Industrial Internship Report on

”Agricultural crop production in india”

Prepared by

[Kavisha Jain]

Executive Summary

This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT).

This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks’ time.

My project was Unveiling the Crop Whisperers: Revolutionizing Indian Agriculture with AI

In the heartlands of India, where the soil tells tales of generations past and the sky paints a canvas of uncertainty, a digital revolution is underway. Meet the Crop Whisperers – a team of data wizards armed with algorithms and innovation, on a mission to transform agricultural fortunes across the nation.

Their quest? To harness the power of AI predictive modeling, unraveling the mysteries of crop yields and weather whims. Armed with historical data and boundless creativity, they sculpt insights from the soil's secrets and the sky's whispers. Decision trees dance, linear regressions weave, XGBoost soars, and Random Forests sway – each algorithm a maestro in the symphony of agricultural prediction.

Through this digital odyssey, the Crop Whisperers pave a path to prosperity, empowering farmers with knowledge, policymakers with foresight, and India with a sustainable agricultural future. In this tale of

technology and tradition, the Crop Whisperers are the heroes, scripting a narrative of hope, growth, and abundance for the fields of India.

This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solutions for that. It was an overall great experience to have this internship.

TABLE OF CONTENTS

Preface

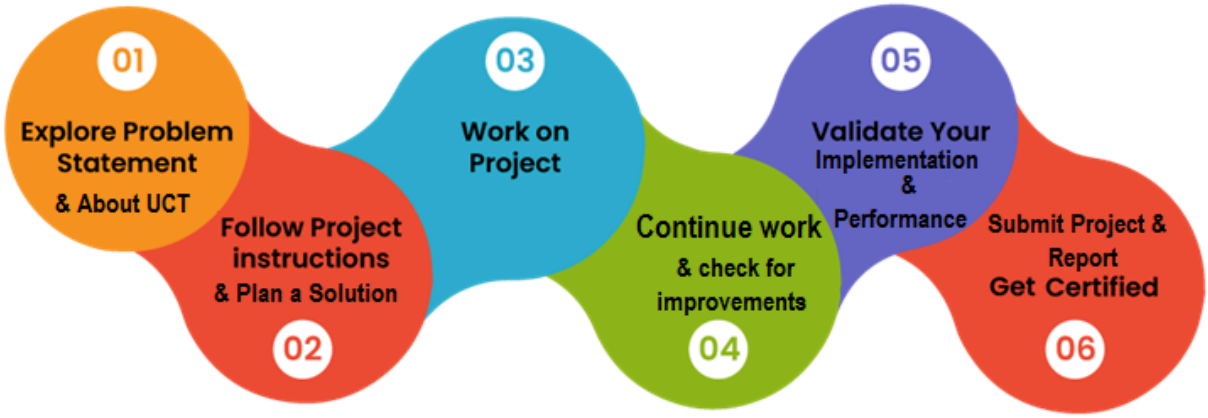
Summary of the whole 6 weeks' work.

About need of relevant Internship in career development.

Brief about Your project/problem statement.

Opportunity given by USC/UCT.

How Program was planned



Your Learnings and overall experience.

Thank to all (with names), who have helped you directly or indirectly.

Your message to your juniors and peers.

Introduction

1.1 About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various **Cutting Edge Technologies e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end** etc.

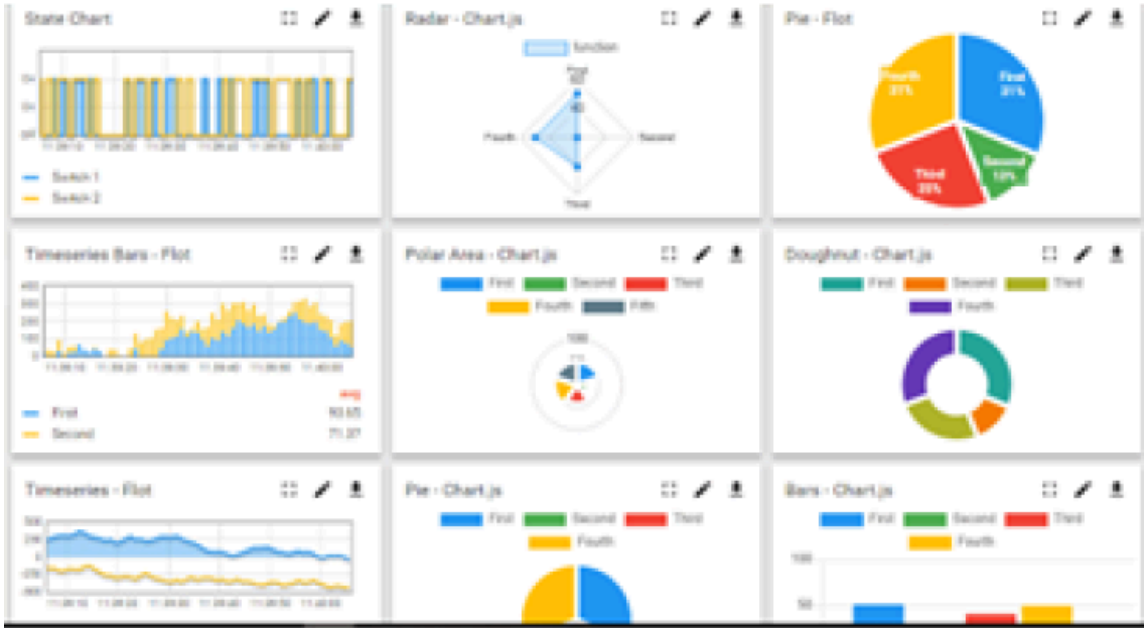
i. UCT IoT Platform (uct Insight)

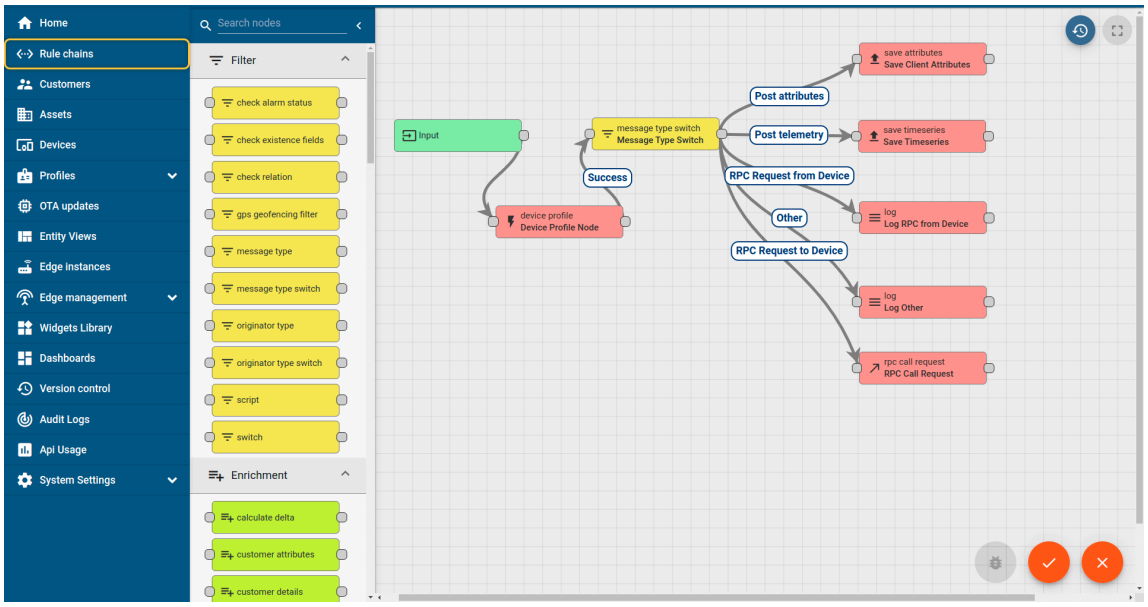
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable “insight” for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA
- It supports both cloud and on-premises deployments.

It has features to

- Build Your own dashboard
- Analytics and Reporting
- Alert and Notification
- Integration with third party application(Power BI, SAP, ERP)
- Rule Engine





ii. Smart Factory Platform (FACTORY WATCH)

Factory watch is a platform for smart factory needs.

It provides Users/ Factory

- with a scalable solution for their Production and asset monitoring
- OEE and predictive maintenance solution scaling up to digital twin for your assets.
- to unleased the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
- A modular architecture that allows users to choose the service that they what to start and then can scale to more complex solutions as per their demands.

Its unique SaaS model helps users to save time, cost and money.

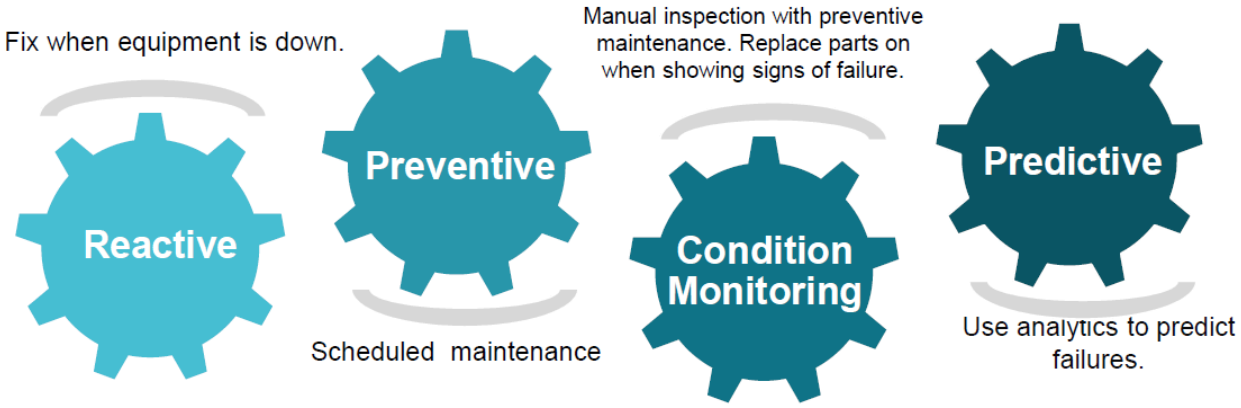
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i

iii. LoRaWAN based Solution

UCT is one of the early adopters of LoRAWAN teschnology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

iv. Predictive Maintenance

UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



1.2 About upskill Campus (USC)

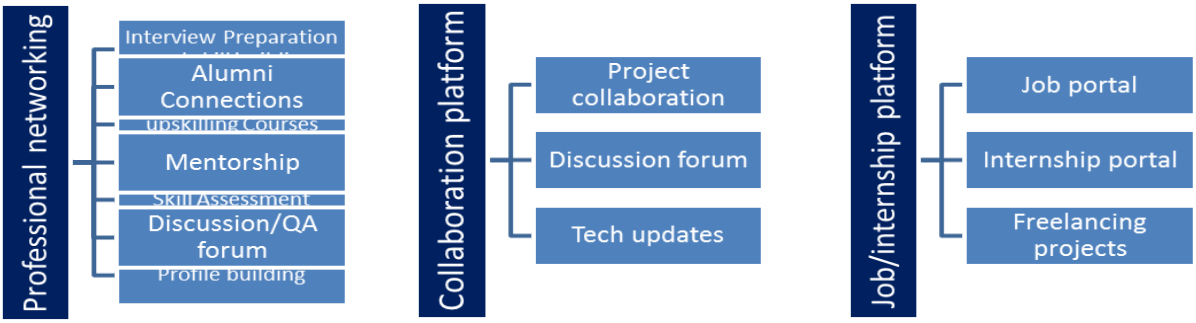
upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.

Seeing need of upskilling in self paced manner along-with additional support services e.g. Internship, projects, interaction with Industry experts, Career growth Services

upSkill Campus aiming to upskill 1 million learners in next 5 year

https://www.upskillcampus.com



1.3 The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

1.4 Objectives of this Internship program

The objective for this internship program was to

☛ get practical experience of working in the industry.

☛ to solve real world problems.

☛ to have improved job prospects.

☛ to have Improved understanding of our field and its applications.

☛ to have Personal growth like better communication and problem solving.

1.5 Reference

[1]Journal/Research Paper references:

[1] Ahmad, I., Saeed, U., Fahad, M., Ullah, A., Habib-ur-Rahman, M., Ahmad, A., Judge, J., 2018. Yield forecasting of spring maize using remote sensing and crop modeling in Faisalabad-Punjab Pakistan. J. Indian Soc. Remote Sens. 46 (10), 1701–1711. https://doi.org/10.1007/s12524-018-0825-8.

[2] Ali, I., Cawkwell, F., Dwyer, E., Green, S., 2017. Modeling managed grassland biomass estimation by using multitemporal remote sensing data—a machine learning ap- proach. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 10 (7), 3254–3264. https:// doi.org/10.1109/JSTARS.2016.2561618.

[3] Girish, L., Gangadhar, S., Bharath, T., Balaji, K., n.d. Crop Yield and Rainfall Prediction in Tumakuru District using Machine Learning. Ijream.Org. Retrieved from https:// www.ijream.org/papers/NCTFRD2018015.pdf.

1.6 Glossary

Terms	Acronym
ML	Machine learning
AI	Artificial intelligence
XGBOOST	eXtreme Gradient Boosting

Problem Statement

In the assigned problem statement

3.1 Problem statement:

Agriculture is a vital aspect of human civilization, providing sustenance and livelihood for millions worldwide. India, with its diverse climates, vast agricultural landscapes, and rich agricultural heritage, plays a significant role in the global agricultural sector. This report explores India's extensive dataset on crop production since 2001, including various crop types, years of production, quantity, cultivation area, and associated costs across different states. India's agricultural sector is a testament to its rich history and connection to the land. As the world's second-most populous country, India's agricultural prowess has far-reaching implications on global food security, trade, and environmental sustainability. The country's agricultural prowess is not limited to domestic concerns.

3.2 Factors associated with the problem statement:

India's diverse agro-climatic zones provide a diverse array of crops, each contributing unique offerings to the nation's production. The dataset analyzed includes their types, production quantities, cultivation areas, and associated costs. India's prominence in the agricultural sector is due to its significant contributions to global food production and its immense agricultural potential. The country ranks second in arable land globally, making it an agricultural powerhouse. Agriculture in India employs a substantial portion of the population, particularly in rural areas, where it is the primary occupation. The agricultural sector not only provides employment but also plays a crucial role in ensuring food security and poverty alleviation.

3.3 About the dataset:

The dataset provides a detailed overview of India's agricultural landscape, revealing the factors affecting crop production and sustainability. It covers various crops grown in different regions, revealing the country's geographical diversity. The dataset also offers insights into annual crop production quantities, providing a quantitative understanding of crop output over time. It also identifies factors affecting yield fluctuations, such as climate variability, technological advancements, and market demands. The spatial distribution of crops across the country, measured in hectares, reveals regional specialization, resource allocation, and the impact of land-use patterns on agricultural productivity.

3.4 More About the dataset:

The dataset focuses on the economic aspects of crop cultivation, incorporating cost-related data to understand financial challenges faced by farmers and potential barriers to agricultural sustainability. It helps policymakers and stakeholders develop strategies to optimize input usage, improve efficiency, and enhance farmers' livelihoods. The dataset also includes information on cultivation area in hectares, enabling the assessment of crop productivity and resource allocation in different regions. This information guides policymakers and farmers towards optimal land use and crop selection strategies.

Understanding the cost dynamics in the agricultural sector is crucial for evaluating profitability, economic viability, and sustainability of farming practices. Factors affecting profitability include input costs, labor expenses, and market conditions. This knowledge can help formulate strategies to enhance agricultural productivity and address challenges faced by farmers.

The project aims to harness data science and machine learning to predict suitable crops for specific agricultural regions. By training and evaluating various machine learning models, accurate prediction algorithms can guide farmers in selecting crops most suitable for their location, climate conditions, and available resources. Such predictions have the potential to increase agricultural productivity, reduce risks, and contribute to the sustainable development of India's agricultural sector.

Existing and Proposed solution

METHODOLOGY

Figure2.1: Help determine any outlier

2.2 Data Analysis and EDA:

Exploratory Data Analysis Techniques need to be performed for examining the relationship between variables, identify trends, correlations, and distribution within the dataset. Visualization tools in the python language can really become handy in such cases where a deeper understanding of the dataset is required.

To know, how the dataset is distributed within itself, a boxplot can be handy which was also used for the better understanding.

Figure 1.2: Boxplot, to understand distribution

A simple plot can be used to show what crop was produced in which quantity and in what quantity.

Figure 2.2: Plot to understand the produce over years

It is important to understand that India has a good number of states and that too with different climatic conditions all together. Hence, a statewise understanding of what crops are cultivated and produced holds a significant importance to us.

Figure 2.3: Bar graph, to understand the state-wise produce of different crops

2.3 Comparative Analysis:

It's very important to realise that a country with more than 140+ crore people, the area vs yield of crop has to be checked at regular intervals. Therefore, to understand this well, though there were four graphs for one financial year, a combined graph was made to showcase the Area vs Yield.

Figure 2.4: Area vs Yield in 2006-2007

Figure 2.5: Area vs Yield in 2010-2011

With the rapid increase in population over the years, a comparative study needs to be done, about how the area, yield and production have increased or decreased over the years, for every individual crop if possible. The below graphs show the pattern of increase or decrease.

Figure 2.6: Production in 2006-07 vs 2010-11

Figure 2.7: Land utilized in year 2006-07 vs 2010-11

Figure 2.8: Yield in the year 2006-06 vs 2010-11

The graphs above can also be used while understanding why the control in population is required, as there are two things that are somewhat in contrast as of today, urbanisation and growth in agriculture, because to feed a continuously growing population we need larger areas, but again because of increasing population, we need places to live.

Figure 2.9: Per hectare cost of major crops

Figure 210: Per quintal cost of production

Figure 2.11: Crop yield

From the pie charts that have been displayed above, certain useful information can be drawn which include:

*Crop Cultivation-

-Sugarcane has a highest Yield in India (82.2%)

-Moong has a lowest Yield in India (0.437%)

*Cultivation cost for crop-

- Highest for sugarcane

- Lowest for moong

- Average for 'paddy, maze, cotton, wheat' etc.

*Production cost of crop-

- Sugarcane has very minimal production cost

- Moong has the highest production cost and hence, lowest yield in the country.

Figure 2.12: Statewise crop production, per hectare.

Figure 2.13: State-wise, production cost, per quintal

The need of these pie chart is to gather the information at a root level, which can help us tell someone, which crop can be an alternative source of income/cultivation for the farmers. In the current times, nobody should rely on a single source of income.

*State-wise agricultural observations-

- Karnataka is the highest in yield.

- Odisha is least in yield.

- Andhra Pradesh and Maharashtra spend maximum in cultivation as well in production cost for agriculture.

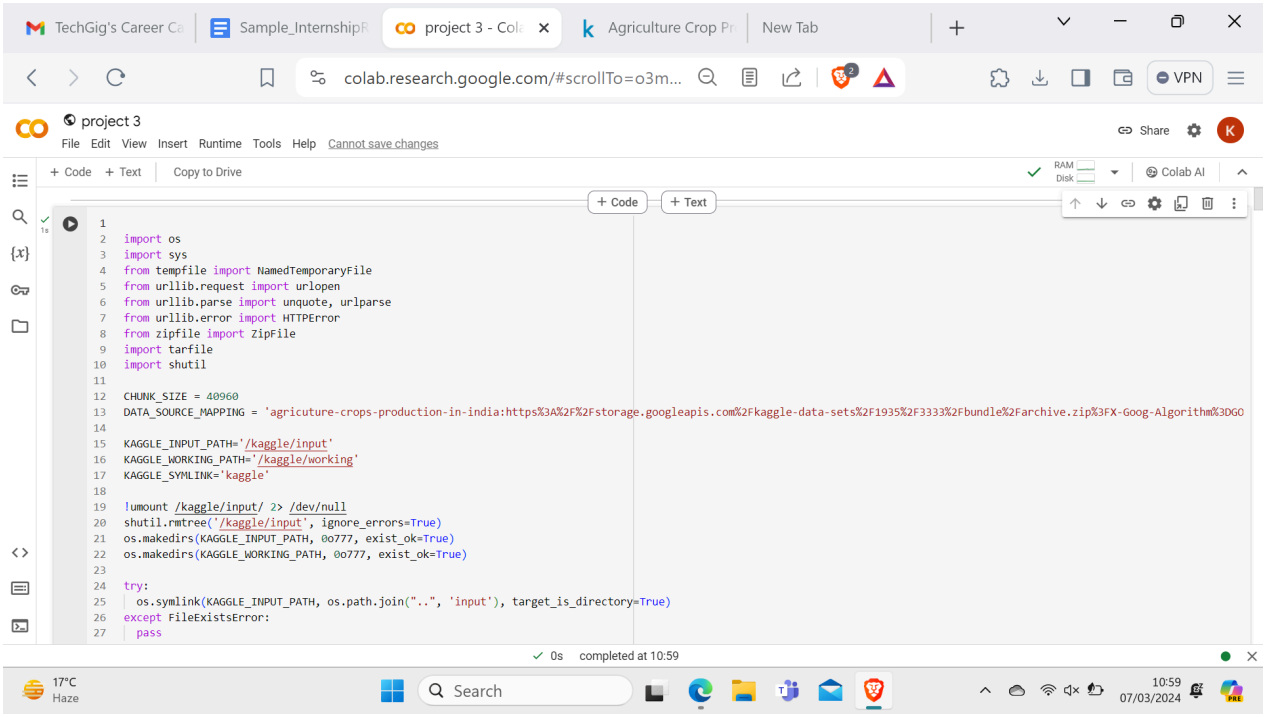
- Bihar and West Bengal spends very less in cultivation as well in production cost for agriculture.

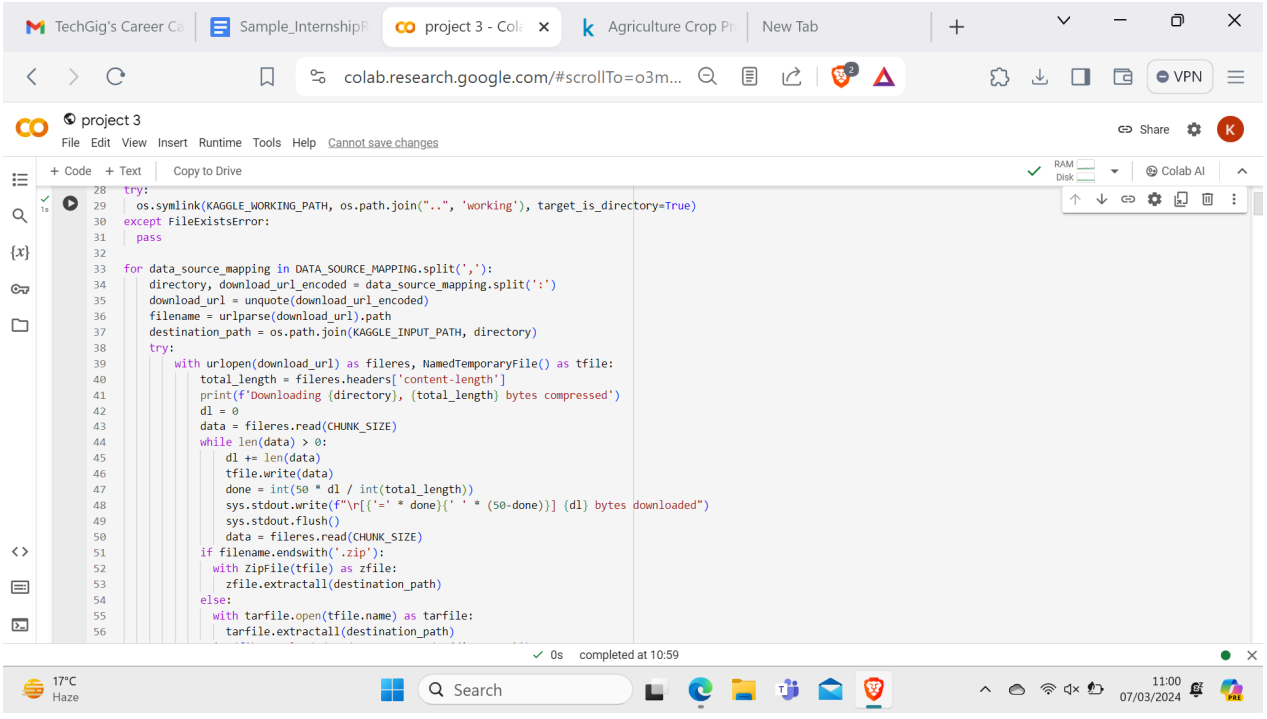
Figure 2.14: Part of code showing the accuracy of the random forest model applied to the dataset for predicting the agricultural crop in some particular part of India at some point of time.

