimated value varies from the actual value of the dataset. We can find out the confidence interval from these two coefficients by a simple formula. The upper confidence interval is the estimate plus 2 times standard error and the lower confidence interval is the estimate minus 2 times standard error. We get 0.016 as the lower bound and 0.046 as the upper bound. This means there is a 95% chance that the actual value lies between this interval. As this confidence interval does not contain Zero this estimate is statistically significant. If we consider the null hypothesis that this variable (mean rain) does not affect the dependent variable (Onion price) the estimate is almost zero and the confidence interval will contain zero. But as our confidence interval is far from zero it will give a low p-value implying the estimate is none zero and is statistically significant.

We can check to see if our analysis is true by extracting the summary of a imputation using Tidyverse.
We get the summary as shown below and we see that our analysis was interpreted correctly.

Output 1: Summary of linear model implemented in imputation 1

```
##                    term    estimate   std.error statistic    conf.low
## (Intercept) (Intercept) 12.75679141 1.112845718 11.463217 10.56450590
## rain_mean     rain_mean  0.03299165 0.007414873  4.449389  0.01838449
##                  conf.high      p.value
## (Intercept) 14.94907692 1.637585e-24
## rain_mean    0.04759881 1.323011e-05
```

As we check the summary of different imputations the p-value and the other coefficients remain almost identical. Therefore our imputation performed was a success. Figure 4(a) shows the density plot of original data red in colour and a few other random imputations. the density plot of the imputation even though are not identical they are acceptable. An imputation with a density close to the original one is blue in colour in figure 4(a) and is the imputation number 2. Figure 4(b) shows a linear plot of imputations and the original plot (red) it is clear that they are almost identical.

Fig 4(a): Density plot

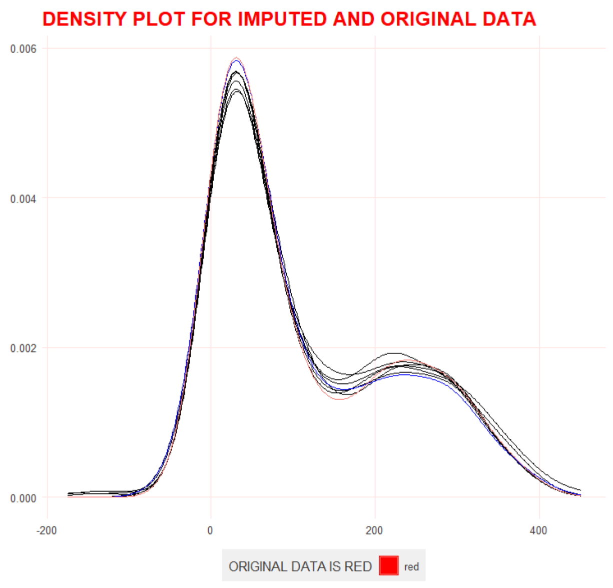
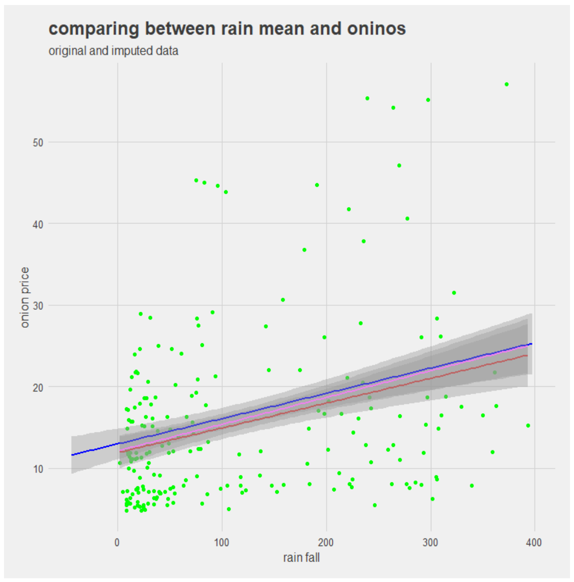