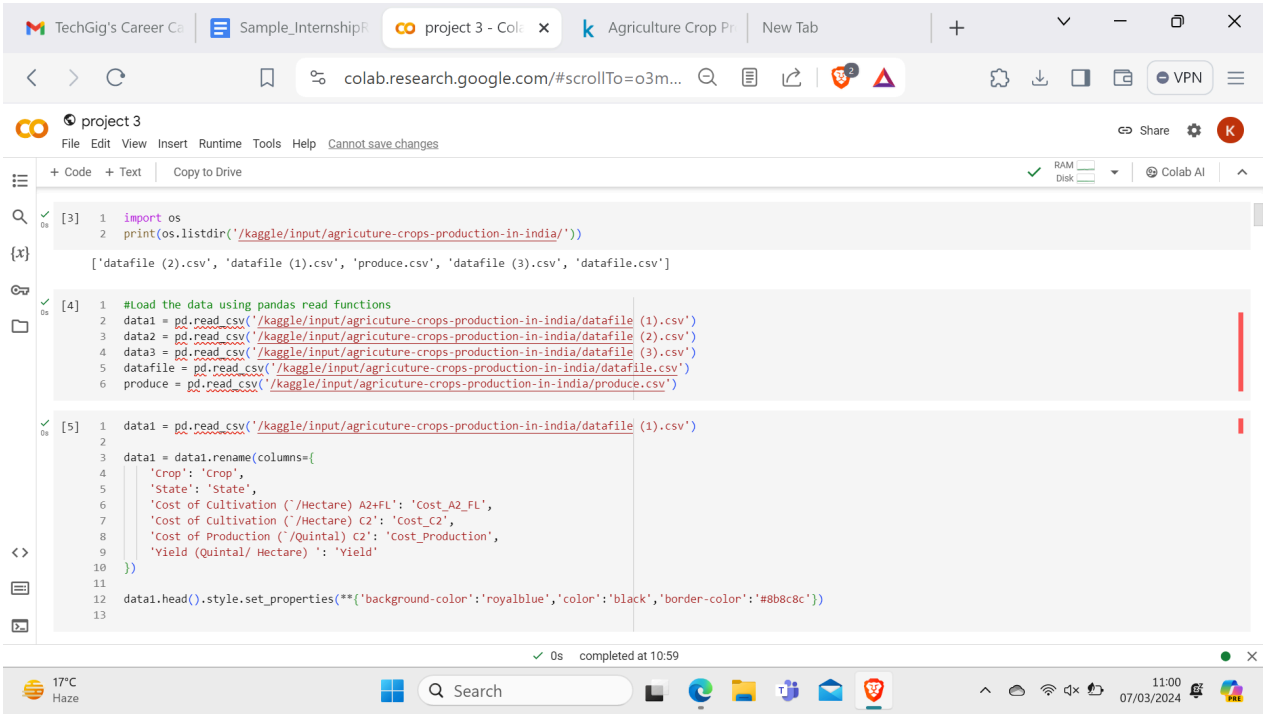
For confidential reasons, as the project is given by a particular organisation to work upon, due to confidential reasons, the code cannot be shown in entirety.

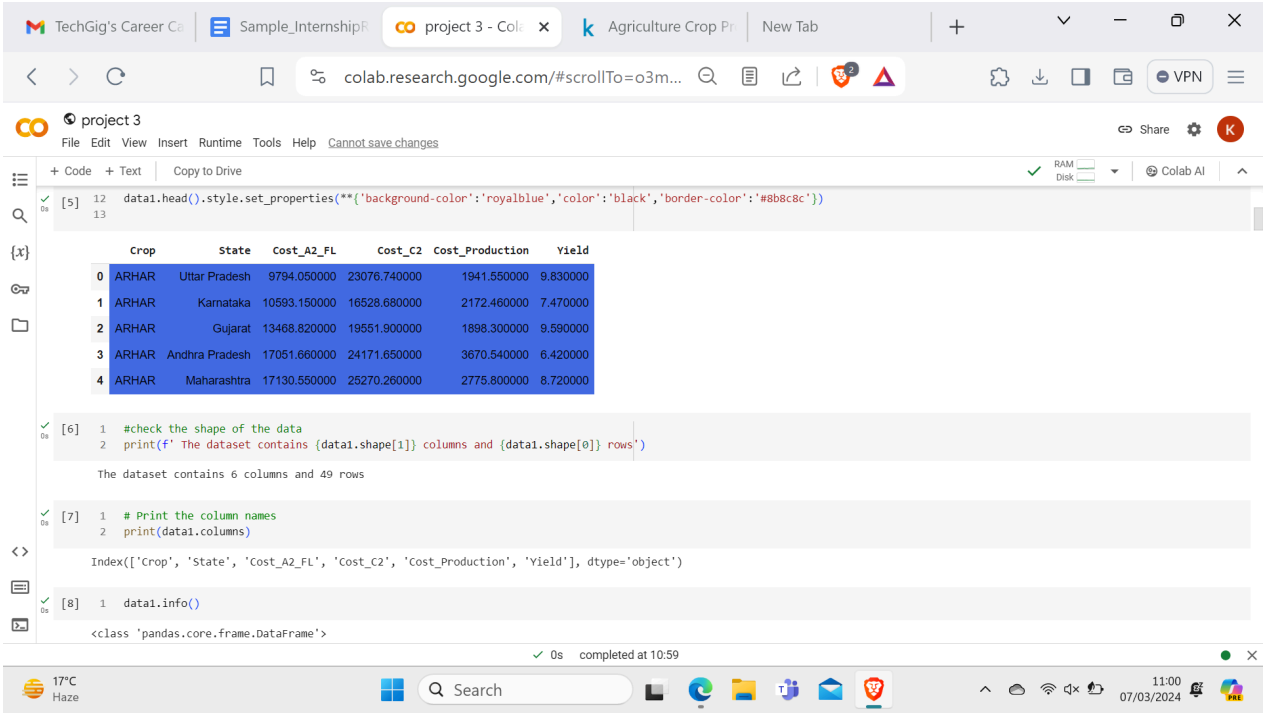
To gather the appropriate information about the figures and their usage, kindly refer to the last page, under the references column.

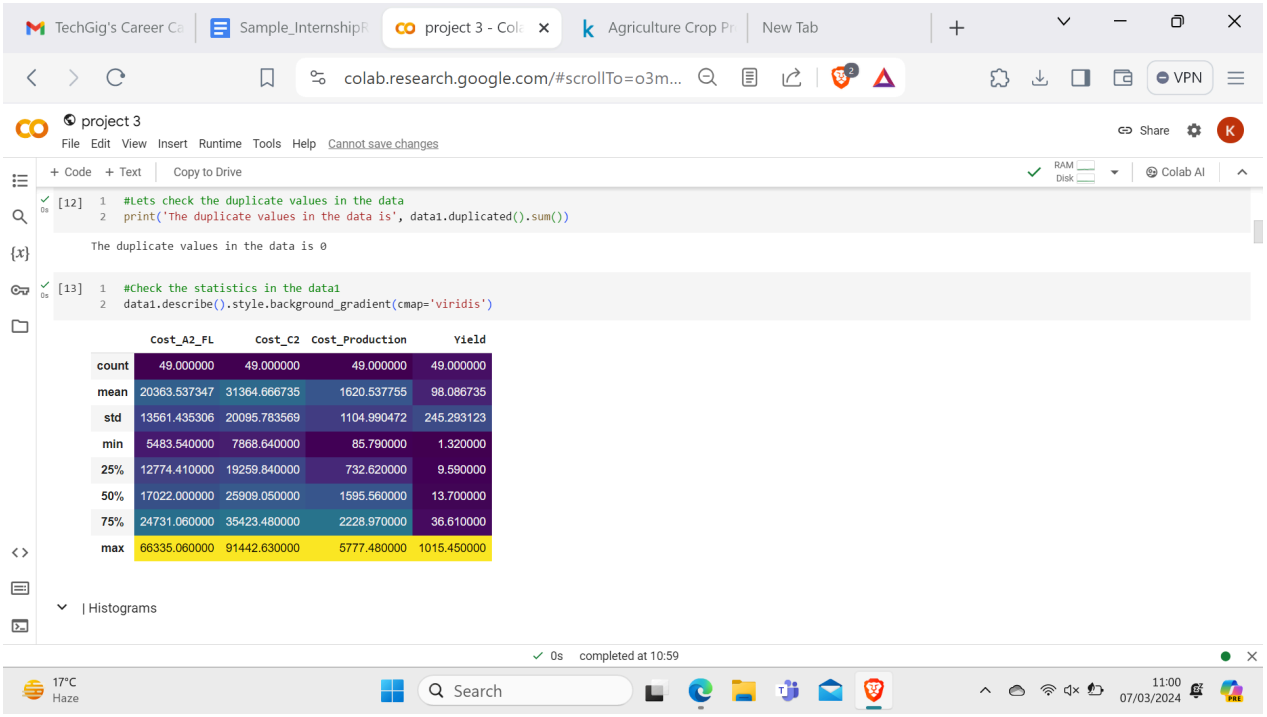
1.7 Code submission (Github link)

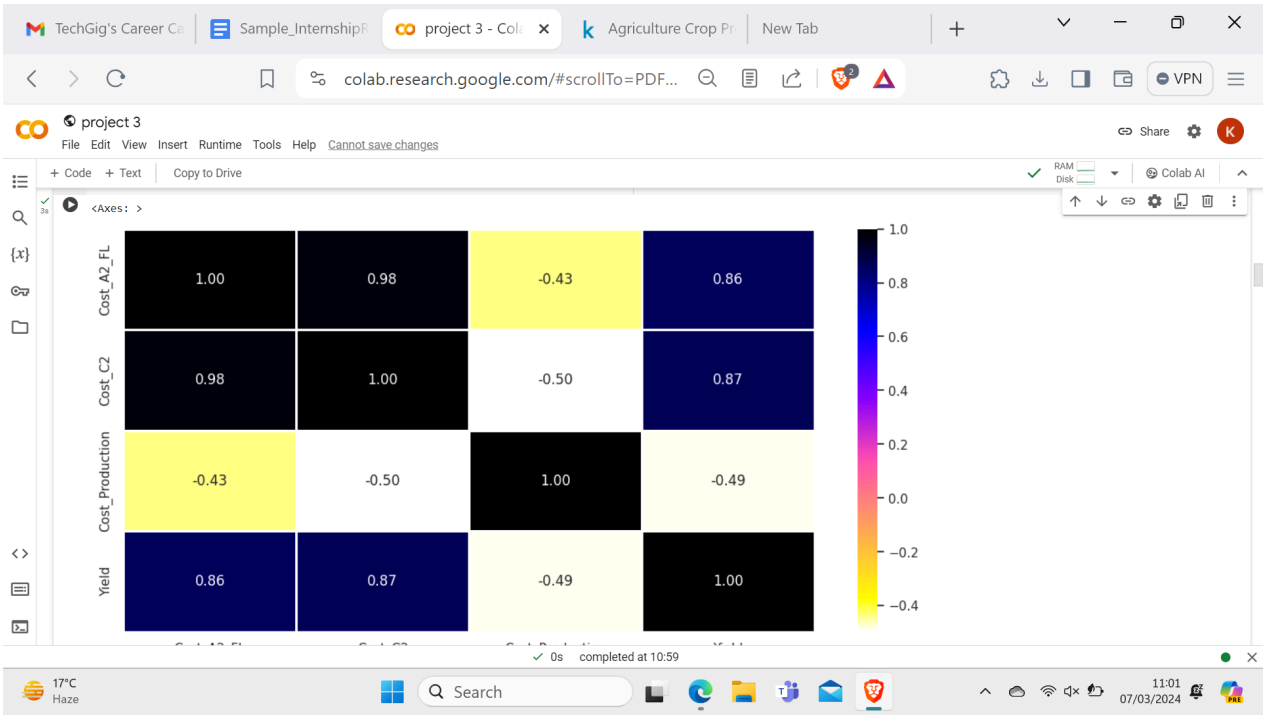

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil
CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'agricuture-crops-production-in-india:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F1935%2F3333%2Fbundle%2Farchive.zip%3FX-Goog-Algorithm%3DGO
KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'
!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)
try:
os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
pass
0s completed at 10:59

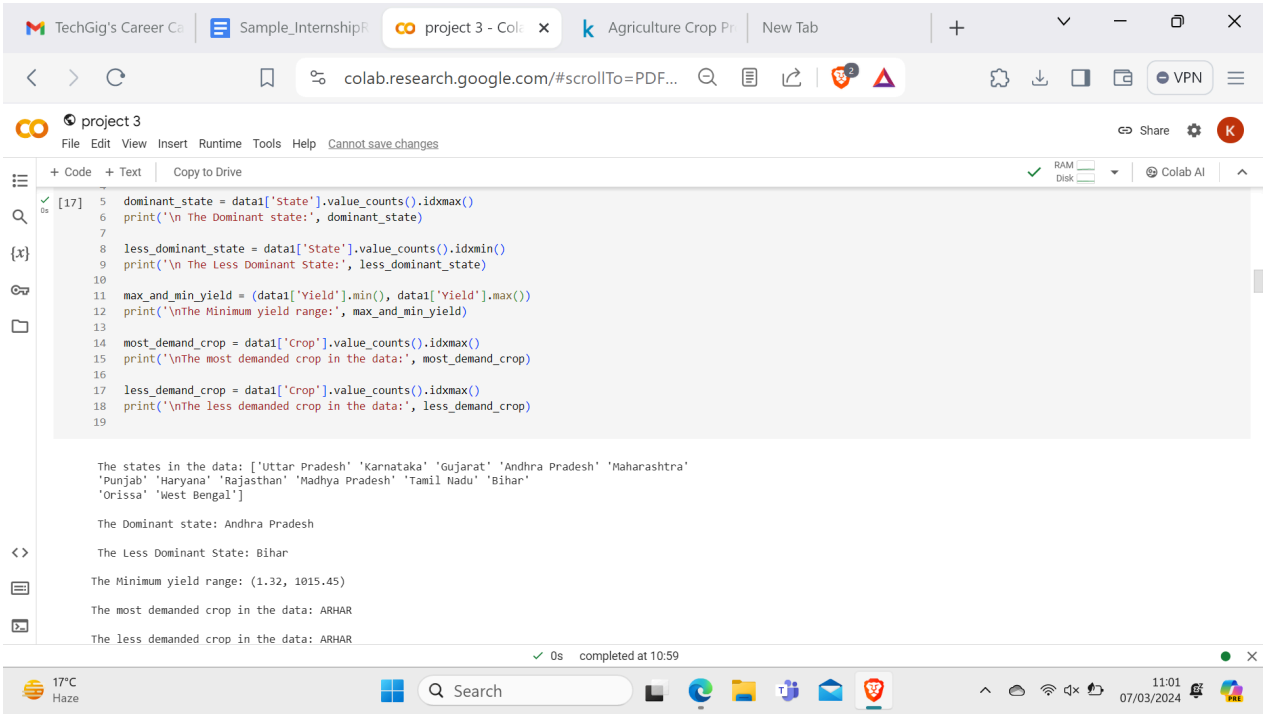

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
try:
os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
pass
for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
directory, download_url_encoded = data_source_mapping.split(':')
download_url = unquote(download_url_encoded)
filename = urlparse(download_url).path
destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
try:
with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
total_length = fileres.headers['content-length']
print(f'Downloading {directory}, {total_length} bytes compressed')
dl = 0
data = fileres.read(CHUNK_SIZE)
while len(data) > 0:
dl += len(data)
tfile.write(data)
done = int(50 * dl / int(total_length))
sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
sys.stdout.flush()
data = fileres.read(CHUNK_SIZE)
if filename.endswith('.zip'):
with ZipFile(tfile) as zfile:
zfile.extractall(destination_path)
else:
with tarfile.open(tfile.name) as tarfile:
tarfile.extractall(destination_path)
0s completed at 10:59

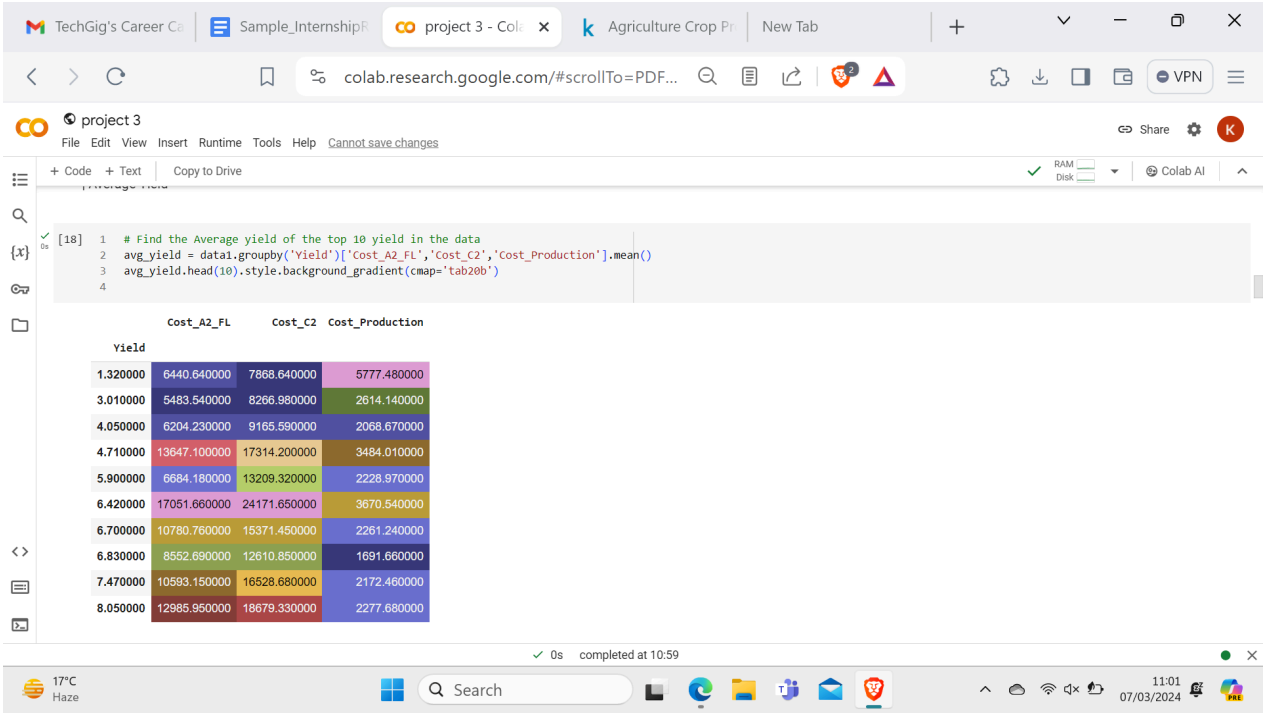

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[3] import os
print(os.listdir('/kaggle/input/agricuture-crops-production-in-india/'))
['datafile (2).csv', 'datafile (1).csv', 'produce.csv', 'datafile (3).csv', 'datafile.csv']
[4] #Load the data using pandas read functions
data1 = pd.read_csv('/kaggle/input/agricuture-crops-production-in-india/datafile (1).csv')
data2 = pd.read_csv('/kaggle/input/agricuture-crops-production-in-india/datafile (2).csv')
data3 = pd.read_csv('/kaggle/input/agricuture-crops-production-in-india/datafile (3).csv')
datafile = pd.read_csv('/kaggle/input/agricuture-crops-production-in-india/datafile.csv')
produce = pd.read_csv('/kaggle/input/agricuture-crops-production-in-india/produce.csv')
[5] data1 = pd.read_csv('/kaggle/input/agricuture-crops-production-in-india/datafile (1).csv')
data1 = data1.rename(columns={
'Crop': 'Crop',
'State': 'State',
'Cost of Cultivation (`/Hectare) A2+FL': 'Cost_A2_FL',
'Cost of Cultivation (`/Hectare) C2': 'Cost_C2',
'Cost of Production (`/Quintal) C2': 'Cost_Production',
'Yield (Quintal/ Hectare) ': 'Yield'
})
data1.head().style.set_properties(**{'background-color':'royalblue','color':'black','border-color':'#8b8c8c'})
0s completed at 10:59


project 3
File Edit View Insert Runtime Tools Help Cannot save changes
[5] data1.head().style.set_properties(**{'background-color':'royalblue','color':'black','border-color':'#8b8c8c'})
Crop State Cost_A2_FL Cost_C2 Cost_Production Yield
0 ARHAR Uttar Pradesh 9794.050000 23076.740000 1941.550000 9.830000
1 ARHAR Karnataka 10593.150000 16528.680000 2172.460000 7.470000
2 ARHAR Gujarat 13468.820000 19551.900000 1898.300000 9.590000
3 ARHAR Andhra Pradesh 17051.660000 24171.650000 3670.540000 6.420000
4 ARHAR Maharashtra 17130.550000 25270.260000 2775.800000 8.720000
[6] #check the shape of the data
print(f' The dataset contains {data1.shape[1]} columns and {data1.shape[0]} rows')
The dataset contains 6 columns and 49 rows
[7] # Print the column names
print(data1.columns)
Index(['Crop', 'State', 'Cost_A2_FL', 'Cost_C2', 'Cost_Production', 'Yield'], dtype='object')
[8] data1.info()
<class 'pandas.core.frame.DataFrame'>
0s completed at 10:59

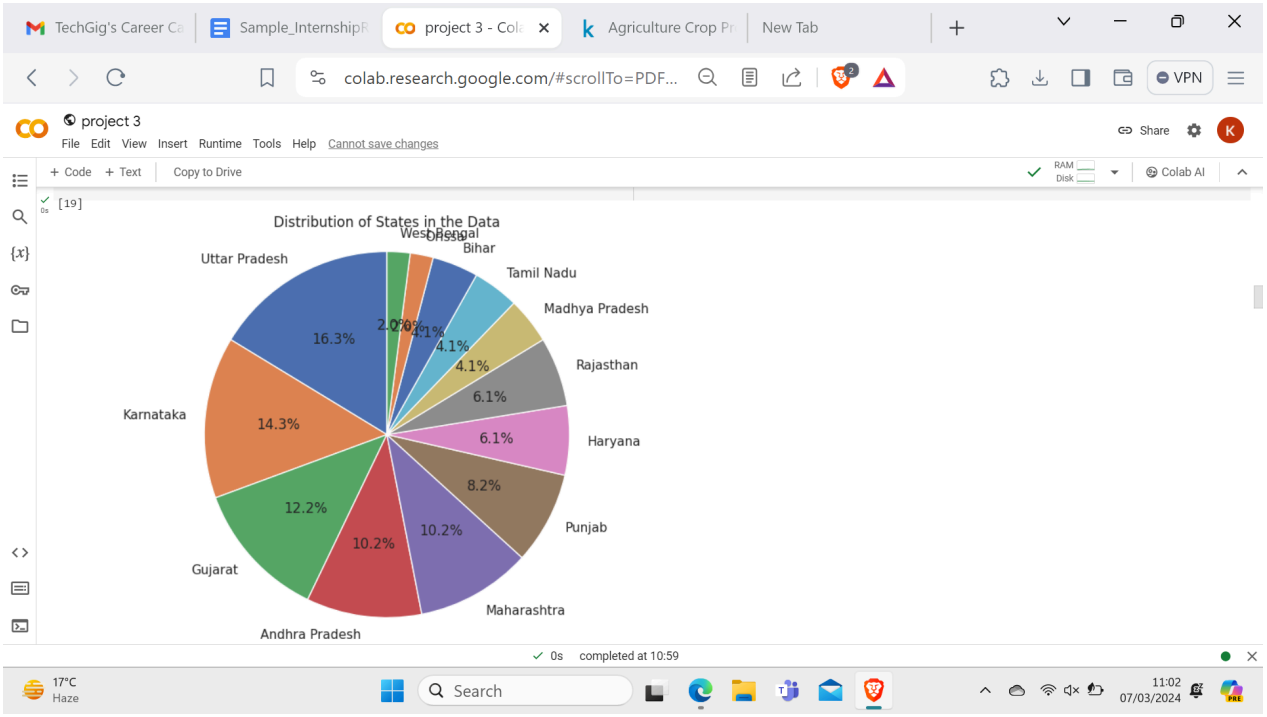

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[12] #Lets check the duplicate values in the data
print('The duplicate values in the data is', data1.duplicated().sum())
The duplicate values in the data is 0
[13] #Check the statistics in the data1
data1.describe().style.background_gradient(cmap='viridis')
Cost_A2_FL Cost_C2 Cost_Production Yield
count 49.000000 49.000000 49.000000 49.000000
mean 20363.537347 31364.666735 1620.537755 98.086735
std 13561.435306 20095.783569 1104.990472 245.293123
min 5483.540000 7868.640000 85.790000 1.320000
25% 12774.410000 19259.840000 732.620000 9.590000
50% 17022.000000 25909.050000 1595.560000 13.700000
75% 24731.060000 35423.480000 2228.970000 36.610000
max 66335.060000 91442.630000 5777.480000 1015.450000
| Histograms
0s completed at 10:59

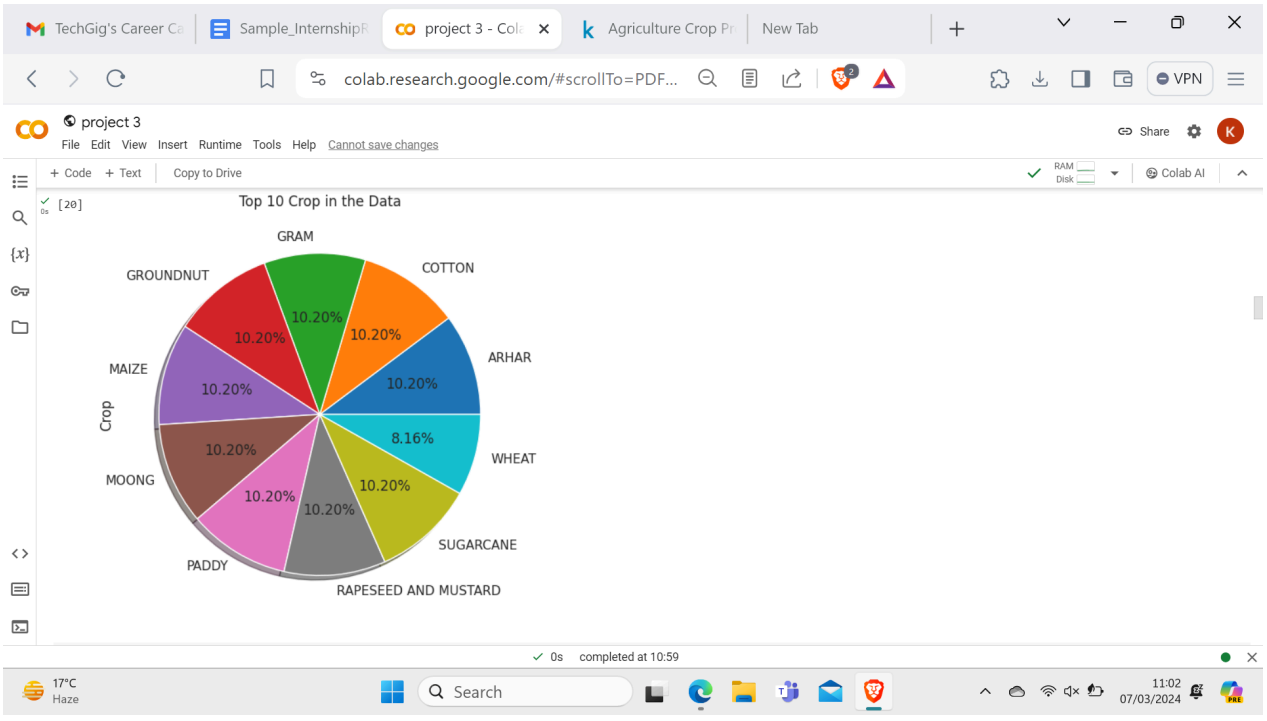

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
<Axes: >
Cost_A2_FL
Cost_C2
Cost_Production
Yield
1.00 0.98 -0.43 0.86
0.98 1.00 -0.50 0.87
-0.43 -0.50 1.00 -0.49
0.86 0.87 -0.49 1.00
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4
0s completed at 10:59

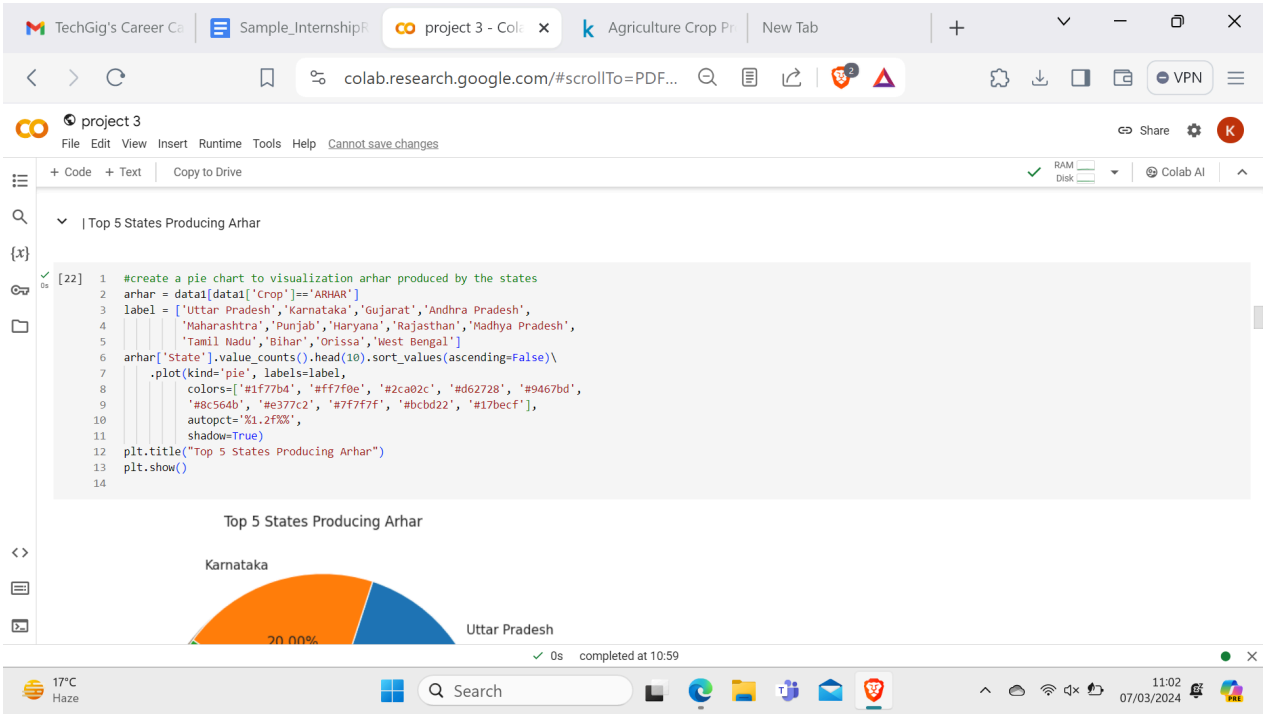


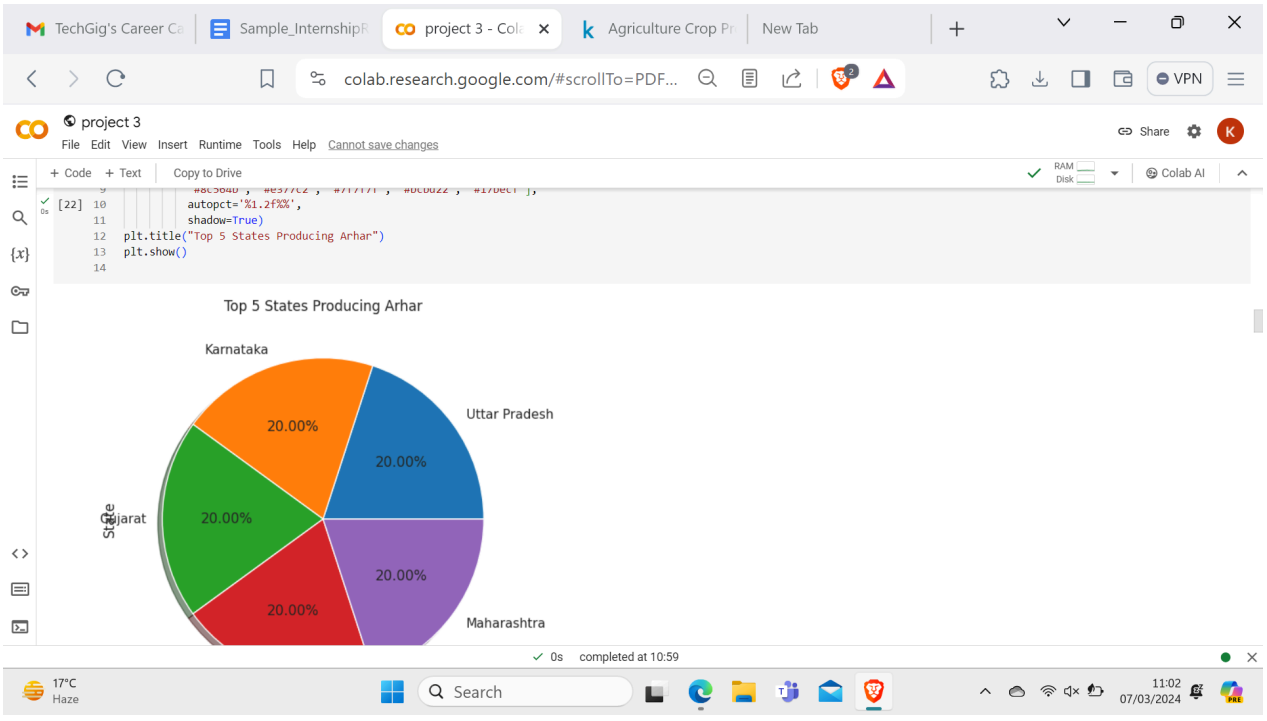


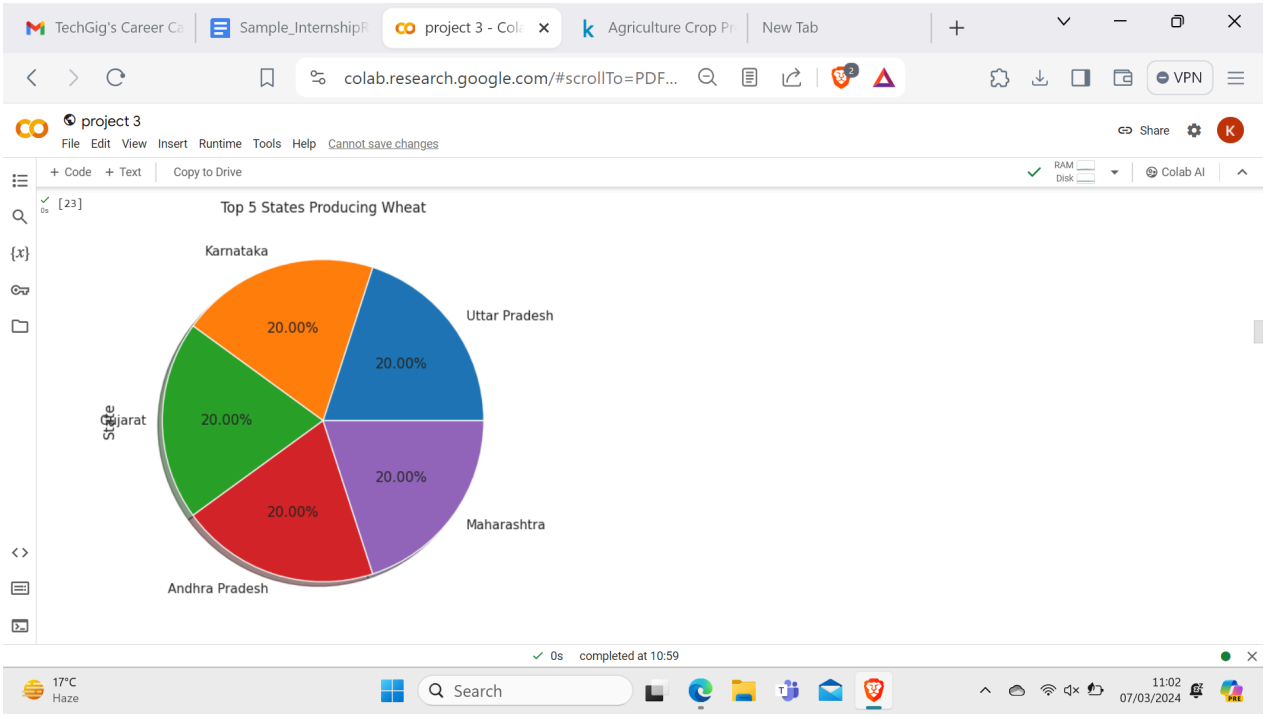
Yield	Cost_A2_FL	Cost_C2	Cost_Production
1.320000	6440.640000	7868.640000	5777.480000
3.010000	5483.540000	8266.980000	2614.140000
4.050000	6204.230000	9165.590000	2068.670000
4.710000	13647.100000	17314.200000	3484.010000
5.900000	6684.180000	13209.320000	2228.970000
6.420000	17051.660000	24171.650000	3670.540000
6.700000	10780.760000	15371.450000	2261.240000
6.830000	8552.690000	12610.850000	1691.660000
7.470000	10593.150000	16528.680000	2172.460000
8.050000	12985.950000	18679.330000	2277.680000

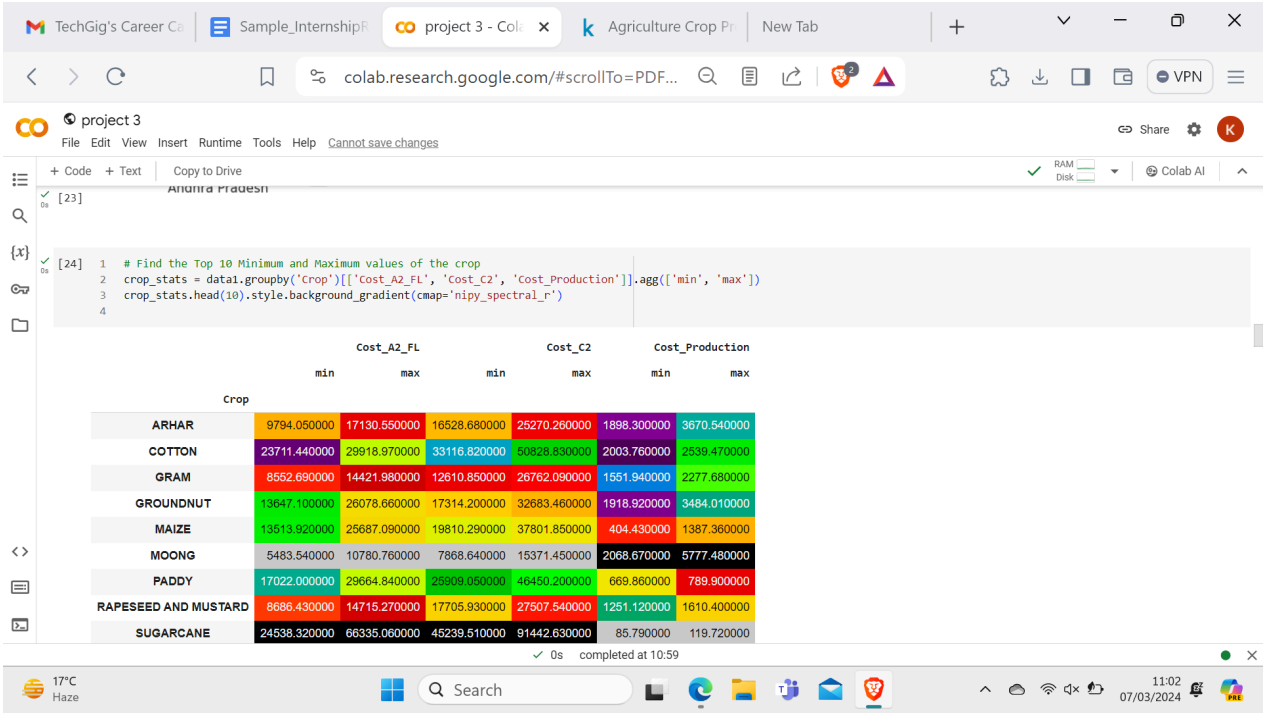

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[19]
Distribution of States in the Data
Uttar Pradesh
Bihar
Tamil Nadu
Madhya Pradesh
Rajasthan
Haryana
Punjab
Maharashtra
Andhra Pradesh
Gujarat
Karnataka
16.3%
4.1%
4.1%
4.1%
6.1%
6.1%
8.2%
10.2%
10.2%
12.2%
14.3%
0s completed at 10:59


project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[20]
Top 10 Crop in the Data
GRAM
GROUNDNUT
COTTON
MAIZE
ARHAR
MOONG
WHEAT
PADDY
SUGARCANE
RAPESEED AND MUSTARD
Crop
10.20%
8.16%
0s completed at 10:59


project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
| Top 5 States Producing Arhar
[22]
#create a pie chart to visualization arhar produced by the states
arhar = data1[data1['Crop']=='ARHAR']
label = ['Uttar Pradesh','Karnataka','Gujarat','Andhra Pradesh',
'Maharashtra','Punjab','Haryana','Rajasthan','Madhya Pradesh',
'Tamil Nadu','Bihar','Orissa','West Bengal']
arhar['State'].value_counts().head(10).sort_values(ascending=False)\
.plot(kind='pie', labels=label,
colors=['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd',
'#8c564b', '#e377c2', '#7f7f7f', '#bcbd22', '#17becf'],
autopct='%1.2f%%',
shadow=True)
plt.title("Top 5 States Producing Arhar")
plt.show()
Top 5 States Producing Arhar
Karnataka
Uttar Pradesh
20.00%
completed at 10:59
17°C
Haze
Search
11:02
07/03/2024

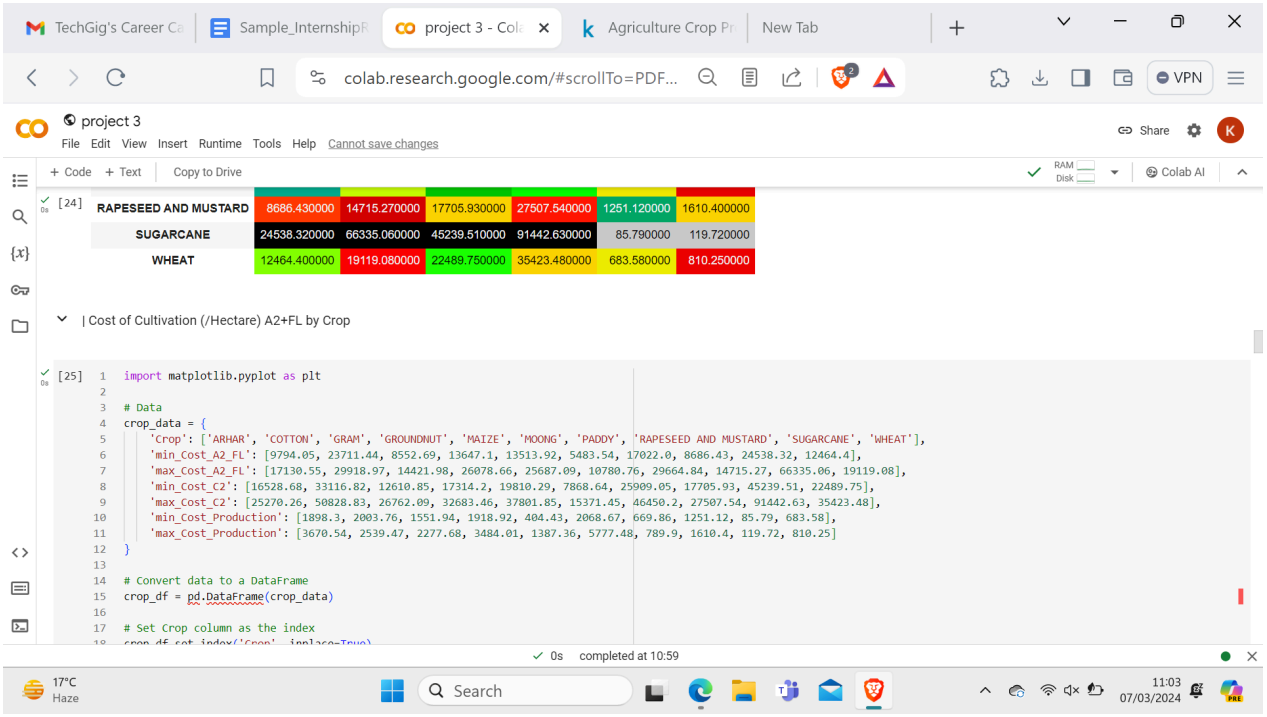

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[22]
autopct='%1.2f%%',
shadow=True)
plt.title("Top 5 States Producing Arhar")
plt.show()
Top 5 States Producing Arhar
Karnataka
Uttar Pradesh
Gujarat
Maharashtra
20.00%
20.00%
20.00%
20.00%
State
completed at 10:59
17°C
Haze
Search
11:02
07/03/2024