Fig 4(b): Linear plot



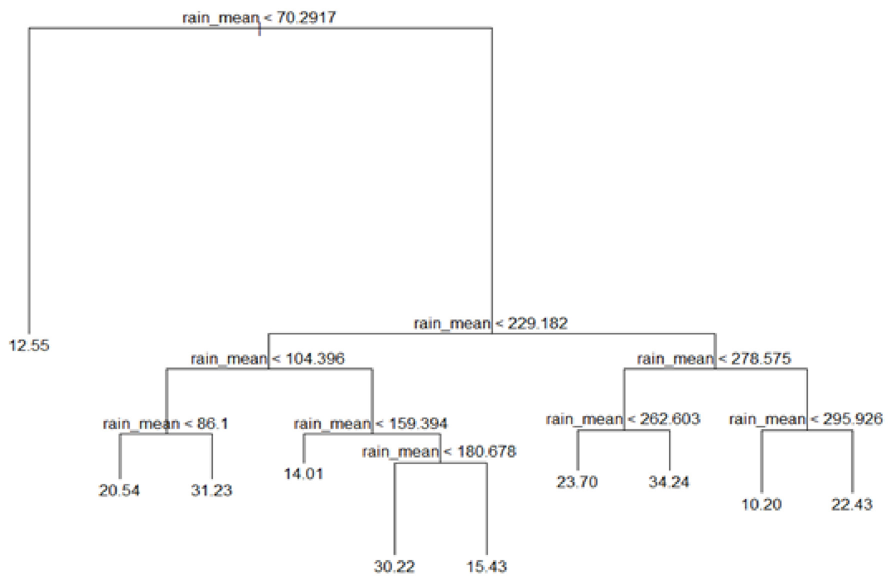
For the second part of my analysis, the relation between the price of mustard oil and rainfall is considered. To perform this an imputation is taken from the list of imputations. The imputation 10 was chosen for the analysis. After implementing the linear model, we get a small P-value with an estimate greater than zero. This linear model is performed multiple times by creating a lag in the mean rainfall to simulate the effect of different time periods of rainfall. The test shows a relation between rainfall and the price of mustard oil. The thought that rainfall does not affect the price of a product with long shelf life like mustard oil (9-12 months) was wrong. There could be other factors influencing this result. One thing that can be inferred is that rainfall affects the crop every month. This can be inferred from Table 7. Here a hypothesis that the crops harvesting time must be close to 1 month can be made. On using external resource, it is verified that it takes only 1 month for mustard to grow.

Table 7: Different coefficient and P-value for different interval of rain received

Lag time	ESTIMATE	STD ERROR	T value	P- value
3 month	0.034	0.01421	2.4	0.016
4 month	0.031	0.01421	2.2	0.026
5 month	0.0349	0.0141	2.4	0.014
6 month	0.029	0.01424	2.0	0.042

To find the non-linear relationships in a dataset we can use a tool called a decision tree. A decision tree can be used to predict values. Here the tree package is used to perform a decision tree. There are two modes of decision trees and here we are performing a regression-based decision tree. The first step is to make a model. Here the decision tree is used to predict the price of onion based on the mean rain. The second step is to split the data into two types, a training set and a test set. We split the data set in such a way that the split has similar distribution in training and test data. During the decision tree process, the data is split into groups according to features that will result in higher similarity within the groups. This process is done till we get pure leaf nodes as in figure 5. A prediction of the value of our data point is made by finding the average of all values in the leaf nodes. Some values that influence the training process are called hyperparameters. These parameters can have a great impact on the model's performance. Hyperparameters include the minimum number of data points needed for split, maximum depth of a tree and penalty for complexity. Finding the optimal parameters is called tuning.

Fig 5: Decision tree



After creating a decision tree as shown in figure 5 the model was implemented to a predictor function. Table 8(a) shows the predicted value and the actual value. To check the accuracy of the decision tree MSE or mean square error was calculated. The mean square error is high indicating low accuracy. The predictions shown in the table vary by a great margin from the original value. Which is represented by the high mean square error. To check if the accuracy will be increased by reducing the number of branches pruning was implemented. To find a suitable size to prune the tree cross-validation is performed and a graph plotting the MSE against the number of nodes is plotted as shown in figure 6. On putting the number of nodes as 2 we find the mean square error to reduce. But the number of predictions will decrease to 2 which is not suitable. Therefore, we select our original model.