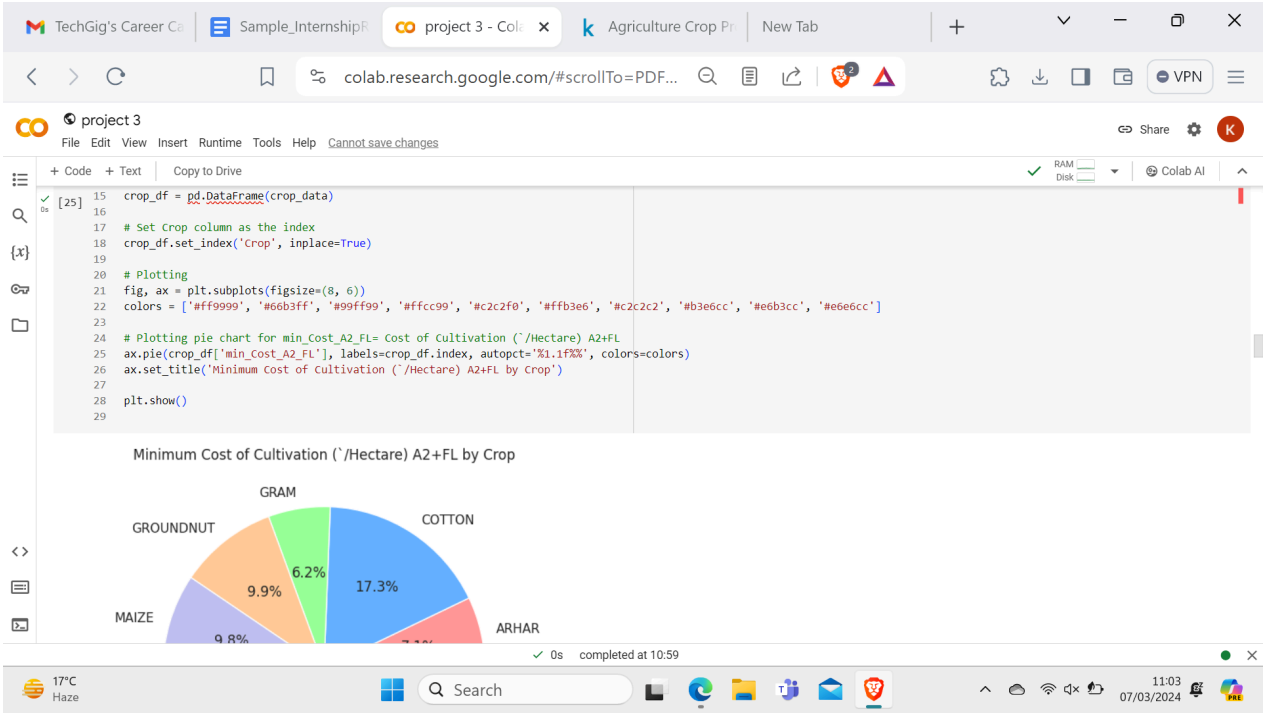


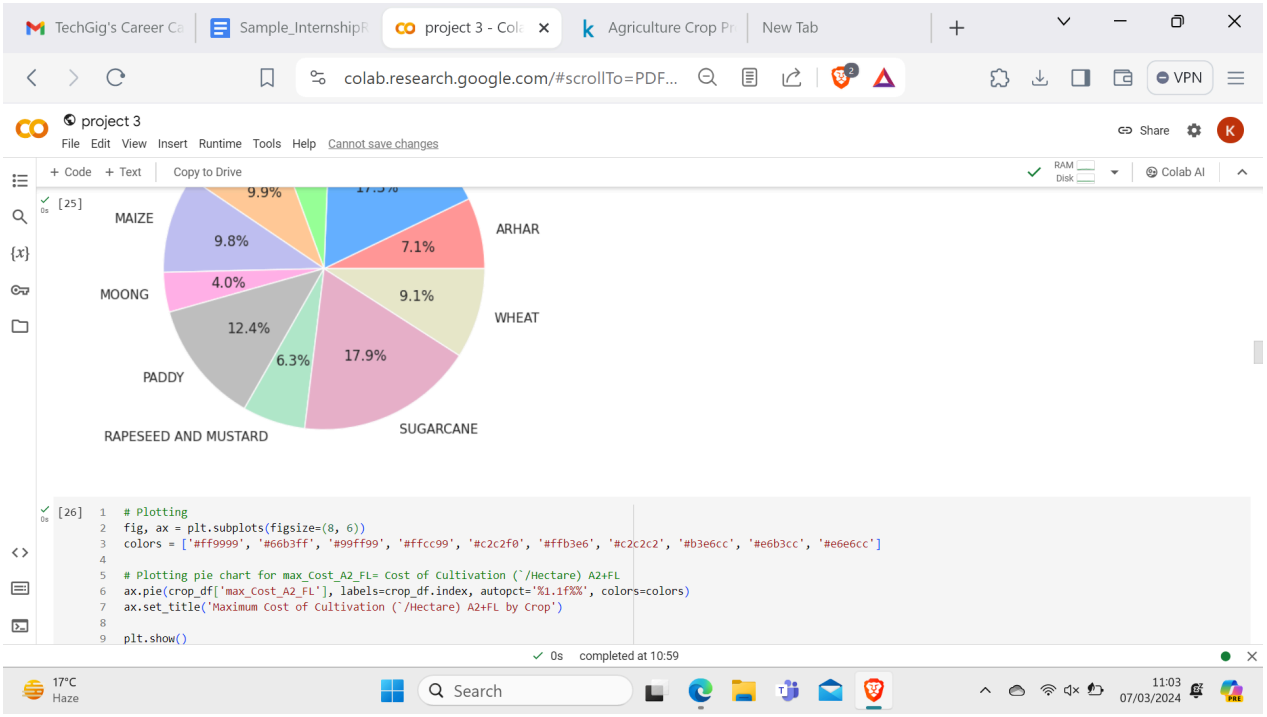


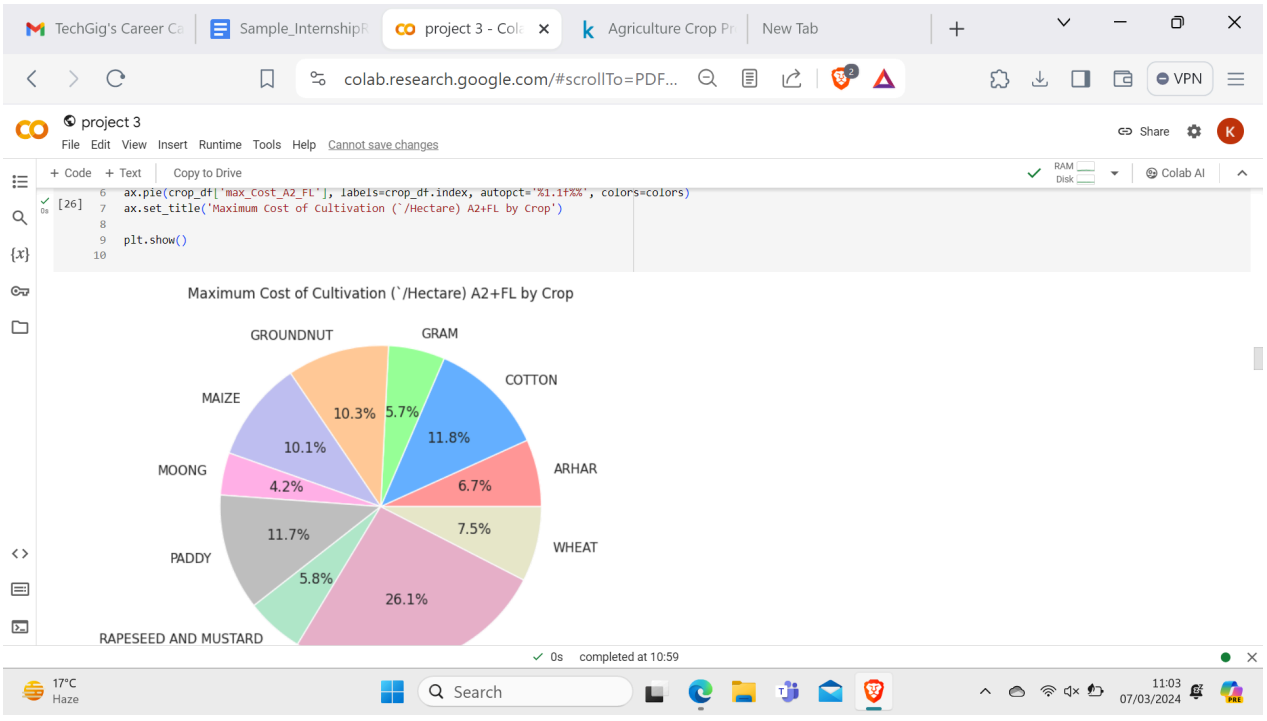
```
# Find the Top 10 Minimum and Maximum values of the crop
crop_stats = data1.groupby('Crop')[['Cost_A2_FL', 'Cost_C2', 'Cost_Production']].agg(['min', 'max'])
crop_stats.head(10).style.background_gradient(cmap='nipy_spectral_r')
```

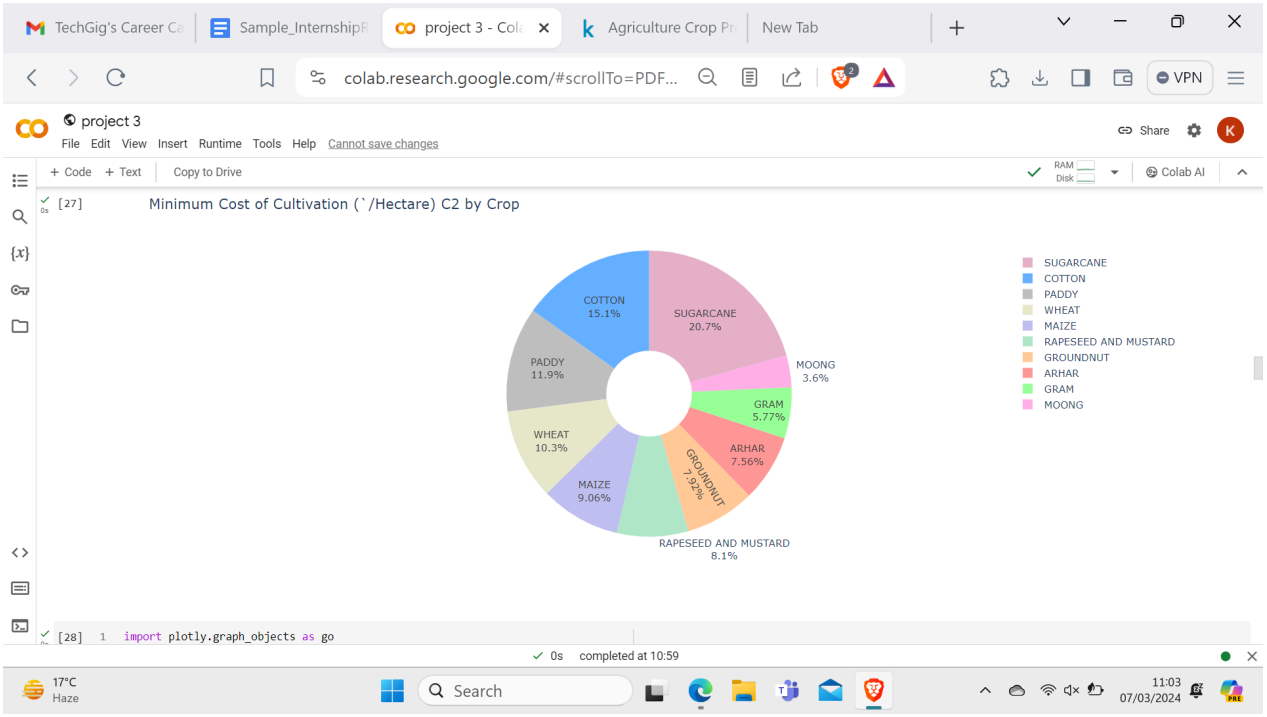
Crop	Cost_A2_FL		Cost_C2		Cost_Production	
	min	max	min	max	min	max
ARHAR	9794.050000	17130.550000	16528.680000	25270.260000	1898.300000	3670.540000
COTTON	23711.440000	29918.970000	33116.820000	50828.830000	2003.760000	2539.470000
GRAM	8552.690000	14421.980000	12610.850000	26762.090000	1551.940000	2277.680000
GROUNDNUT	13647.100000	26078.660000	17314.200000	32683.460000	1918.920000	3484.010000
MAIZE	13513.920000	25687.090000	19810.290000	37801.850000	404.430000	1387.360000
MOONG	5483.540000	10780.760000	7868.640000	15371.450000	2068.670000	5777.480000
PADDY	17022.000000	29664.840000	25909.050000	46450.200000	669.860000	789.900000
RAPESEED AND MUSTARD	8686.430000	14715.270000	17705.930000	27507.540000	1251.120000	1610.400000
SUGARCANE	24538.320000	66335.060000	45239.510000	91442.630000	85.790000	119.720000

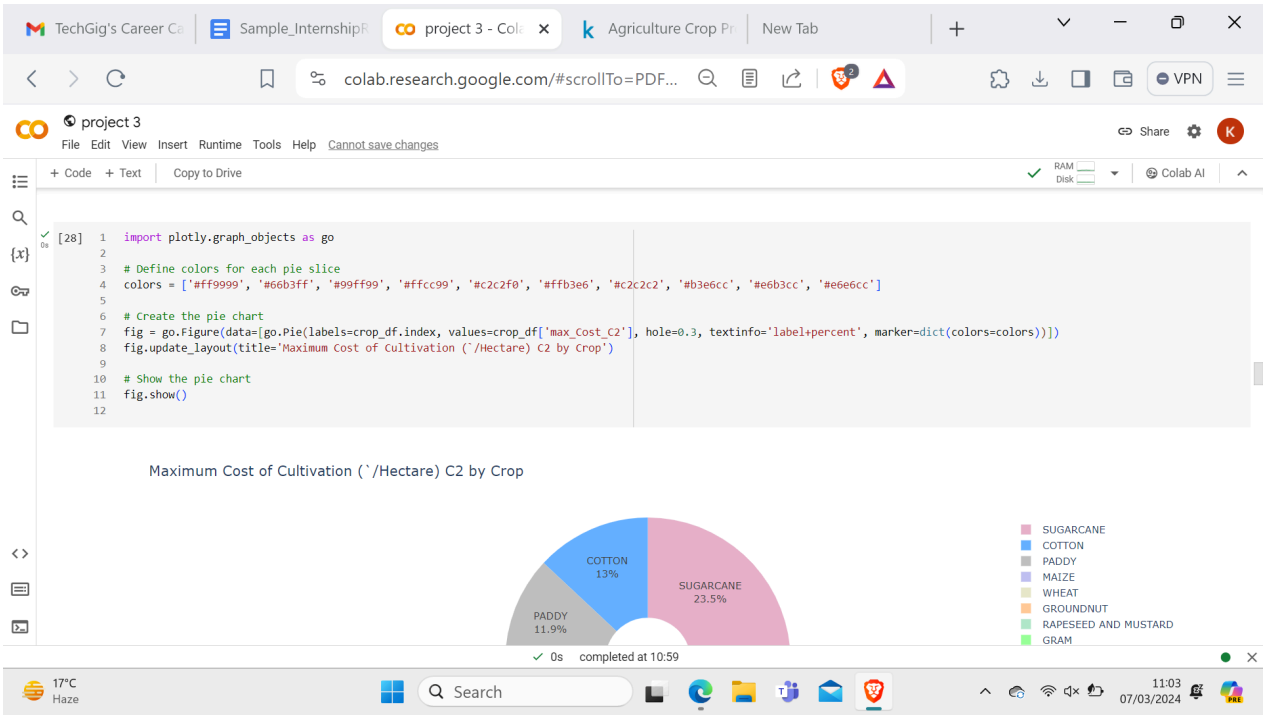

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[24]
RAPESEED AND MUSTARD 8686.430000 14715.270000 17705.930000 27507.540000 1251.120000 1610.400000
SUGARCANE 24538.320000 66335.060000 45239.510000 91442.630000 85.790000 119.720000
WHEAT 12464.400000 19119.080000 22489.750000 35423.480000 683.580000 810.250000
| Cost of Cultivation (/Hectare) A2+FL by Crop
[25]
import matplotlib.pyplot as plt
# Data
crop_data = {
'Crop': ['ARHAR', 'COTTON', 'GRAM', 'GROUNDNUT', 'MAIZE', 'MOONG', 'PADDY', 'RAPESEED AND MUSTARD', 'SUGARCANE', 'WHEAT'],
'min_Cost_A2_FL': [9794.05, 23711.44, 8552.69, 13647.1, 13513.92, 5483.54, 17022.0, 8686.43, 24538.32, 12464.4],
'max_Cost_A2_FL': [17130.55, 29918.97, 14421.98, 26078.66, 25687.09, 10780.76, 29664.84, 14715.27, 66335.06, 19119.08],
'min_Cost_C2': [16528.68, 33116.82, 12610.85, 17314.2, 19810.29, 7868.64, 25909.05, 17705.93, 45239.51, 22489.75],
'max_Cost_C2': [25270.26, 50828.83, 26762.09, 32683.46, 37801.85, 15371.45, 46450.2, 27507.54, 91442.63, 35423.48],
'min_Cost_Production': [1898.3, 2003.76, 1551.94, 1918.92, 404.43, 2068.67, 669.86, 1251.12, 85.79, 683.58],
'max_Cost_Production': [3670.54, 2539.47, 2277.68, 3484.01, 1387.36, 5777.48, 789.9, 1610.4, 119.72, 810.25]
}
# Convert data to a DataFrame
crop_df = pd.DataFrame(crop_data)
# Set Crop column as the index
0s completed at 10:59
17°C Haze
Search
11:03 07/03/2024

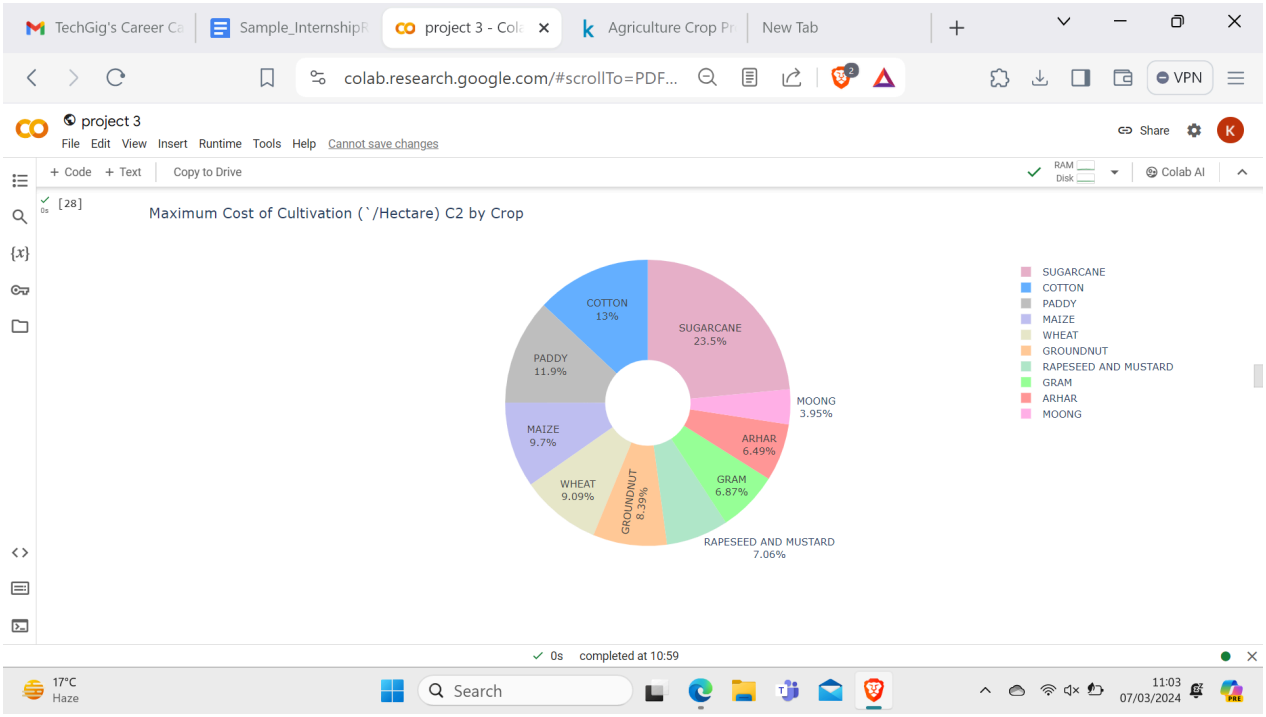

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[25]
crop_df = pd.DataFrame(crop_data)
# Set Crop column as the index
crop_df.set_index('Crop', inplace=True)
# Plotting
fig, ax = plt.subplots(figsize=(8, 6))
colors = ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99', '#c2c2f0', '#ffb3e6', '#c2c2c2', '#b3e6cc', '#e6b3cc', '#e6e6cc']
# Plotting pie chart for min_Cost_A2_FL= Cost of Cultivation (`/Hectare) A2+FL
ax.pie(crop_df['min_Cost_A2_FL'], labels=crop_df.index, autopct='%1.1f%%', colors=colors)
ax.set_title('Minimum Cost of Cultivation (`/Hectare) A2+FL by Crop')
plt.show()
Minimum Cost of Cultivation (`/Hectare) A2+FL by Crop
GRAM
GROUNDNUT
COTTON
6.2%
9.9%
17.3%
MAIZE
ARHAR
9.8%
0s completed at 10:59
17°C Haze
Search
11:03 07/03/2024

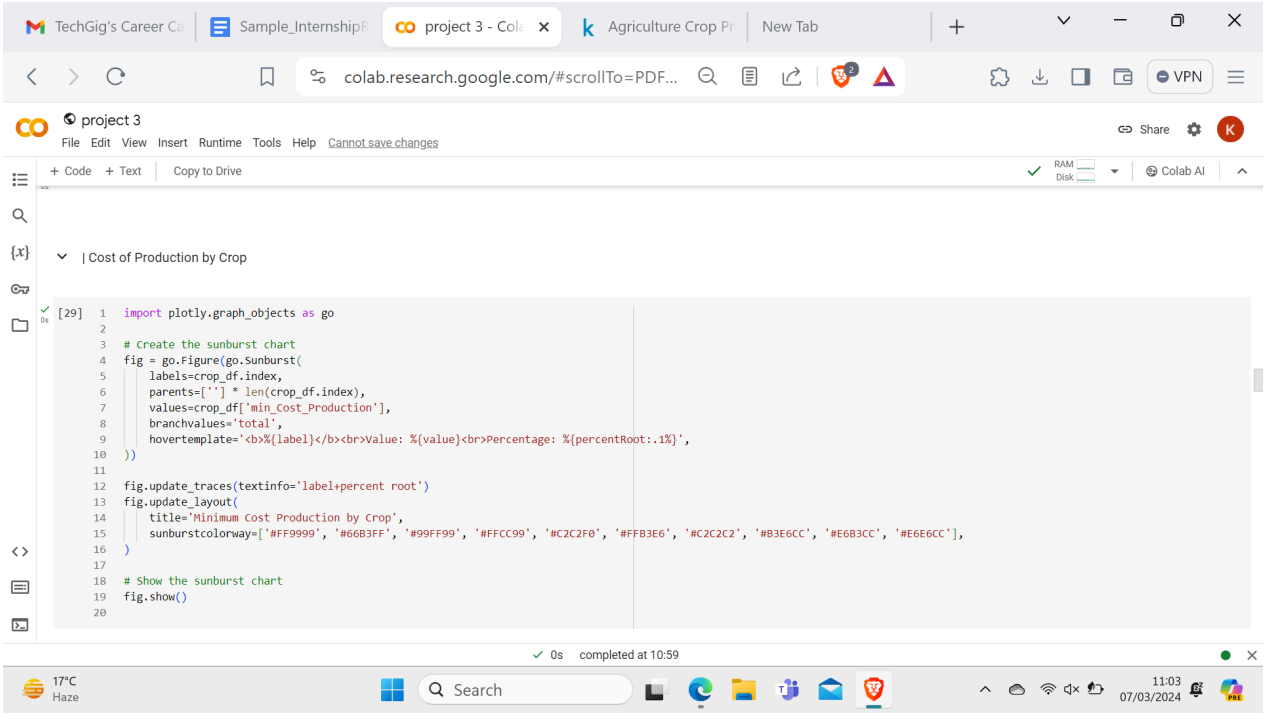

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[25]
MAIZE
9.8%
ARHAR
7.1%
MOONG
4.0%
9.1%
WHEAT
12.4%
6.3%
17.9%
PADDY
RAPESEED AND MUSTARD
SUGARCANE
[26]
Plotting
fig, ax = plt.subplots(figsize=(8, 6))
colors = ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99', '#c2c2f0', '#ffb3e6', '#c2c2c2', '#b3e6cc', '#e6b3cc', '#e6e6cc']
# Plotting pie chart for max_Cost_A2_FL= Cost of Cultivation (`/Hectare) A2+FL
ax.pie(crop_df['max_Cost_A2_FL'], labels=crop_df.index, autopct='%1.1f%%', colors=colors)
ax.set_title('Maximum Cost of Cultivation (`/Hectare) A2+FL by Crop')
plt.show()
0s completed at 10:59

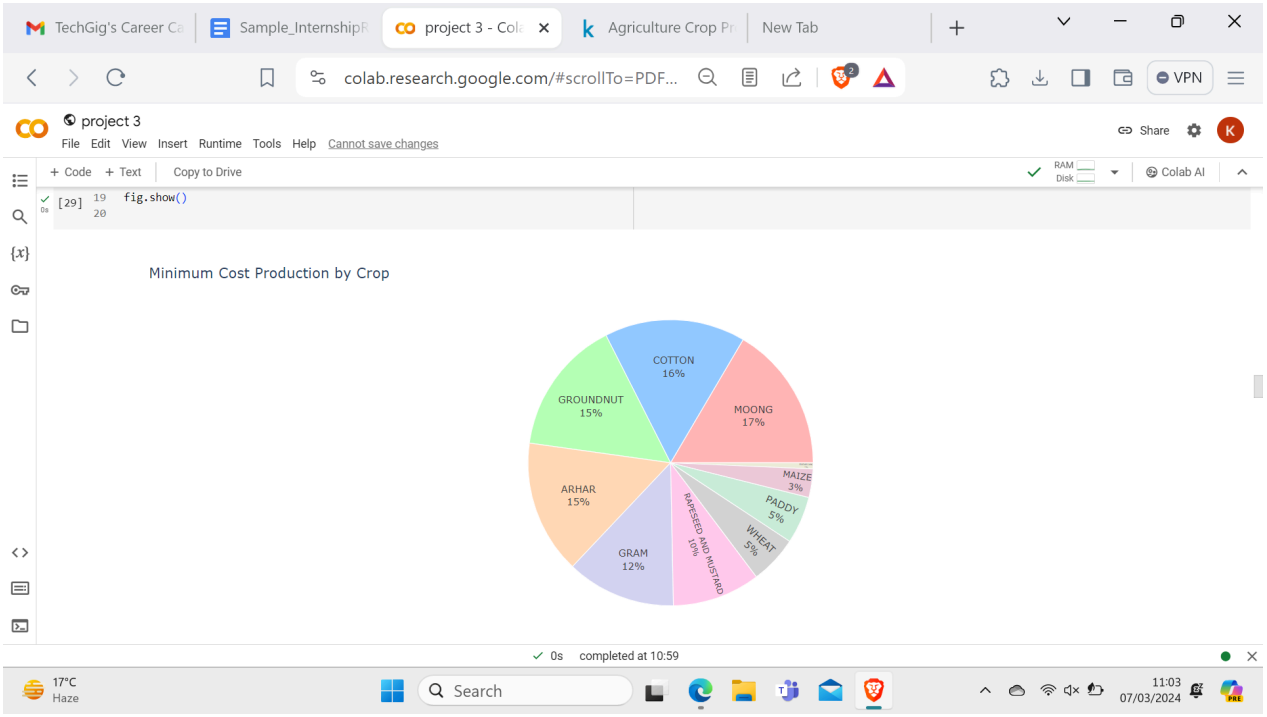

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[26]
ax.pie(crop_df['max_Cost_A2_FL'], labels=crop_df.index, autopct='%1.1f%%', colors=colors)
ax.set_title('Maximum Cost of Cultivation (`/Hectare) A2+FL by Crop')
plt.show()
Maximum Cost of Cultivation (`/Hectare) A2+FL by Crop
GROUNDNUT
GRAM
COTTON
MAIZE
10.3%
5.7%
11.8%
10.1%
MOONG
ARHAR
4.2%
6.7%
7.5%
11.7%
PADDY
WHEAT
5.8%
26.1%
RAPESEED AND MUSTARD
0s completed at 10:59

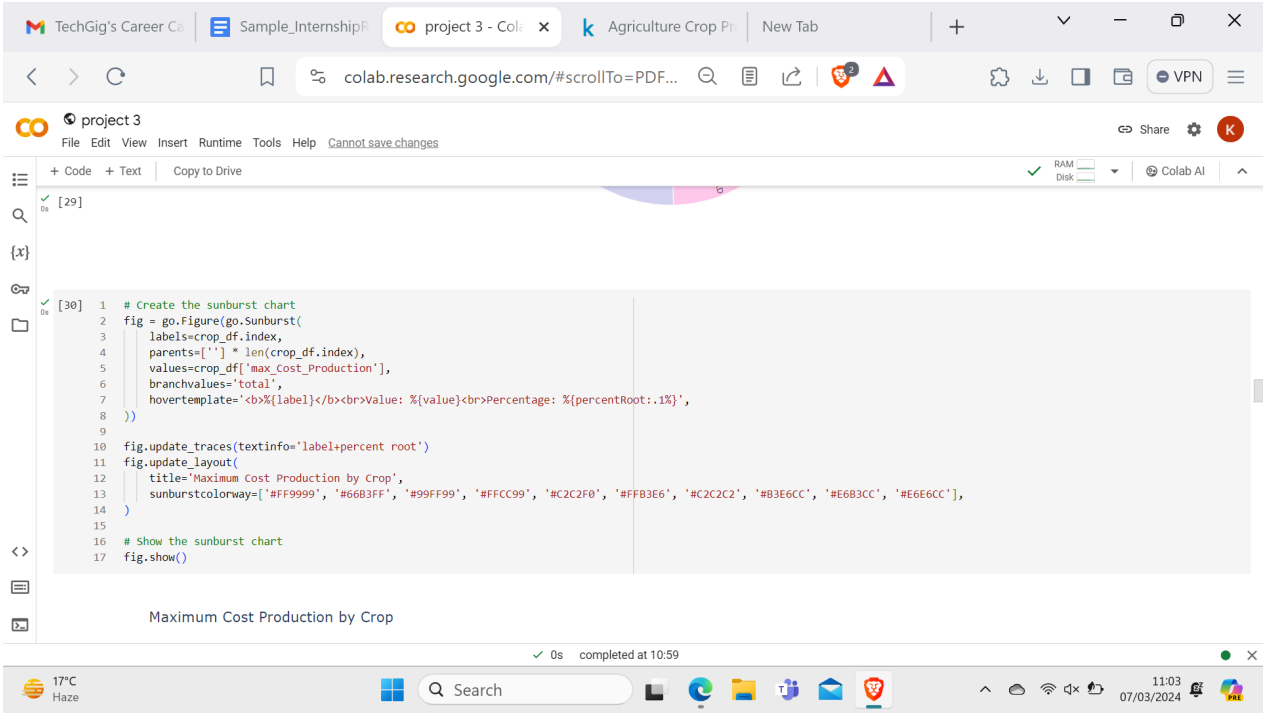

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[27]
Minimum Cost of Cultivation (`/Hectare) C2 by Crop
SUGARCANE 20.7%
COTTON 15.1%
PADDY 11.9%
WHEAT 10.3%
MAIZE 9.06%
RAPESEED AND MUSTARD 8.1%
GROUNDNUT 7.92%
ARHAR 7.56%
GRAM 5.77%
MOONG 3.6%
SUGARCANE
COTTON
PADDY
WHEAT
MAIZE
RAPESEED AND MUSTARD
GROUNDNUT
ARHAR
GRAM
MOONG
[28] 1 import plotly.graph_objects as go
0s completed at 10:59

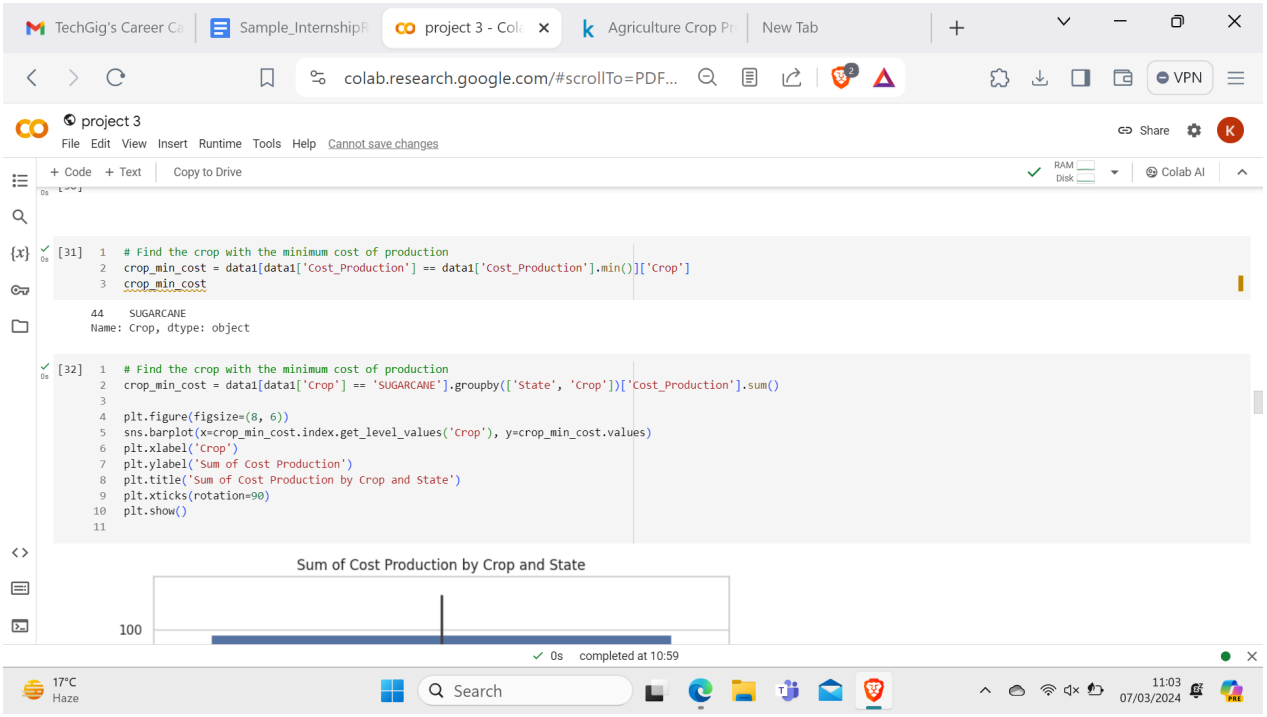

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[28]
import plotly.graph_objects as go
# Define colors for each pie slice
colors = ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99', '#c2c2f0', '#ffb3e6', '#c2c2c2', '#b3e6cc', '#e6b3cc', '#e6e6cc']
# Create the pie chart
fig = go.Figure(data=[go.Pie(labels=crop_df.index, values=crop_df['max_Cost_C2'], hole=0.3, textinfo='label+percent', marker=dict(colors=colors))])
fig.update_layout(title='Maximum Cost of Cultivation (`/Hectare) C2 by Crop')
# Show the pie chart
fig.show()
Maximum Cost of Cultivation (`/Hectare) C2 by Crop
COTTON 13%
SUGARCANE 23.5%
PADDY 11.9%
SUGARCANE
COTTON
PADDY
MAIZE
WHEAT
GROUNDNUT
RAPESEED AND MUSTARD
GRAM
0s completed at 10:59

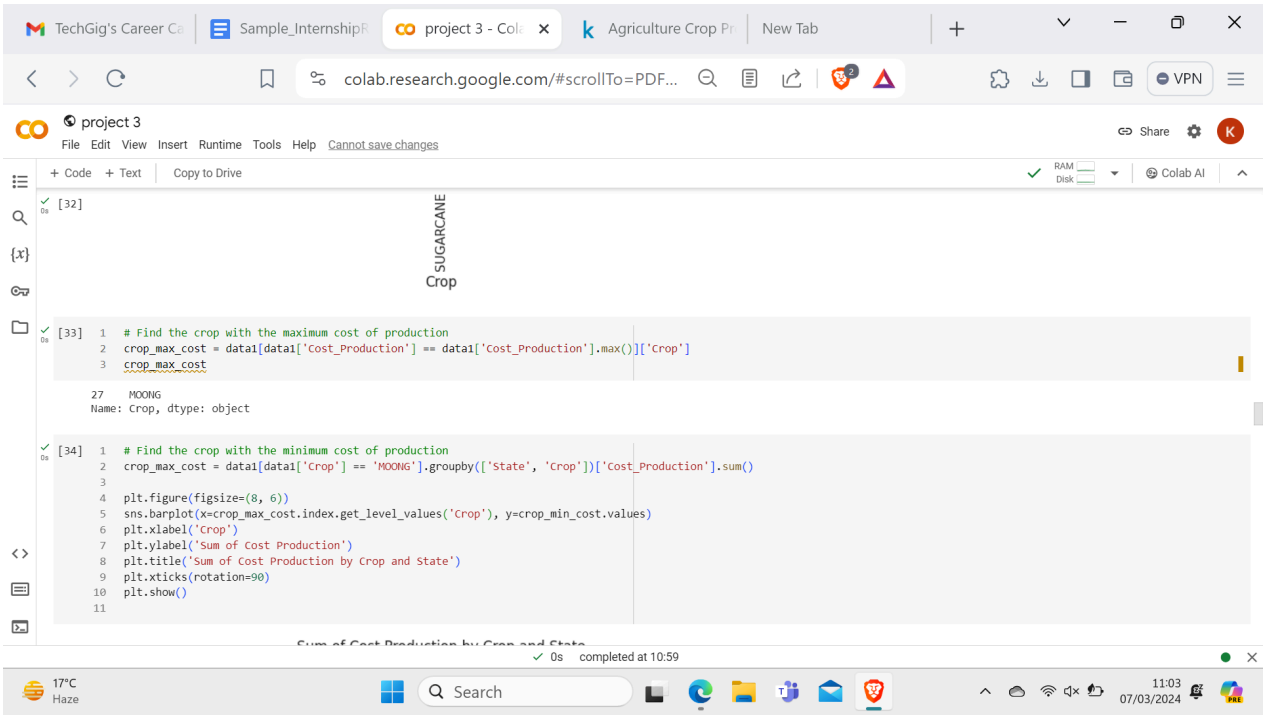

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[28]
Maximum Cost of Cultivation (`/Hectare) C2 by Crop
COTTON 13%
SUGARCANE 23.5%
PADDY 11.9%
MOONG 3.95%
MAIZE 9.7%
ARHAR 6.49%
WHEAT 9.09%
GROUNDNUT 8.39%
GRAM 6.87%
RAPESEED AND MUSTARD 7.06%
SUGARCANE
COTTON
PADDY
MAIZE
WHEAT
GROUNDNUT
RAPESEED AND MUSTARD
GRAM
ARHAR
MOONG
0s completed at 10:59

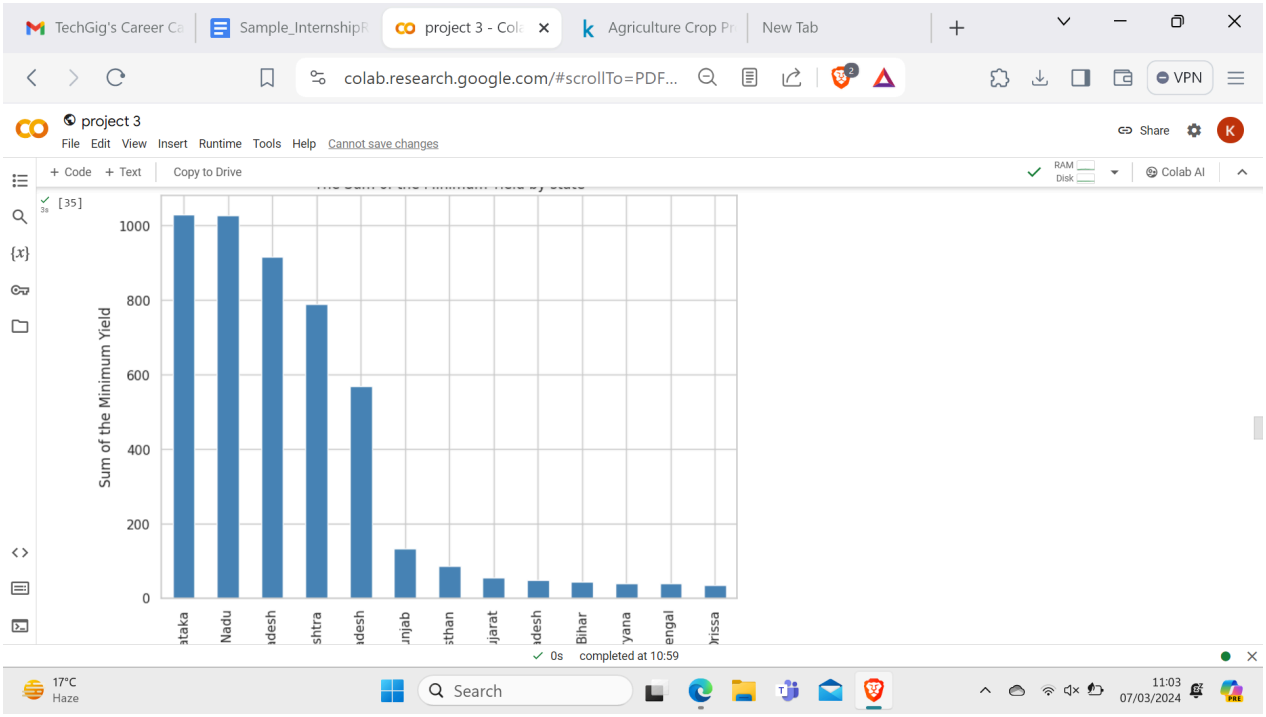

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
| Cost of Production by Crop
[29]
import plotly.graph_objects as go
# Create the sunburst chart
fig = go.Figure(go.Sunburst(
labels=crop_df.index,
parents=[''] * len(crop_df.index),
values=crop_df['min_Cost_Production'],
branchvalues='total',
hovertemplate='%{label}
Value: %{value}
Percentage: %{percentRoot:.1%}',
))
fig.update_traces(textinfo='label+percent root')
fig.update_layout(
title='Minimum Cost Production by Crop',
sunburstcolorway=['#FF9999', '#66B3FF', '#99FF99', '#FFCC99', '#C2C2F0', '#FFB3E6', '#C2C2C2', '#B3E6CC', '#E6B3CC', '#E6E6CC'],
)
# Show the sunburst chart
fig.show()
0s completed at 10:59

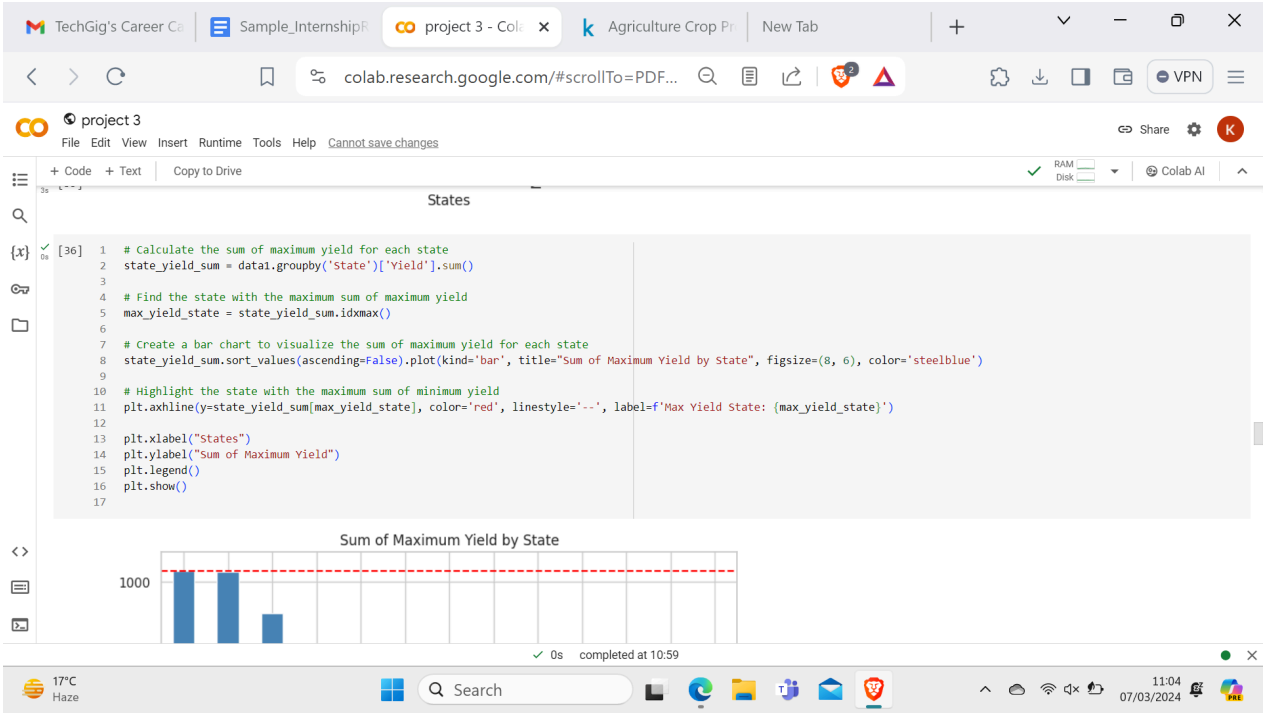

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[29] fig.show()
Minimum Cost Production by Crop
COTTON 16%
MOONG 17%
GROUNDNUT 15%
ARHAR 15%
MAIZE 3%
PADDY 5%
WHEAT 5%
RAPESEED AND MUSTARD 10%
GRAM 12%
0s completed at 10:59

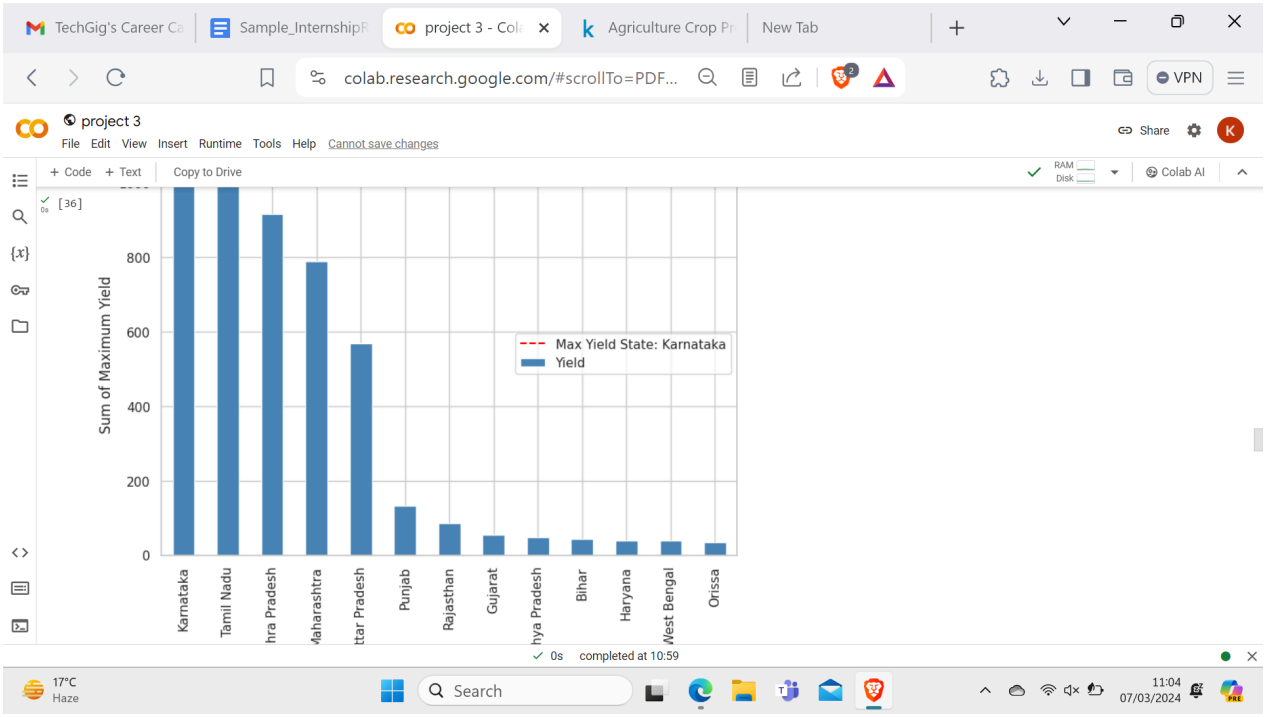

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[29]
[30]
# Create the sunburst chart
fig = go.Figure(go.Sunburst(
labels=crop_df.index,
parents=[''] * len(crop_df.index),
values=crop_df['max_Cost_Production'],
branchvalues='total',
hovertemplate='%{label}
Value: %{value}
Percentage: %{percentRoot:.1%}',
))
fig.update_traces(textinfo='label+percent root')
fig.update_layout(
title='Maximum Cost Production by Crop',
sunburstcolorway=['#FF9999', '#66B3FF', '#99FF99', '#FFCC99', '#C2C2F0', '#FFB3E6', '#C2C2C2', '#B3E6CC', '#E6B3CC', '#E6E6CC'],
)
# Show the sunburst chart
fig.show()
Maximum Cost Production by Crop
0s completed at 10:59

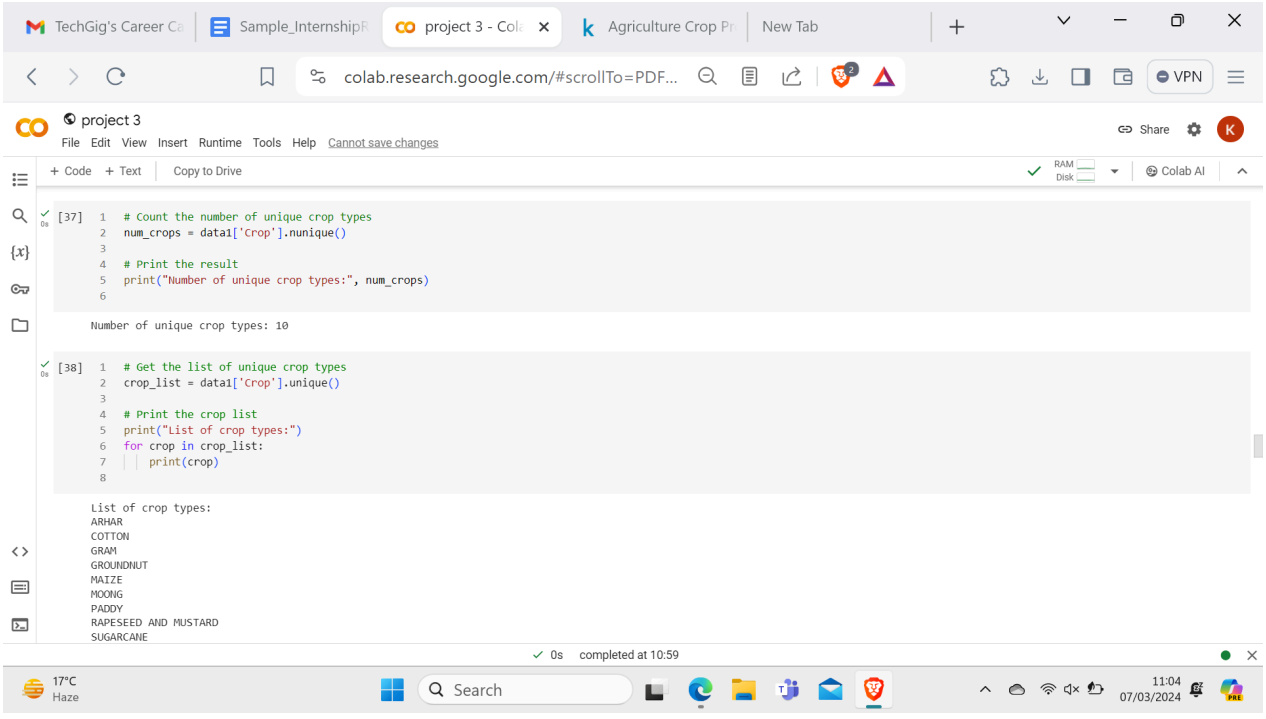

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[31] # Find the crop with the minimum cost of production
crop_min_cost = data1[data1['Cost_Production'] == data1['Cost_Production'].min()]['Crop']
crop_min_cost
44 SUGARCANE
Name: Crop, dtype: object
[32] # Find the crop with the minimum cost of production
crop_min_cost = data1[data1['Crop'] == 'SUGARCANE'].groupby(['State', 'Crop'])['Cost_Production'].sum()
plt.figure(figsize=(8, 6))
sns.barplot(x=crop_min_cost.index.get_level_values('Crop'), y=crop_min_cost.values)
plt.xlabel('Crop')
plt.ylabel('Sum of Cost Production')
plt.title('Sum of Cost Production by Crop and State')
plt.xticks(rotation=90)
plt.show()
Sum of Cost Production by Crop and State
100
0s completed at 10:59