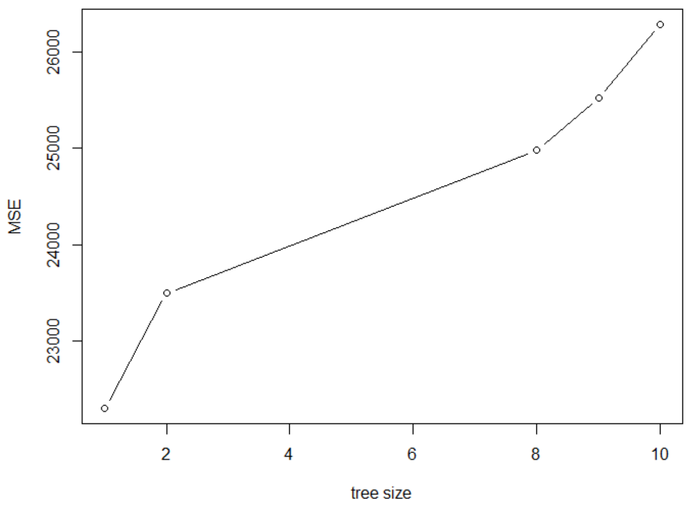
Table 8(a)

Predicted Price	Actual price
12.54	5.68
31.23	7.45
20.53	23.89

Table 8(b)

Pruned	Mean Square Error	Root mean square error
No	117.2	10.8
Yes	79.07	8.89

Fig 6: mean square error for different tree size



LIMITATIONS

In the analysis, only a few factors that affect or influence the variables could be taken into account. In order to get a more accurate analysis, various tuning and models must be implemented. To find how much each variable affects one another a large dataset is required. In the case of how rain influences the price of food products, only a simple lagging function was used by changing the column of the dataset based on the time taken for the crop to grow. Various factors that affect the growth of a crop could not be taken into account. Only a simple linear model could be implemented with the current data set for more accurate results a large dataset with various variables must be used. Simple Linear models use many assumptions and this will lead to reduced accuracy. A proper relation between food products with long shelf life and rainfall could not be analysed. In the case of the decision tree, the number of data available to split into train and test data was low which caused the high means square error and low prediction accuracy.

CONCLUSIONS

The statistical analysis was done using statistical models. The analysis resulted in proving a relationship between rainfall and the price of onion. Anova hypothesis was performed on the dataset to show the relationship between different categories of rain and the price of onion. The dataset had multiple missing values and implicit missing values. This missingness was dealt with using the Amelia multiple imputation method here an expectation maximisation algorithm was used to impute missing values.

A linear model was implemented on the multiply imputed dataset and the relation between the price of Onion and rainfall was established. Rubins's rule was used on the extracted coefficient to get the estimate and standard error from which the confidence interval was calculated. From the confidence interval and coefficients, a relation was established. An analysis of the effect of food product with long shelf life was made using a linear model. To perform a non-linear prediction a decision tree was made. The model was first created then data was split into train and test data. The train data was used to train the model. Analysis of the trained model was done by implementing the test data on the model. A predictor function was created and implemented on the model. The model made predictions with low accuracy.