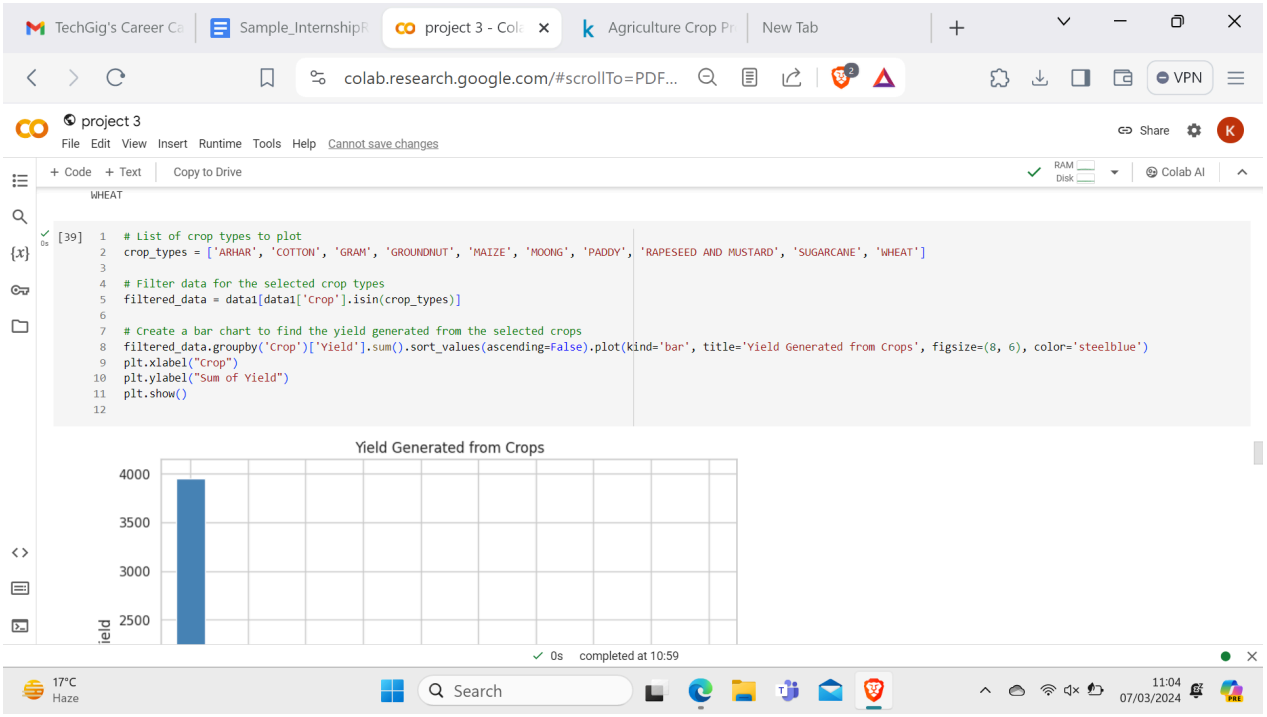

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[32]
SUGARCANE
Crop
[33] # Find the crop with the maximum cost of production
crop_max_cost = data1[data1['Cost_Production'] == data1['Cost_Production'].max()]['Crop']
crop_max_cost
27 MOONG
Name: Crop, dtype: object
[34] # Find the crop with the minimum cost of production
crop_max_cost = data1[data1['Crop'] == 'MOONG'].groupby(['State', 'Crop'])['Cost_Production'].sum()
plt.figure(figsize=(8, 6))
sns.barplot(x=crop_max_cost.index.get_level_values('Crop'), y=crop_min_cost.values)
plt.xlabel('Crop')
plt.ylabel('Sum of Cost Production')
plt.title('Sum of Cost Production by Crop and State')
plt.xticks(rotation=90)
plt.show()
0s completed at 10:59

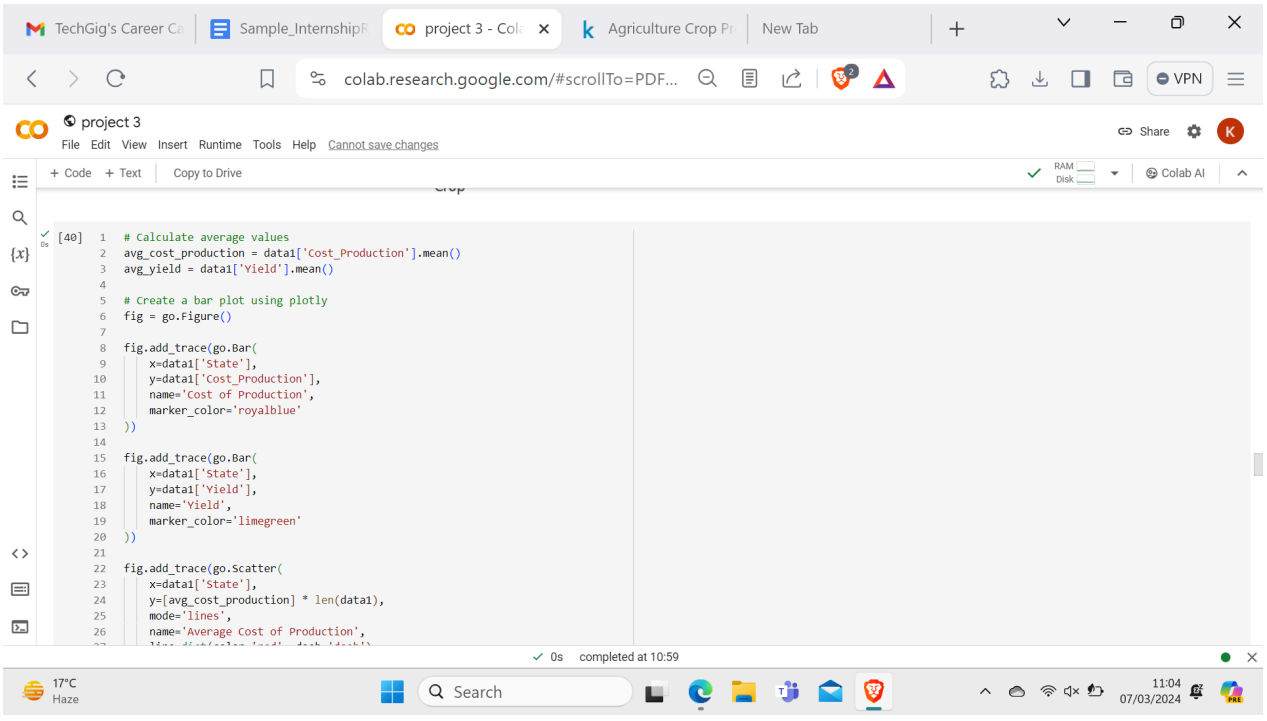

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[35]
Sum of the Minimum Yield
1000
800
600
400
200
0
Nadu
Bihar
completed at 10:59
11:03
07/03/2024

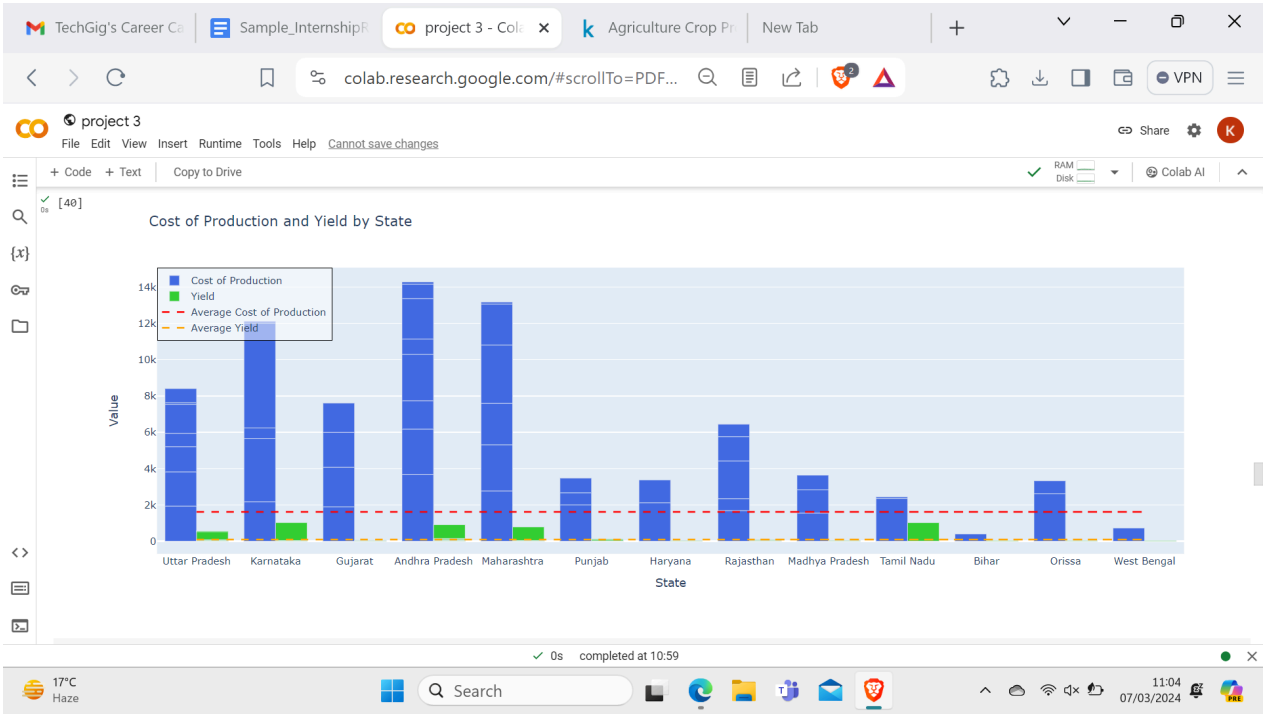

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
States
[36]
# Calculate the sum of maximum yield for each state
state_yield_sum = data1.groupby('State')['Yield'].sum()
# Find the state with the maximum sum of maximum yield
max_yield_state = state_yield_sum.idxmax()
# Create a bar chart to visualize the sum of maximum yield for each state
state_yield_sum.sort_values(ascending=False).plot(kind='bar', title="Sum of Maximum Yield by State", figsize=(8, 6), color='steelblue')
# Highlight the state with the maximum sum of minimum yield
plt.axhline(y=state_yield_sum[max_yield_state], color='red', linestyle='--', label=f'Max Yield State: {max_yield_state}')
plt.xlabel("States")
plt.ylabel("Sum of Maximum Yield")
plt.legend()
plt.show()
Sum of Maximum Yield by State
1000
completed at 10:59
11:04
07/03/2024

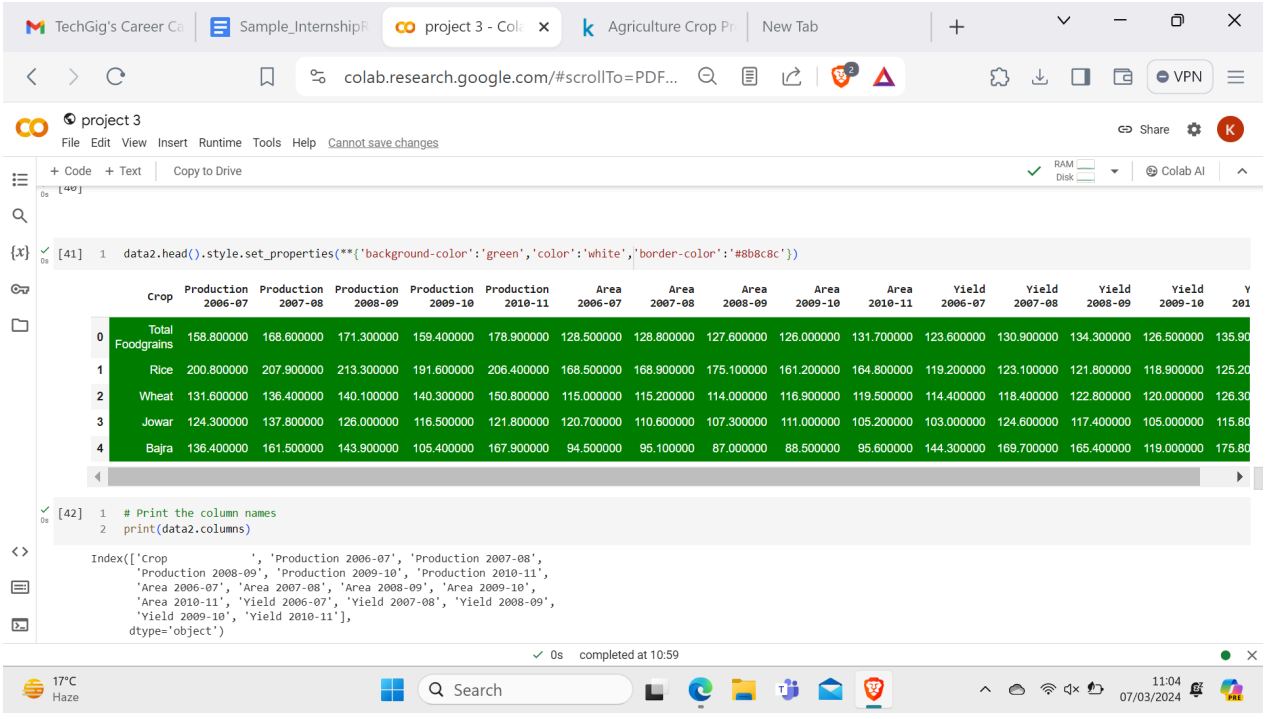

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[36]
Sum of Maximum Yield
800
600
400
200
0
Max Yield State: Karnataka
Yield
Karnataka
Tamil Nadu
Punjab
Rajasthan
Gujarat
Bihar
Haryana
Orissa
0s completed at 10:59


project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[37]
# Count the number of unique crop types
num_crops = data1['Crop'].nunique()
# Print the result
print("Number of unique crop types:", num_crops)
Number of unique crop types: 10
[38]
# Get the list of unique crop types
crop_list = data1['Crop'].unique()
# Print the crop list
print("List of crop types:")
for crop in crop_list:
print(crop)
List of crop types:
ARHAR
COTTON
GRAM
GROUNDNUT
MAIZE
MOONG
PADDY
RAPESEED AND MUSTARD
SUGARCANE
0s completed at 10:59

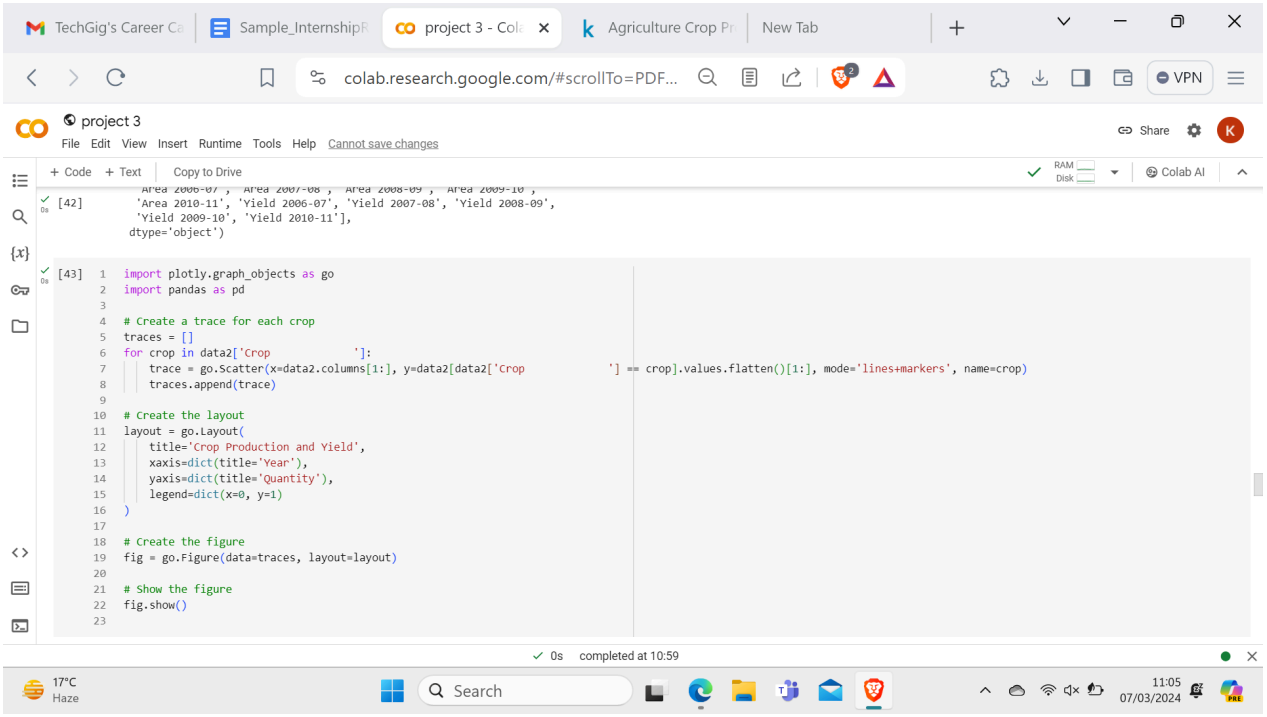

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
WHEAT
[39]
# List of crop types to plot
crop_types = ['ARHAR', 'COTTON', 'GRAM', 'GROUNDNUT', 'MAIZE', 'MOONG', 'PADDY', 'RAPESEED AND MUSTARD', 'SUGARCANE', 'WHEAT']
# Filter data for the selected crop types
filtered_data = data1[data1['Crop'].isin(crop_types)]
# Create a bar chart to find the yield generated from the selected crops
filtered_data.groupby('Crop')['Yield'].sum().sort_values(ascending=False).plot(kind='bar', title='Yield Generated from Crops', figsize=(8, 6), color='steelblue')
plt.xlabel("Crop")
plt.ylabel("Sum of Yield")
plt.show()
Yield Generated from Crops
4000
3500
3000
2500
0s completed at 10:59

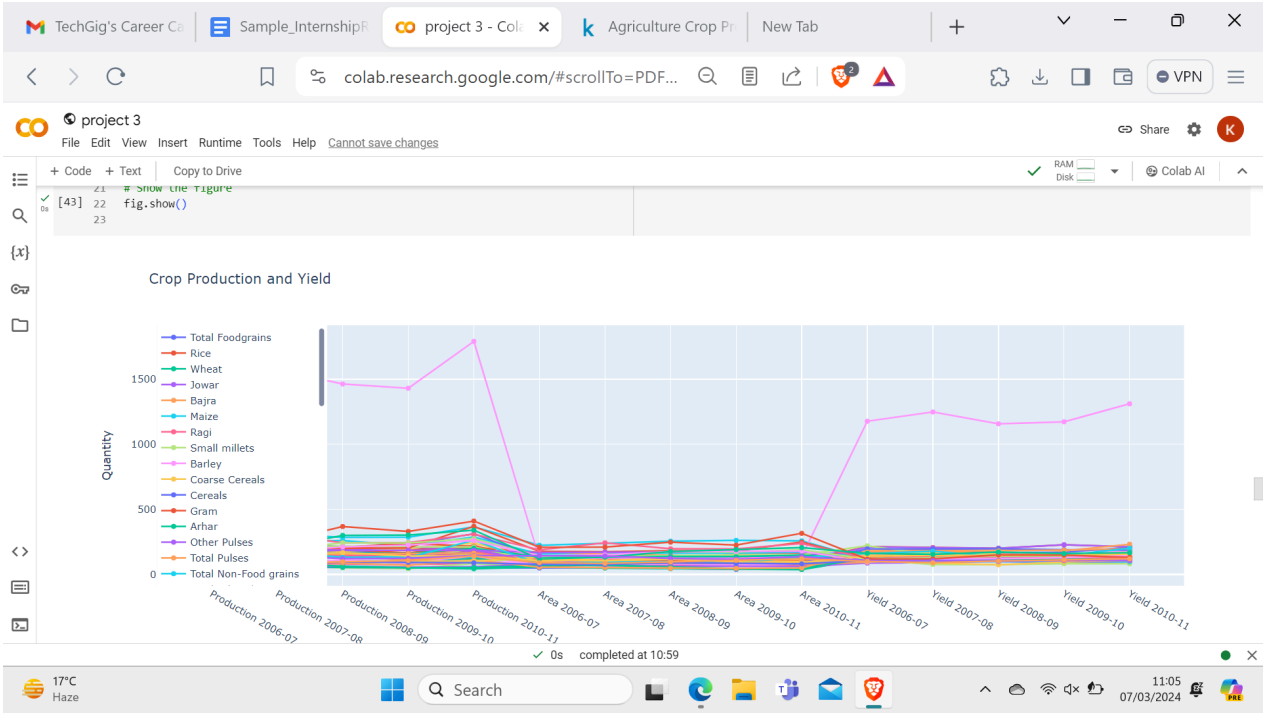

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[40]
# Calculate average values
avg_cost_production = data1['Cost_Production'].mean()
avg_yield = data1['Yield'].mean()
# Create a bar plot using plotly
fig = go.Figure()
fig.add_trace(go.Bar(
x=data1['State'],
y=data1['Cost_Production'],
name='Cost of Production',
marker_color='royalblue'
))
fig.add_trace(go.Bar(
x=data1['State'],
y=data1['Yield'],
name='Yield',
marker_color='limegreen'
))
fig.add_trace(go.Scatter(
x=data1['State'],
y=[avg_cost_production] * len(data1),
mode='lines',
name='Average Cost of Production',
0s completed at 10:59