APPENDIX

```
library(Amelia)
library(tidyverse)
library(mice)
library(naniar)
library(broom)
library(ggplot2)
library(ggthemes)
library(survival)
library(corrplot)
library(ggThemeAssist)
library(esquisse)
library(tidymodels)
library(agricolae)
library(Stat2Data)
library(tree)
library(VIM)


#FOOD COST DATASET
food_price = read.csv("https://raw.githubusercontent.com/juanpaul750/indian-food-price/main/india.csv")

# WRANGLING
food_price = food_price %>%
  rename( Mustard_oil = "Oil..mustard.", Potato = "Potatoes", Date = "date"  )
head(food_price)

# TABS AND PLOTS OF MISSING VALUES
md.pattern(food_price)
missing_dt_food = miss_var_summary(food_price)
missmap(food_price)

# RAIN DATASET
rain = read.csv("https://raw.githubusercontent.com/juanpaul750/rain-in-
india/main/Sub_Division_IMD_2017.csv")

# WRANGLING
rain = rain%>%
  select(SUBDIVISION, YEAR, JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV,
DEC) %>%
  rename(Region = "SUBDIVISION")

# ALL IMPLICIT MISSING DATA
rain = rain%>%
  tidyr::complete(Region,YEAR)  # Note: found some implicit data

# TABS AND PLOTS OF MISSING VALUES
missing_dt_rain = miss_var_summary(rain)
missing_dt_rain = missing_dt_rain %>%
  rename(no_of_miss = "n_miss", percent_miss = "pct_miss")

gg_miss_var(rain) + labs(y = " number of missing values")+
  theme(axis.title = element_text(size = 20),
        axis.text = element_text(size = 15))
gg_miss_var(rain, show_pct = T) + theme(axis.title = element_text(size = 20),
    axis.text = element_text(size = 15))

# PLOTIN USING BIND SHADOW
food_price %>%
  bind_shadow()%>%
  ggplot(aes(x= Potato,
             colour = Onions_NA)) +
  geom_density()


# DATA WRANGLING
rain = arrange(rain, YEAR)
rain = rain[rain$YEAR > 1998, ]

# CREATING MEAN RAIN FALL DATASET
df = group_by (rain,YEAR)
dtd = summarise(df, JAN = mean(JAN), FEB = mean(FEB), MAR = mean(MAR), APR = mean(APR),
MAY = mean(MAY), JUN = mean(JUN),
                JUL = mean(JUL),AUG = mean(AUG), SEP = mean(SEP), OCT = mean(OCT), NOV =
mean(NOV), DEC = mean(DEC) )

# USE GATHER FUNTION TO CREAT A LONG FORMAT DATASET
rain_column = dtd%>%
  gather(month,rain_mean,-1)%>%
  arrange(YEAR)

# CREATING NEW COLUMN WITH RAIN CATEGORY
rain_column = rain_column[-c(1:5),]
rain_column = mutate(rain_column, sl.no = c(1:223),rain_category =
                       case_when(rain_mean <= 15 ~ "Light Rain",
                                 rain_mean <= 50 ~ "Med Rain",
                                 rain_mean <= 120 ~ "Heavy Rain",
                                 rain_mean > 120 ~ "Very Heavy Rain",))
rain_column$rain_category = as.factor(rain_column$rain_category)
food_price = mutate(food_price, sl.no = c(9:248))

# MERGE THE TWO DATASET
food_rain = merge(food_price,rain_column, by = 'sl.no', all = TRUE)
food_rain =
select(food_rain,Date,Chickpeas,Mustard_oil,Potato,Rice,Sugar,Wheat,Onions,rain_mean,rain_category )

# T-TEST TO CHECK MAR
food_rain_ttest = food_rain[-c(1:8),]
```

```
# T - TEST FOR MUS
food_rain_ttest = food_rain_ttest %>%
  mutate(missing_oil = is.na(Mustard_oil))

# T test for combo
missing_oil_light = food_rain_ttest %>%
  filter(rain_category == "Light Rain") %>%
  pull(missing_oil)

missing_oil_med = food_rain_ttest %>%
  filter(rain_category == "Med Rain") %>%
  pull(missing_oil)

missing_oil_heavy = food_rain_ttest %>%
  filter(rain_category == "Heavy Rain") %>%
  pull(missing_oil)

missing_oil_vheavy = food_rain_ttest %>%
  filter(rain_category == "Very Heavy Rain") %>%
  pull(missing_oil)

t.test(missing_oil_light, missing_oil_med)
t.test(missing_oil_vheavy, missing_oil_med)
t.test(missing_oil_heavy, missing_oil_med)
t.test(missing_oil_light, missing_oil_heavy)
t.test(missing_oil_vheavy, missing_oil_heavy)

# VISUALISING TEST FOR MAR

food_rain_ttest %>%
  select(rain_category,Onions)%>%
  spineMiss()

food_rain_ttest %>%
  select(rain_category, Mustard_oil)%>%
  spineMiss()


# CREATING THE LAG TO ACCOUNT FOR THE CROP HARVEST TIME
# LAG TO COMPARE ONION PRICE AND RAIN MEAN
f = 3