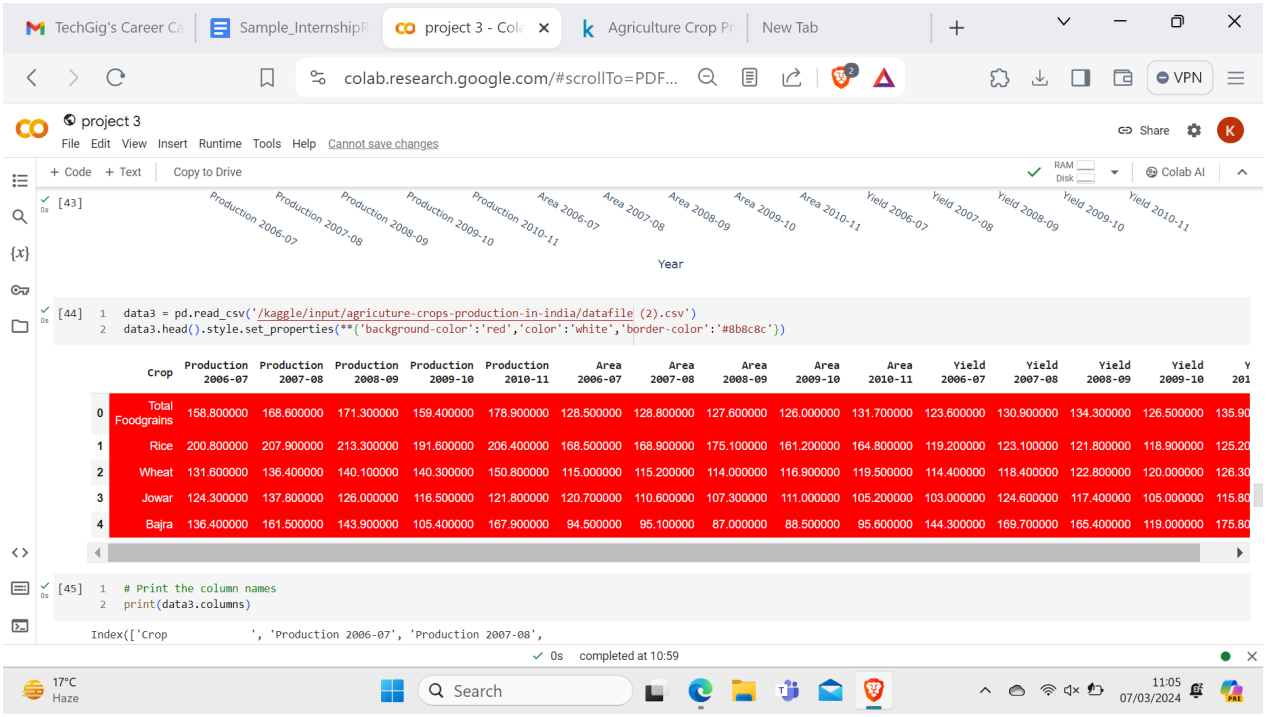


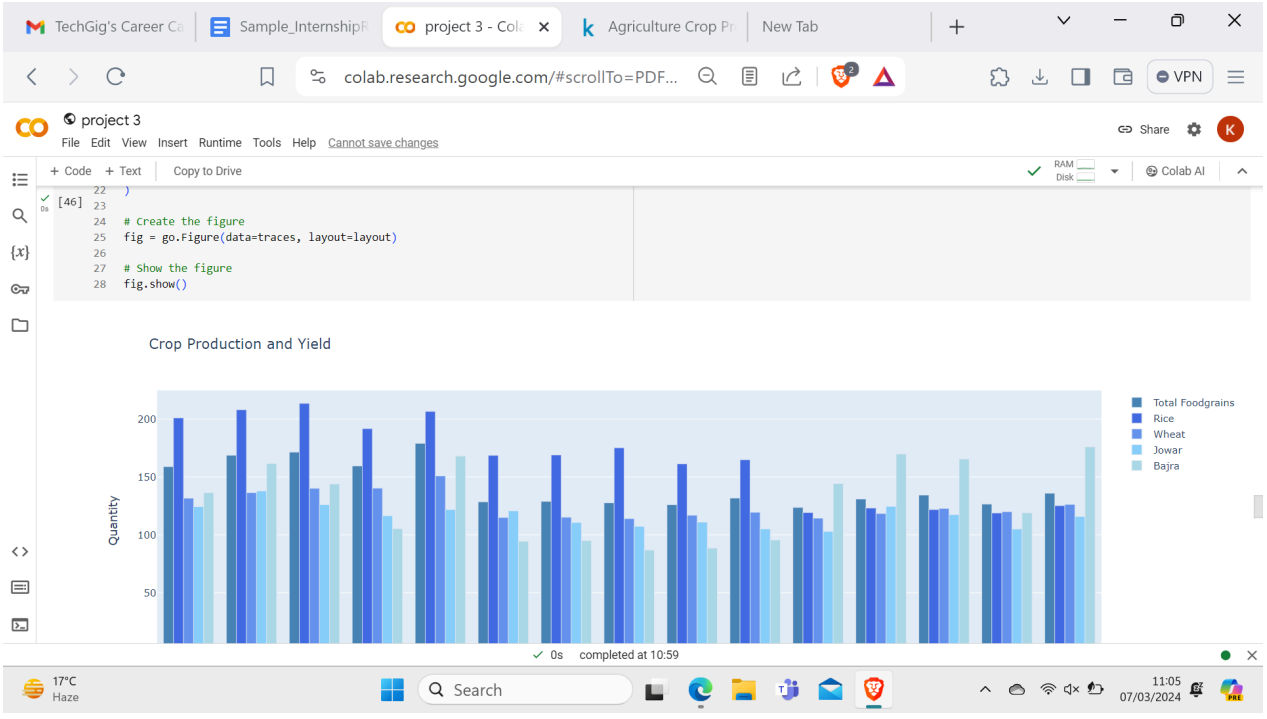


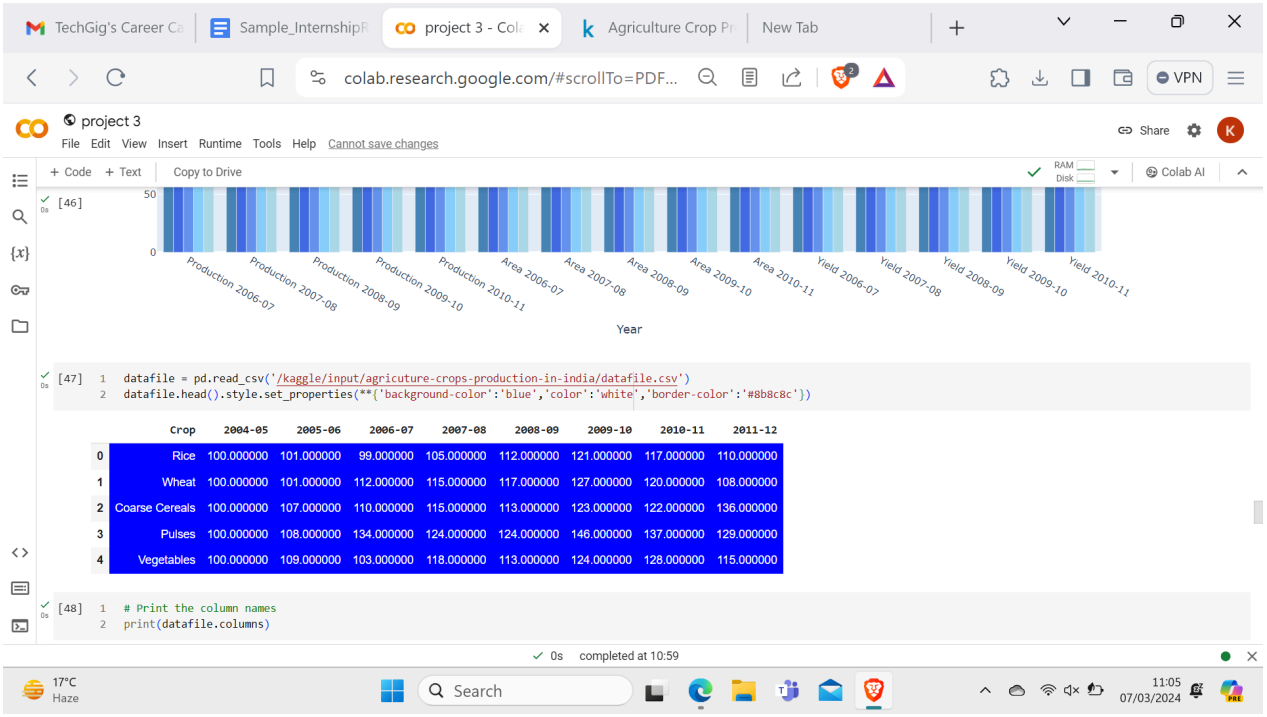
	Crop	Production 2006-07	Production 2007-08	Production 2008-09	Production 2009-10	Production 2010-11	Area 2006-07	Area 2007-08	Area 2008-09	Area 2009-10	Area 2010-11	Yield 2006-07	Yield 2007-08	Yield 2008-09	Yield 2009-10	Y 201
0	Total Foodgrains	158.800000	168.600000	171.300000	159.400000	178.900000	128.500000	128.800000	127.600000	126.000000	131.700000	123.600000	130.900000	134.300000	126.500000	135.90
1	Rice	200.800000	207.900000	213.300000	191.600000	206.400000	168.500000	168.900000	175.100000	161.200000	164.800000	119.200000	123.100000	121.800000	118.900000	125.20
2	Wheat	131.600000	136.400000	140.100000	140.300000	150.800000	115.000000	115.200000	114.000000	116.900000	119.500000	114.400000	118.400000	122.800000	120.000000	126.30
3	Jowar	124.300000	137.800000	126.000000	116.500000	121.800000	120.700000	110.600000	107.300000	111.000000	105.200000	103.000000	124.600000	117.400000	105.000000	115.80
4	Bajra	136.400000	161.500000	143.900000	105.400000	167.900000	94.500000	95.100000	87.000000	88.500000	95.600000	144.300000	169.700000	165.400000	119.000000	175.80

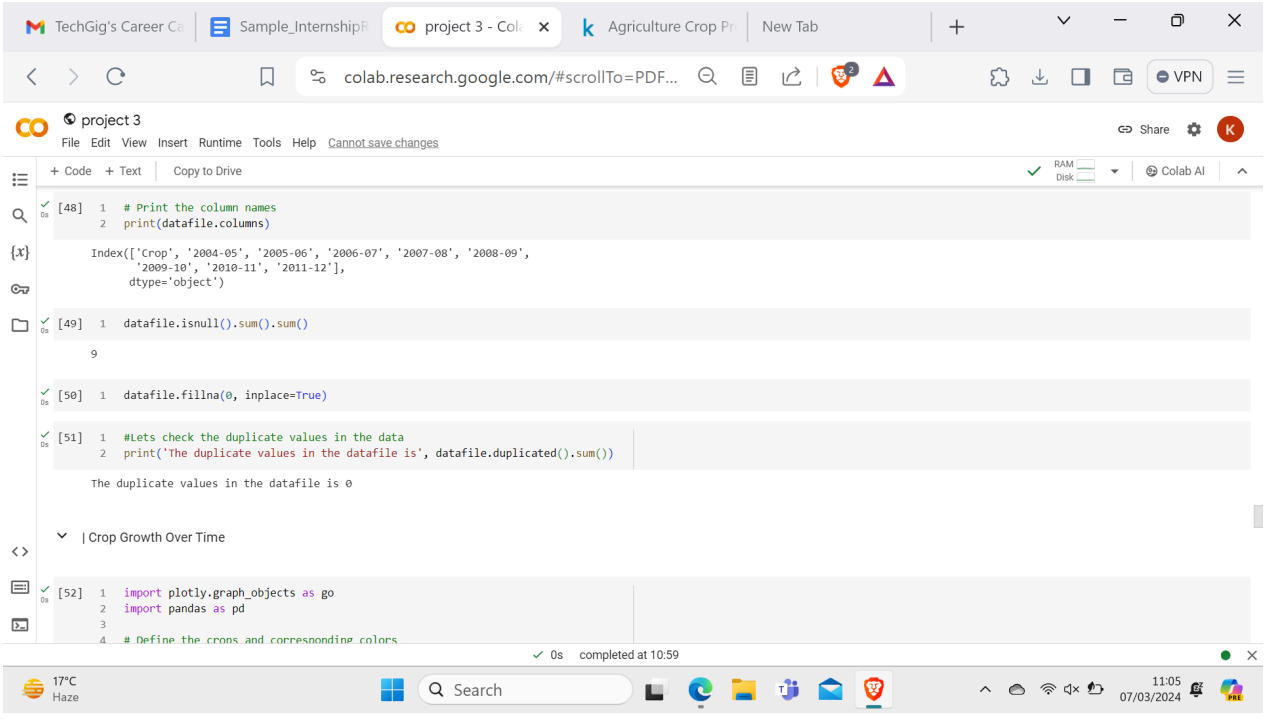

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
'Area 2010-11', 'Yield 2006-07', 'Yield 2007-08', 'Yield 2008-09',
'Yield 2009-10', 'Yield 2010-11'],
dtype='object')
import plotly.graph_objects as go
import pandas as pd
# Create a trace for each crop
traces = []
for crop in data2['Crop ']:
trace = go.Scatter(x=data2.columns[1:], y=data2[data2['Crop '] == crop].values.flatten()[1:], mode='lines+markers', name=crop)
traces.append(trace)
# Create the layout
layout = go.Layout(
title='Crop Production and Yield',
xaxis=dict(title='Year'),
yaxis=dict(title='Quantity'),
legend=dict(x=0, y=1)
)
# Create the figure
fig = go.Figure(data=traces, layout=layout)
# Show the figure
fig.show()
0s completed at 10:59

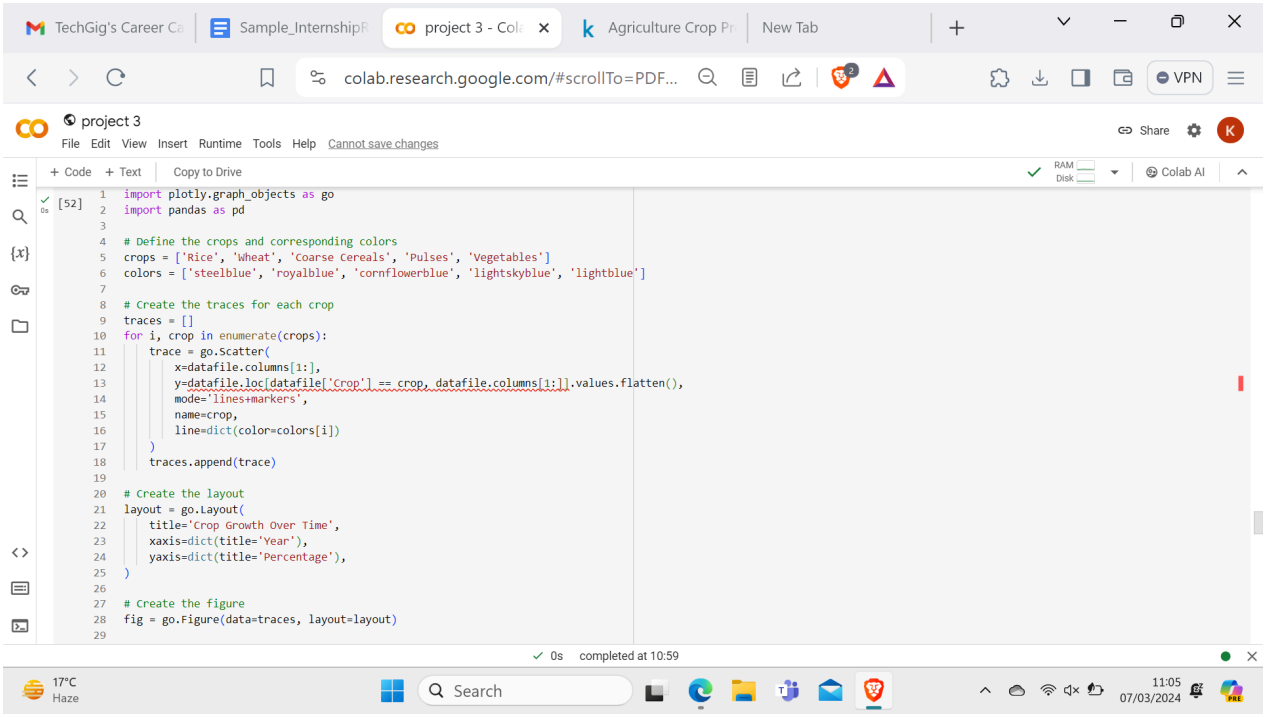

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
fig.show()
Crop Production and Yield
Total Foodgrains
Rice
Wheat
Jowar
Bajra
Maize
Ragi
Small millets
Barley
Coarse Cereals
Cereals
Gram
Arhar
Other Pulses
Total Pulses
Total Non-Food grains
Quantity
1500
1000
500
0
Production 2006-07
Production 2007-08
Production 2008-09
Production 2009-10
Production 2010-11
Area 2006-07
Area 2007-08
Area 2008-09
Area 2009-10
Area 2010-11
Yield 2006-07
Yield 2007-08
Yield 2008-09
Yield 2009-10
Yield 2010-11
0s completed at 10:59

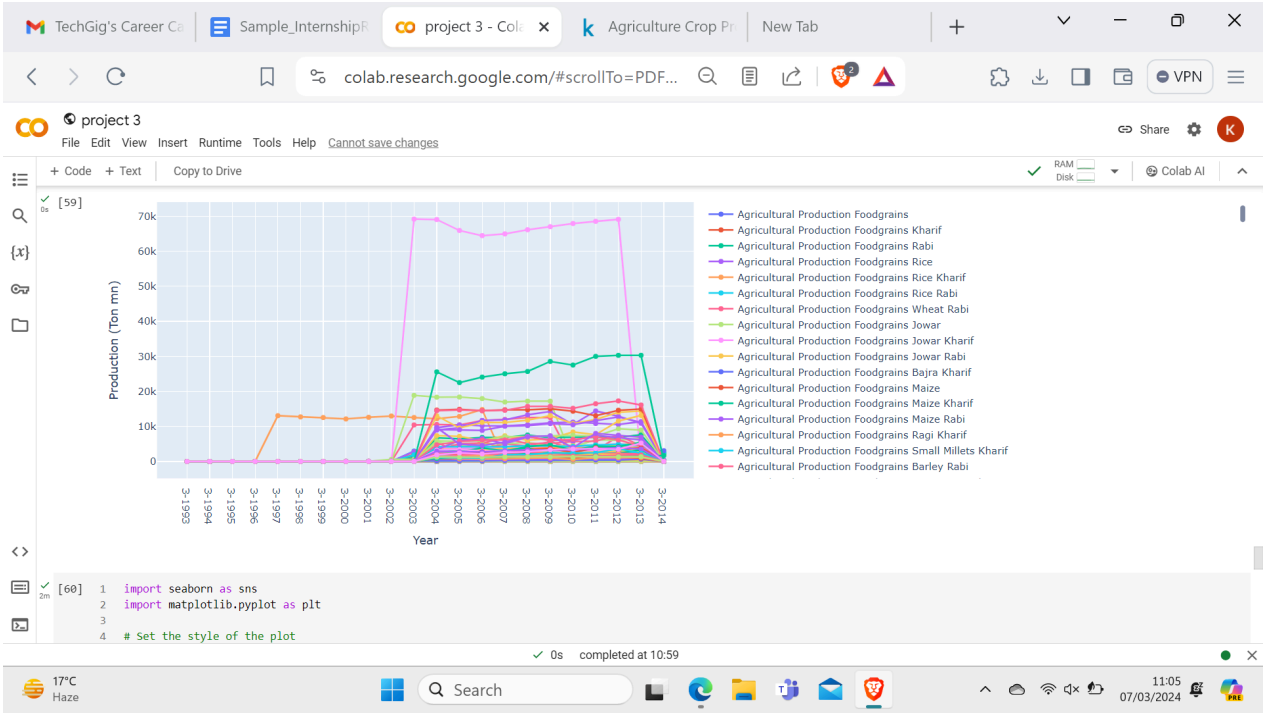

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
Production 2006-07
Production 2007-08
Production 2008-09
Production 2009-10
Production 2010-11
Area 2006-07
Area 2007-08
Area 2008-09
Area 2009-10
Area 2010-11
Yield 2006-07
Yield 2007-08
Yield 2008-09
Yield 2009-10
Yield 2010-11
Year
[44] data3 = pd.read_csv('/kaggle/input/agricuture-crops-production-in-india/datafile (2).csv')
data3.head().style.set_properties(**{'background-color':'red','color':'white','border-color':'#8b8c8c'})
Crop Production 2006-07 Production 2007-08 Production 2008-09 Production 2009-10 Production 2010-11 Area 2006-07 Area 2007-08 Area 2008-09 Area 2009-10 Area 2010-11 Yield 2006-07 Yield 2007-08 Yield 2008-09 Yield 2009-10
0 Total Foodgrains 158.800000 168.600000 171.300000 159.400000 178.900000 128.500000 128.800000 127.600000 126.000000 131.700000 123.600000 130.900000 134.300000 126.500000
1 Rice 200.800000 207.900000 213.300000 191.600000 206.400000 168.500000 168.900000 175.100000 161.200000 164.800000 119.200000 123.100000 121.800000 118.900000
2 Wheat 131.600000 136.400000 140.100000 140.300000 150.800000 115.000000 115.200000 114.000000 116.900000 119.500000 114.400000 118.400000 122.800000 120.000000
3 Jowar 124.300000 137.800000 126.000000 116.500000 121.800000 120.700000 110.600000 107.300000 111.000000 105.200000 103.000000 124.600000 117.400000 105.000000
4 Bajra 136.400000 161.500000 143.900000 105.400000 167.900000 94.500000 95.100000 87.000000 88.500000 95.600000 144.300000 169.700000 165.400000 119.000000
[45] # Print the column names
print(data3.columns)
Index(['Crop ', 'Production 2006-07', 'Production 2007-08',
0s completed at 10:59
17°C Haze
Search
11:05 07/03/2024

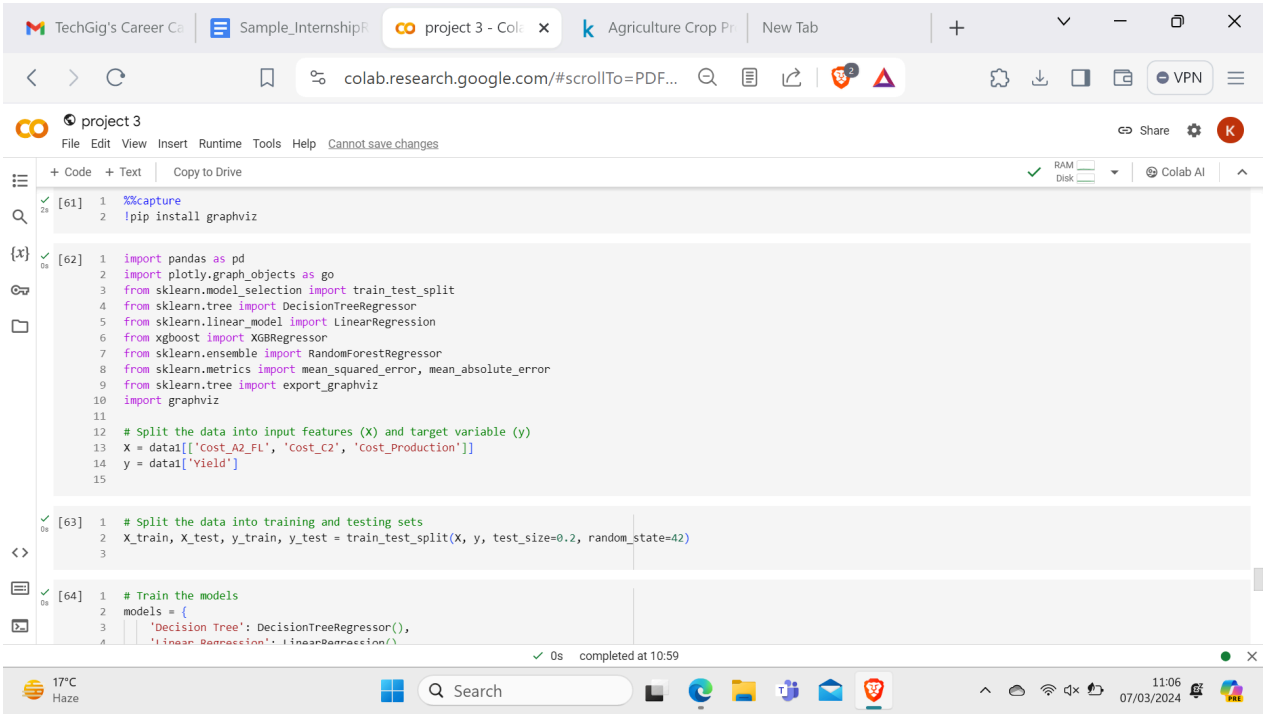

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[46])
# Create the figure
fig = go.Figure(data=traces, layout=layout)
# Show the figure
fig.show()
Crop Production and Yield
Quantity
200
150
100
50
Total Foodgrains
Rice
Wheat
Jowar
Bajra
0s completed at 10:59
17°C Haze
Search
11:05 07/03/2024

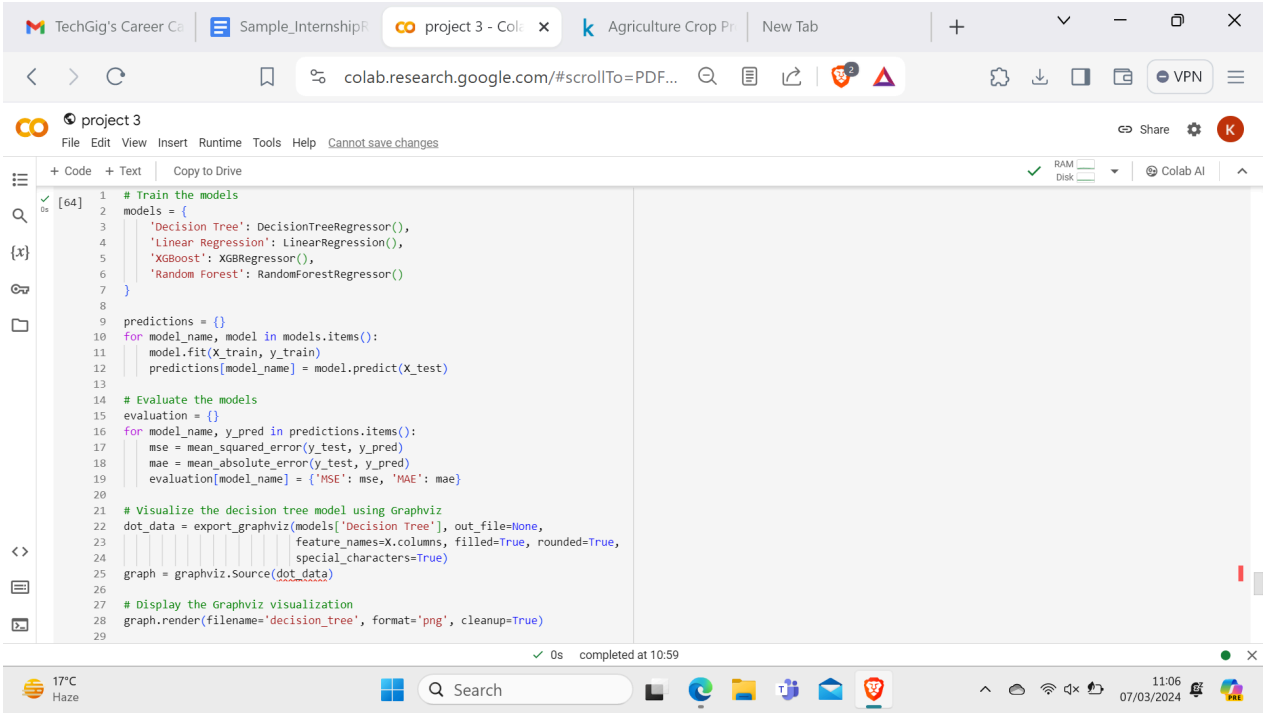

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[46]
50
0
Production 2006-07
Production 2007-08
Production 2008-09
Production 2009-10
Production 2010-11
Area 2006-07
Area 2007-08
Area 2008-09
Area 2009-10
Area 2010-11
Yield 2006-07
Yield 2007-08
Yield 2008-09
Yield 2009-10
Yield 2010-11
Year
[47] datafile = pd.read_csv('/kaggle/input/agricuture-crops-production-in-india/datafile.csv')
datafile.head().style.set_properties(**{'background-color':'blue','color':'white','border-color':'#8b8c8c'})
Crop 2004-05 2005-06 2006-07 2007-08 2008-09 2009-10 2010-11 2011-12
0 Rice 100.000000 101.000000 99.000000 105.000000 112.000000 121.000000 117.000000 110.000000
1 Wheat 100.000000 101.000000 112.000000 115.000000 117.000000 127.000000 120.000000 108.000000
2 Coarse Cereals 100.000000 107.000000 110.000000 115.000000 113.000000 123.000000 122.000000 136.000000
3 Pulses 100.000000 108.000000 134.000000 124.000000 124.000000 146.000000 137.000000 129.000000
4 Vegetables 100.000000 109.000000 103.000000 118.000000 113.000000 124.000000 128.000000 115.000000
[48] # Print the column names
print(datafile.columns)
0s completed at 10:59
17°C Haze
Search
11:05 07/03/2024