lag_onion = food_rain %>%
  mutate(rain_mean = lag(rain_mean, n = f))%>%
  mutate(rain_category = lag(rain_category, n = f))


# ANOVA HYPOTHESIS TESTING TEST WITH onion data
lag_on_grp = arrange(lag_onion, rain_mean)
lag_on_grp$rain_category = factor(lag_on_grp$rain_category , levels=c('Light Rain' , 'Med Rain', 'Heavy
Rain','Very Heavy Rain'))
data_mean = aggregate(lag_on_grp$Onions,list(lag_on_grp$rain_category),mean,na.rm = T)
```

```
# BOX PLOT
lag_on_grp %>%
  filter(!is.na(rain_category)) %>%
  ggplot()+
  aes(x = rain_category, y = Onions, fill = rain_category)+
  geom_boxplot()+
  theme(legend.position="none")+
  stat_summary(fun = 'mean', colour = 'black', geom = 'point' )+
  stat_summary(fun = 'mean', colour = 'black', geom = 'text',
               vjust = 1,aes(label = paste("Mean:",round(..y.., digits = 1))))+
  labs(
    x = "Rain Category",
    y = "Onion Price ( in RS)",
    title = "Mean and Meadian"
  )

# TABLE TO SHOW SD AND MEAN
lag_on_grp %>%
  group_by(rain_category)%>%
  summarize(mean(Onions, na.rm = TRUE), sd(Onions, na.rm =TRUE) )

# ANOVA TEST SUMMARY
categories_of_rain = lag_onion$rain_category
out = aov(lag_onion$Onions ~ categories_of_rain)
summary(out)

#MULTIPLE COMPARISION
# PAIRWISE T TEST WITHOUT CORRECTION
with(lag_onion, pairwise.t.test(Onions, rain_category, p.adjust.method = "none"))

# BONFERRONI CORRECTION
with(lag_onion, pairwise.t.test(Onions, rain_category, p.adjust.method = "bonferroni"))

# TUKEYS CORRECTION FOR PLOTING CONFIDENCE INTERVAL
TukeyHSD(out)
par(mar = c(4.5, 11.5, 4.5, 1.9))
plot(TukeyHSD(out),  las = 1)


# MISSING VALUE IMPUTAION USING AMELIA FOR ONION PRICE ANALYSIS
lag_onion_w = lag_onion[-c(1:8),]
head(lag_onion_w)
itr_food_rain = amelia(x = lag_onion_w, m=30, idvars = "Date", ords = 'rain_category')

# LINEAR MODEL
lm.out.food.rain = lapply(itr_food_rain $ imputation,function(i) lm(Onions ~ rain_mean , data = i))

coefs.food.rain = lm.out.food.rain %>%
  sapply(coef)%>%
  t()
```

```
coefs.food.rain

my_stdE_extr = function(model){summary(model)$coef[,"Std. Error"]}

ses.food.rain = lm.out.food.rain %>%
  sapply(my_stdE_extr)%>%
  t()

mi.meld(coefs.food.rain, ses.food.rain)

# TIDY SUMMARY
al_imp_oni = bind_rows(unclass(itr_food_rain$imputations), .id = "m") %>%
  group_by(m) %>%
  nest()

mod_imputations = al_imp_oni %>%
  mutate(model = data %>%
         map(~ lm(Onions ~ rain_mean , data = .)),tidied = model %>%
         map(~ tidy(., conf.int = TRUE)),glance = model %>%
         map(~ glance(.)))

#models_imputations
mod_imputations %>%
  filter(m == "imp1") %>%
  unnest(tidied)

# FINDING COEFFICENT AND STD ERROR
param_coef_er = mod_imputations %>%
  unnest(tidied) %>%
  select(m, term, estimate, std.error) %>%
  gather(key, value, estimate, std.error) %>%
  spread(term, value)

# EXTRACTING COEFFICIENT
just_coefs = param_coef_er %>%
  filter(key == "estimate") %>%
  ungroup %>%
  select(-m, -key)

# EXTRACTING STD ERROR
just_ses = param_coef_er %>%
  filter(key == "std.error") %>%
  ungroup %>%
  select(-m, -key)

coefs_melded = mi.meld(just_coefs, just_ses)

# EXTRACTING DEGREE OF FREEDOM

model_degree_freedom = mod_imputations %>%
  unnest(glance) %>%
  filter(m == "imp1") %>%
  pull(df.residual)

# SUMMARY
mod_mel_summary = as.data.frame(cbind(t(coefs_melded$q.mi), t(coefs_melded$se.mi))) %>%
  magrittr::set_colnames(c("estimate", "std.error")) %>%
  mutate(term = rownames(.)) %>%
  select(term, everything()) %>%
  mutate(statistic = estimate / std.error,
         conf.low = estimate + std.error * qt(0.025, model_degree_freedom),
         conf.high = estimate + std.error * qt(0.975, model_degree_freedom),
         p.value = 2 * pt(abs(statistic), model_degree_freedom, lower.tail = FALSE))

mod_mel_summary
coefs_melded

# PLOTING
# DENSITY PLOT FOR FEW IMPUTAIONS
ggplot()+
  geom_density(aes(x = rain_mean ), data = itr_food_rain$imputations$imp11)+
  geom_density(aes(x = rain_mean),  data = itr_food_rain$imputations$imp30)+
  geom_density(aes(x = rain_mean), data = itr_food_rain$imputations$imp6)+
  geom_density(aes(x = rain_mean), data = itr_food_rain$imputations$imp16)+
  geom_density(aes(x = rain_mean), data = itr_food_rain$imputations$imp21)+
  geom_density(aes(x = rain_mean), data = itr_food_rain$imputations$imp26)+
  geom_density(aes(x = rain_mean),colour = 'blue', data = itr_food_rain$imputations$imp2)+
  geom_density(aes(x = rain_mean, colour = 'red'),  data = lag_onion )+
  ggtitle("DENSITY PLOT FOR IMPUTED AND ORIGINAL DATA")+
  theme_fivethirtyeight()+theme(axis.title = element_text())+
  labs(x= "rain fall (in mm)",y="DENSITY") +labs(colour = "ORIGINAL")+theme_fivethirtyeight()+
  theme(panel.grid.major = element_line(colour = "mistyrose"),
    plot.title = element_text(colour = "red"),
    panel.background = element_rect(fill = "white"),
    plot.background = element_rect(fill = "white",
        colour = "white"), legend.key = element_rect(fill = "red",
        colour = "red", linetype = "solid"),
    legend.background = element_rect(colour = "red")) +labs(x = "RAIN FALL (in mm)", colour =
"ORIGINAL DATA IS RED",
    fill = NULL)


ggplot(data = regr_model_1, aes(x = rain_mean, y = Onions))+
  geom_point(data = regr_model_1,color="green")+
  stat_smooth(method = 'lm', col = 'red')+
  stat_smooth(method = 'lm',aes(x = rain_mean),colour = 'blue', data = itr_food_rain$imputations$imp2)+
  stat_smooth(method = 'lm',aes(x = rain_mean),colour = 'violet', data =
itr_food_rain$imputations$imp10)+
  labs(x= "rain fall",
       y="onion price")+
  ggtitle("comparing between rain mean and oninos",
          subtitle = "original and imputed data")+
  theme_fivethirtyeight()+
  theme(axis.title = element_text())

# LINEAR MODEL FOR TESTING MUSTARD OIL PRICE

test_must = itr_food_rain$imputations$imp10

mustard_oil = lm(Mustard_oil ~ rain_mean , data = test_must)
summary(mustard_oil)


f = 1
test_must_1 = test_must %>%
  mutate(rain_mean = lag(rain_mean, n = f))%>%
  mutate(rain_category = lag(rain_category, n = f))

mustard_oil_1 = lm(Mustard_oil ~ rain_mean , data = test_must_1)
summary(mustard_oil_1)

f = 1
test_must_2 = test_must_1 %>%
  mutate(rain_mean = lag(rain_mean, n = f))%>%
  mutate(rain_category = lag(rain_category, n = f))

mustard_oil_2 = lm(Mustard_oil ~ rain_mean , data = test_must_2)
summary(mustard_oil_2)

f = 1
test_must_3 = test_must_2 %>%
  mutate(rain_mean = lag(rain_mean, n = f))%>%
  mutate(rain_category = lag(rain_category, n = f))

mustard_oil_3 = lm(Mustard_oil ~ rain_mean , data = test_must_3)
summary(mustard_oil_3)

# DECISION TREE FOR LAG ONION REGRESSION TEST
# Build the specification
#splitting data into train data test data
data_1 = itr_food_rain$imputations$imp2
head(data_1)

set.seed(1)
price_pred_split = initial_split(data_1, strata = Onions)

onion_train = training(price_pred_split)
onion_test  = testing(price_pred_split)
testing_Onion = onion_test$Onions

tree_model = tree(Onions ~ rain_mean, onion_train)
tree_model
```

```
# PLOT TREE MODEL
par(mar=c(1,1,1,1))
plot(tree_model)
text(tree_model, pretty = 0)

# Predict new data
price_pred_onion_num = predict(tree_model,onion_test)
price_pred_onion = predict(tree_model,onion_test)%>%
  bind_cols(onion_test)

#checking the accuracy of the model using testing data
# MEAN SQUARE ERROR (MSE)
MSE = mean((price_pred_onion_num - testing_Onion)^2)
MSE
#ROOT MEAN SQUARE ERROR
RMSE = sqrt(MSE)
RMSE

# TO CHECK THE SUITABLE SIZE OF THE TREE
#CROSS VALIDATION FOR PRUNING THE TREE

cv_tree = cv.tree(tree_model)
par(mar=c(4.5,4,3,2))
plot(cv_tree$size, cv_tree$dev,type = "b", xlab= "tree size",
     ylab ="MSE")

which.min(cv_tree$dev)
cv_tree$size[5]

# Checking to see if pruning helps

pruned_model = prune.tree(tree_model, best = 2)
plot(pruned_model)
text(pruned_model)

# check the accuracy
pruned_pred = predict(pruned_model, onion_test)
MSE_2 = mean((pruned_pred - testing_Onion)^2)

RMSE_2 = sqrt(MSE_2)
RMSE_2
```