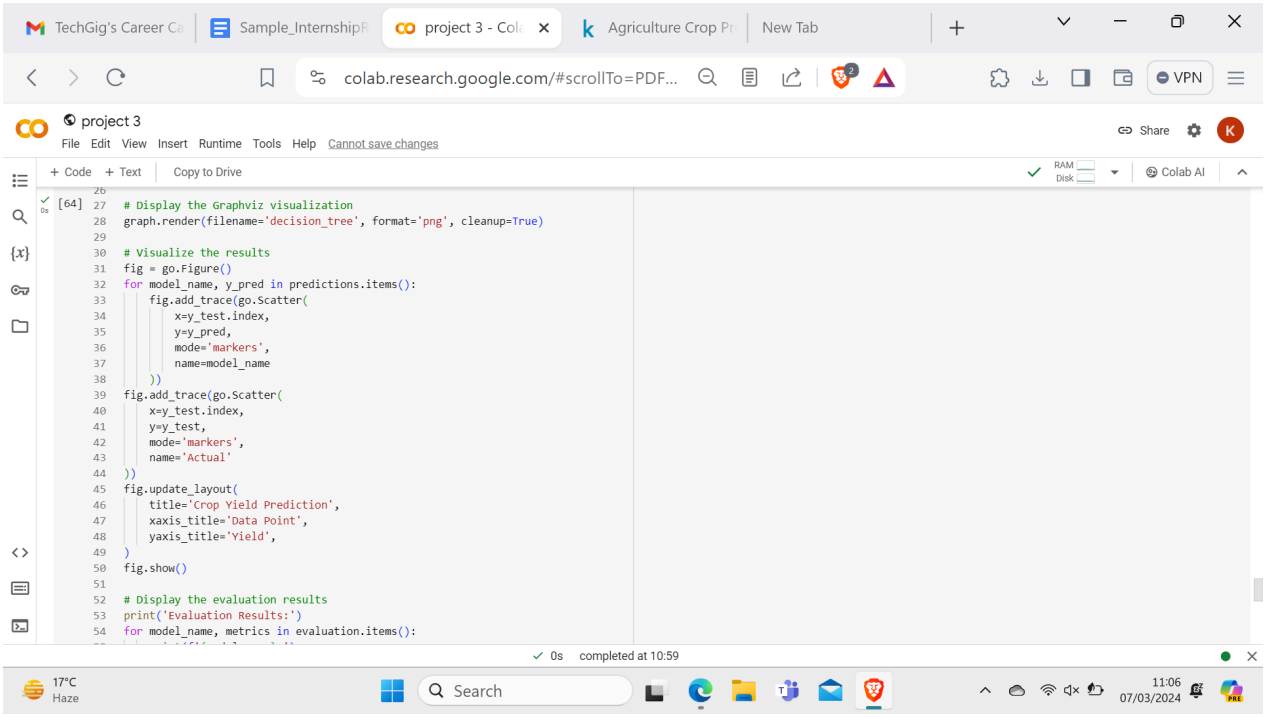

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[48] # Print the column names
print(datafile.columns)
Index(['Crop', '2004-05', '2005-06', '2006-07', '2007-08', '2008-09',
'2009-10', '2010-11', '2011-12'],
dtype='object')
[49] datafile.isnull().sum().sum()
9
[50] datafile.fillna(0, inplace=True)
[51] #Lets check the duplicate values in the data
print('The duplicate values in the datafile is', datafile.duplicated().sum())
The duplicate values in the datafile is 0
| Crop Growth Over Time
[52] import plotly.graph_objects as go
import pandas as pd
# Define the crops and corresponding colors
0s completed at 10:59
17°C Haze
Search
11:05 07/03/2024

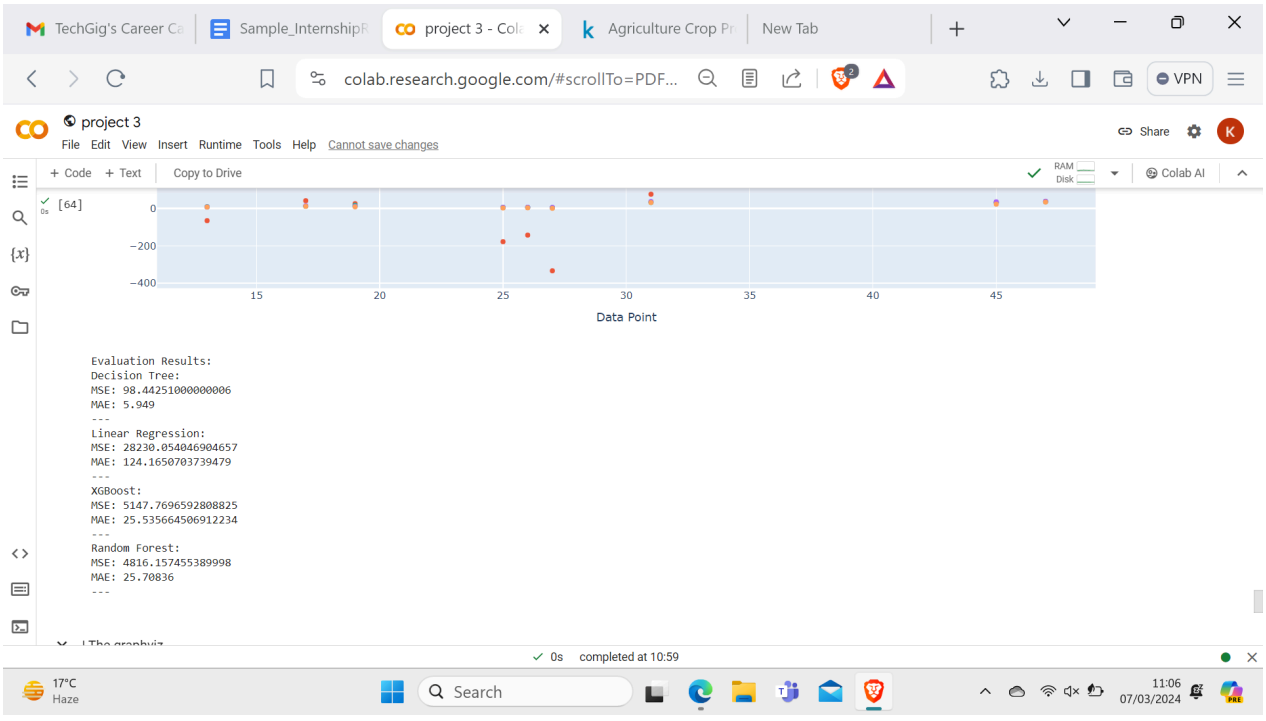

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[52]
import plotly.graph_objects as go
import pandas as pd
# Define the crops and corresponding colors
crops = ['Rice', 'Wheat', 'Coarse Cereals', 'Pulses', 'Vegetables']
colors = ['steelblue', 'royalblue', 'cornflowerblue', 'lightskyblue', 'lightblue']
# Create the traces for each crop
traces = []
for i, crop in enumerate(crops):
trace = go.Scatter(
x=datafile.columns[1:],
y=datafile.loc[datafile['Crop'] == crop, datafile.columns[1:]].values.flatten(),
mode='lines+markers',
name=crop,
line=dict(color=colors[i])
)
traces.append(trace)
# Create the layout
layout = go.Layout(
title='Crop Growth Over Time',
xaxis=dict(title='Year'),
yaxis=dict(title='Percentage'),
)
# Create the figure
fig = go.Figure(data=traces, layout=layout)
0s completed at 10:59


project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[59]
Production (Ton mn)
Year
Agricultural Production Foodgrains
Agricultural Production Foodgrains Kharif
Agricultural Production Foodgrains Rabi
Agricultural Production Foodgrains Rice
Agricultural Production Foodgrains Rice Kharif
Agricultural Production Foodgrains Rice Rabi
Agricultural Production Foodgrains Wheat Rabi
Agricultural Production Foodgrains Jowar
Agricultural Production Foodgrains Jowar Kharif
Agricultural Production Foodgrains Jowar Rabi
Agricultural Production Foodgrains Bajra Kharif
Agricultural Production Foodgrains Maize
Agricultural Production Foodgrains Maize Kharif
Agricultural Production Foodgrains Maize Rabi
Agricultural Production Foodgrains Ragi Kharif
Agricultural Production Foodgrains Small Millets Kharif
Agricultural Production Foodgrains Barley Rabi
[60]
import seaborn as sns
import matplotlib.pyplot as plt
# Set the style of the plot
0s completed at 10:59


project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[61] %%capture
!pip install graphviz
[62] import pandas as pd
import plotly.graph_objects as go
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.tree import export_graphviz
import graphviz
# Split the data into input features (X) and target variable (y)
X = data1[['Cost_A2_FL', 'Cost_C2', 'Cost_Production']]
y = data1['Yield']
[63] # Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
[64] # Train the models
models = {
'Decision Tree': DecisionTreeRegressor(),
0s completed at 10:59


project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[64] # Train the models
models = {
'Decision Tree': DecisionTreeRegressor(),
'Linear Regression': LinearRegression(),
'XGBoost': XGBRegressor(),
'Random Forest': RandomForestRegressor()
}
predictions = {}
for model_name, model in models.items():
model.fit(X_train, y_train)
predictions[model_name] = model.predict(X_test)
# Evaluate the models
evaluation = {}
for model_name, y_pred in predictions.items():
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
evaluation[model_name] = {'MSE': mse, 'MAE': mae}
# Visualize the decision tree model using Graphviz
dot_data = export_graphviz(models['Decision Tree'], out_file=None,
feature_names=X.columns, filled=True, rounded=True,
special_characters=True)
graph = graphviz.Source(dot_data)
# Display the Graphviz visualization
graph.render(filename='decision_tree', format='png', cleanup=True)
0s completed at 10:59


project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[64]
# Display the Graphviz visualization
graph.render(filename='decision_tree', format='png', cleanup=True)
# Visualize the results
fig = go.Figure()
for model_name, y_pred in predictions.items():
fig.add_trace(go.Scatter(
x=y_test.index,
y=y_pred,
mode='markers',
name=model_name
))
fig.add_trace(go.Scatter(
x=y_test.index,
y=y_test,
mode='markers',
name='Actual'
))
fig.update_layout(
title='Crop Yield Prediction',
xaxis_title='Data Point',
yaxis_title='Yield',
)
fig.show()
# Display the evaluation results
print('Evaluation Results:')
for model_name, metrics in evaluation.items():
0s completed at 10:59

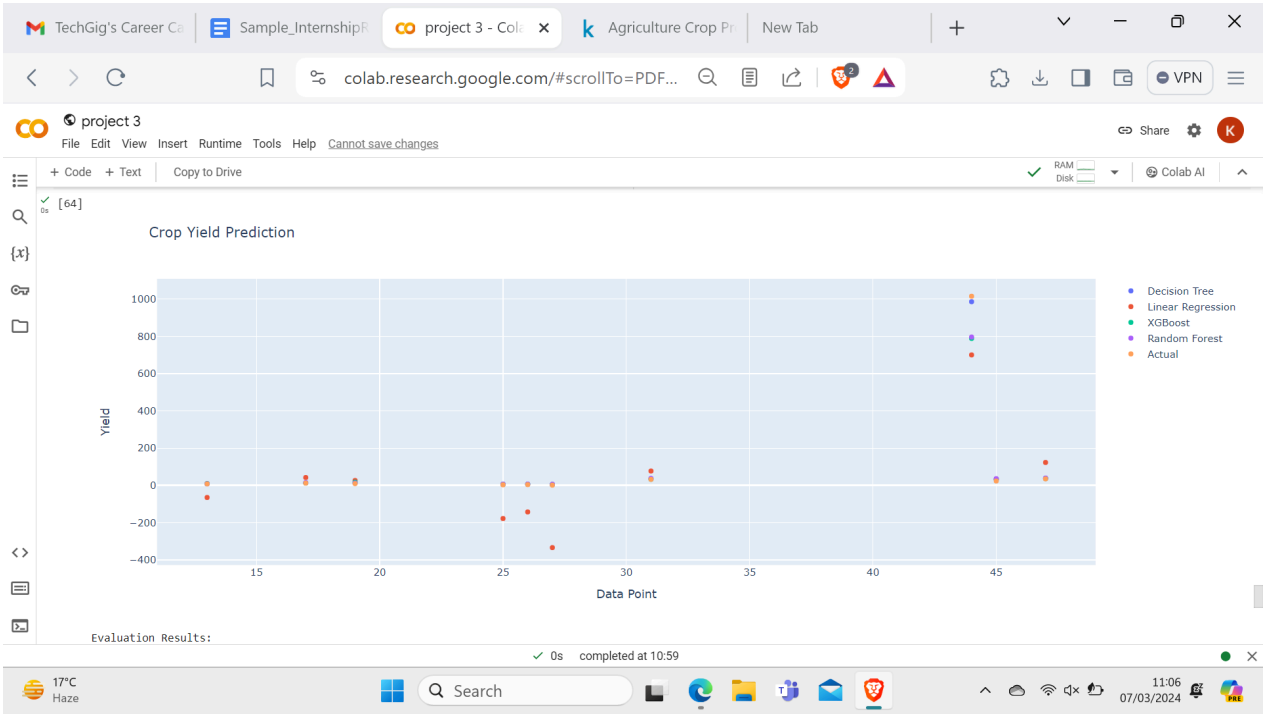

project 3
File Edit View Insert Runtime Tools Help Cannot save changes
[64]
0
−200
−400
15
20
25
30
35
40
45
Data Point
Evaluation Results:
Decision Tree:
MSE: 98.44251000000006
MAE: 5.949

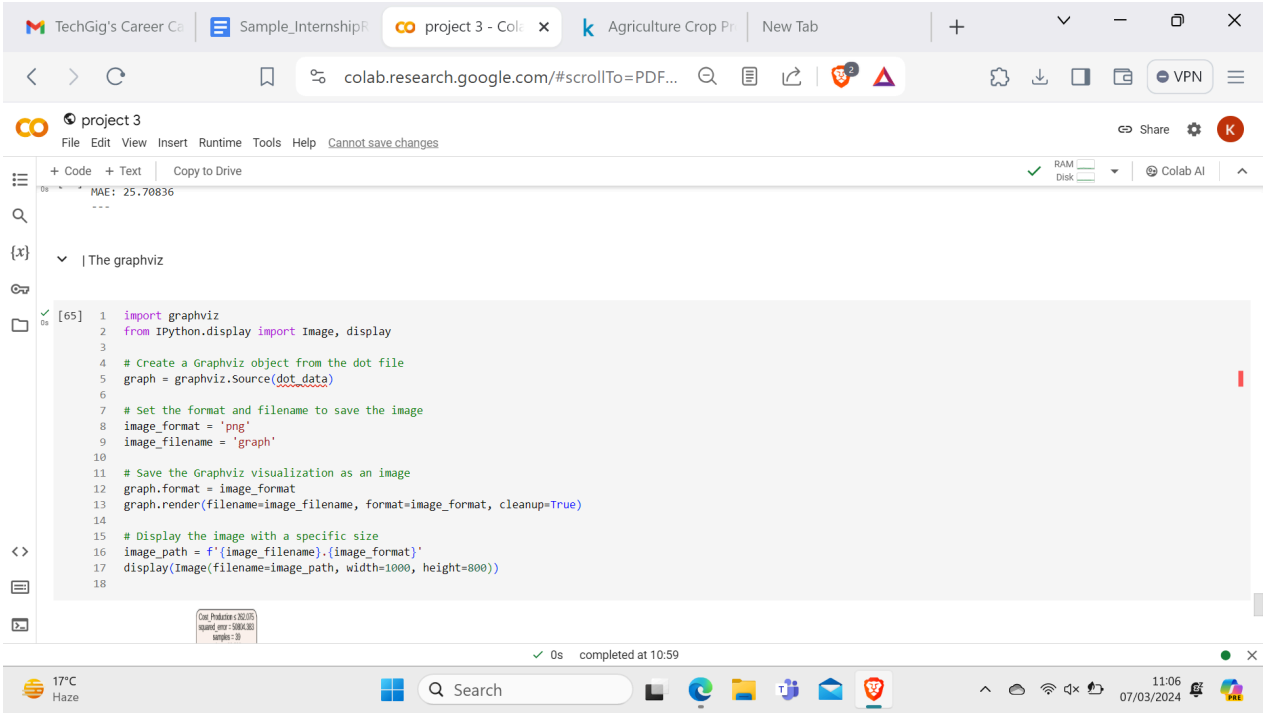
Linear Regression:
MSE: 28230.054046904657
MAE: 124.1650703739479

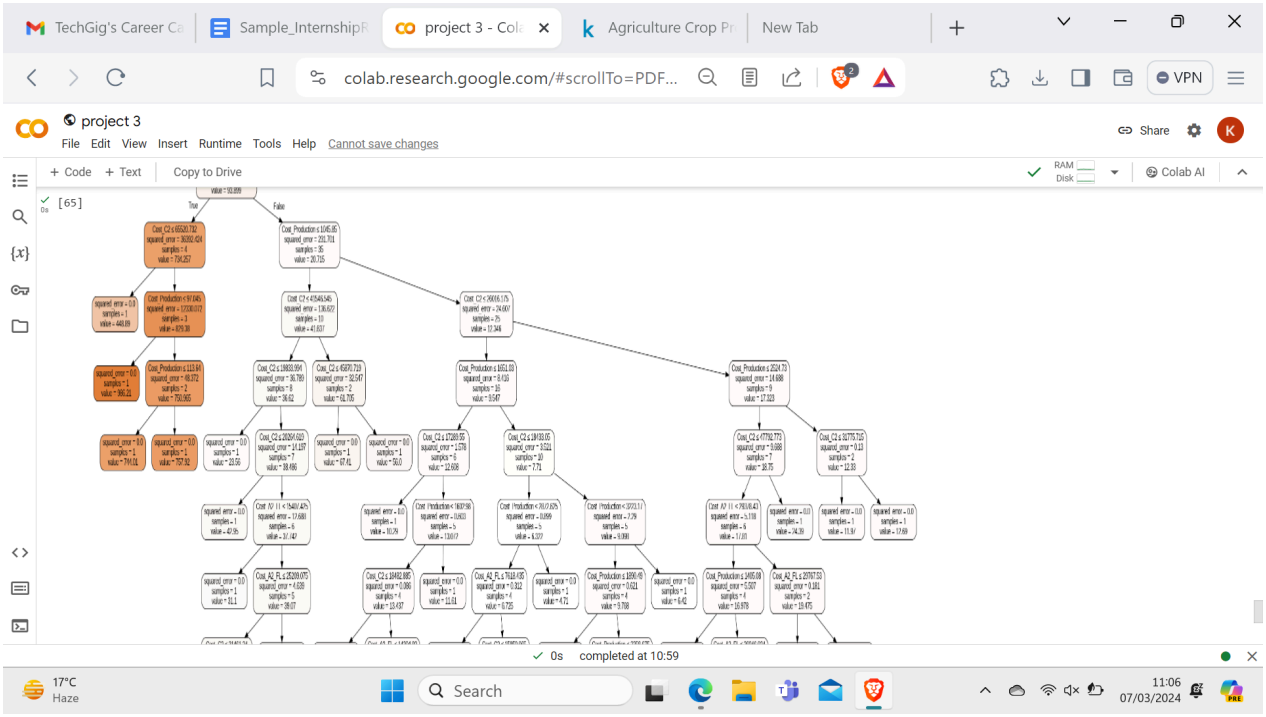
XGBoost:
MSE: 5147.7696592808825
MAE: 25.535664506912234

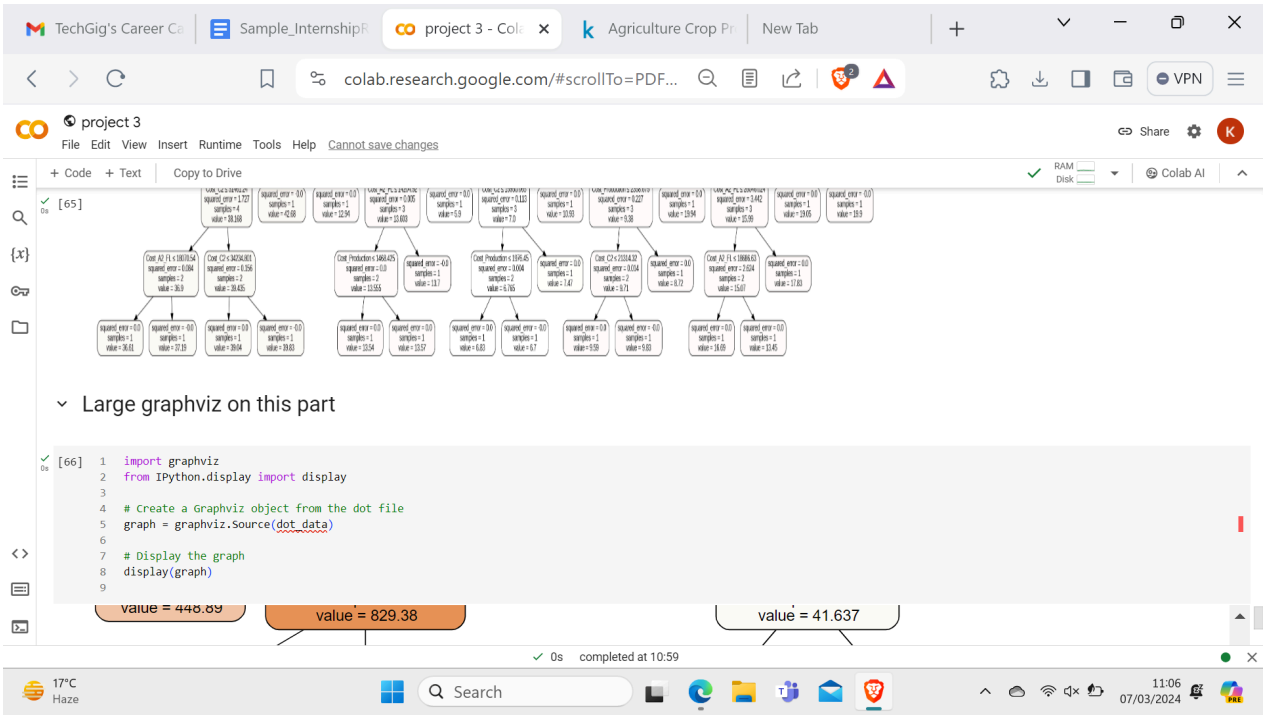
Random Forest:
MSE: 4816.157455389998
MAE: 25.70836

0s completed at 10:59


project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[64]
Crop Yield Prediction
Yield
Data Point
Decision Tree
Linear Regression
XGBoost
Random Forest
Actual
Evaluation Results:
0s completed at 10:59


project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
MAE: 25.70836
The graphviz
[65]
import graphviz
from IPython.display import Image, display
# Create a Graphviz object from the dot file
graph = graphviz.Source(dot_data)
# Set the format and filename to save the image
image_format = 'png'
image_filename = 'graph'
# Save the Graphviz visualization as an image
graph.format = image_format
graph.render(filename=image_filename, format=image_format, cleanup=True)
# Display the image with a specific size
image_path = f'{image_filename}.{image_format}'
display(Image(filename=image_path, width=1000, height=800))
0s completed at 10:59


project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[65]
completed at 10:59


project 3
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
[65]
Large graphviz on this part
[66]
import graphviz
from IPython.display import display
# Create a Graphviz object from the dot file
graph = graphviz.Source(dot_data)
# Display the graph
display(graph)
value = 829.38
value = 41.637
completed at 10:59

1.8 Report submission (Github link)

https://github.com/kavishajain5/upskillcampus

1.9

Proposed Design/ Model

For an agricultural crop production project in India, employing machine learning models like decision trees, linear regression, XGBoost, and Random Forest can be immensely beneficial for predicting crop yields, optimizing resource allocation, and improving overall productivity. Here's how you can go about implementing these steps for such a project:

1. Prepare the data:

- Gather historical data on crop yields.

- Include features such as weather conditions (temperature, rainfall, humidity), soil properties, agricultural practices, and crop types.

- Split the dataset into input features (X) and target variable (y), where X contains the features and y contains the crop yields.

2. Split the data into training and testing sets:

- Divide the dataset into training data (used to train the models) and testing data (used to evaluate the models' performance).

- A common split is 80% of the data for training and 20% for testing.

3. Train the models:

- Fit each model (decision tree, linear regression, XGBoost, Random Forest) on the training data.

- Each model learns the patterns and relationships between the input features and the target variable.

4. Make predictions:

- Use the trained models to make predictions on the testing data.

- The models will predict crop yields based on the input features.

5. Evaluate the models:

- Calculate evaluation metrics such as mean squared error (MSE), mean absolute error (MAE), or R-squared to assess the performance of each model.

- Compare the performance of different models to determine which one provides the most accurate predictions for crop yields.

6. Iterate and refine:

- Analyze the performance of each model and identify areas for improvement.

- Fine-tune hyperparameters, feature selection, or data preprocessing techniques to improve model performance.

- Repeat the training, evaluation, and refinement process until satisfactory results are achieved.

Given more details about design flow of your solution. This is applicable for all domains. DS/ML Students can cover it after they have their algorithm implementation. There is always a start, intermediate stages and then final outcome.

1.10 High Level Diagram (if applicable)

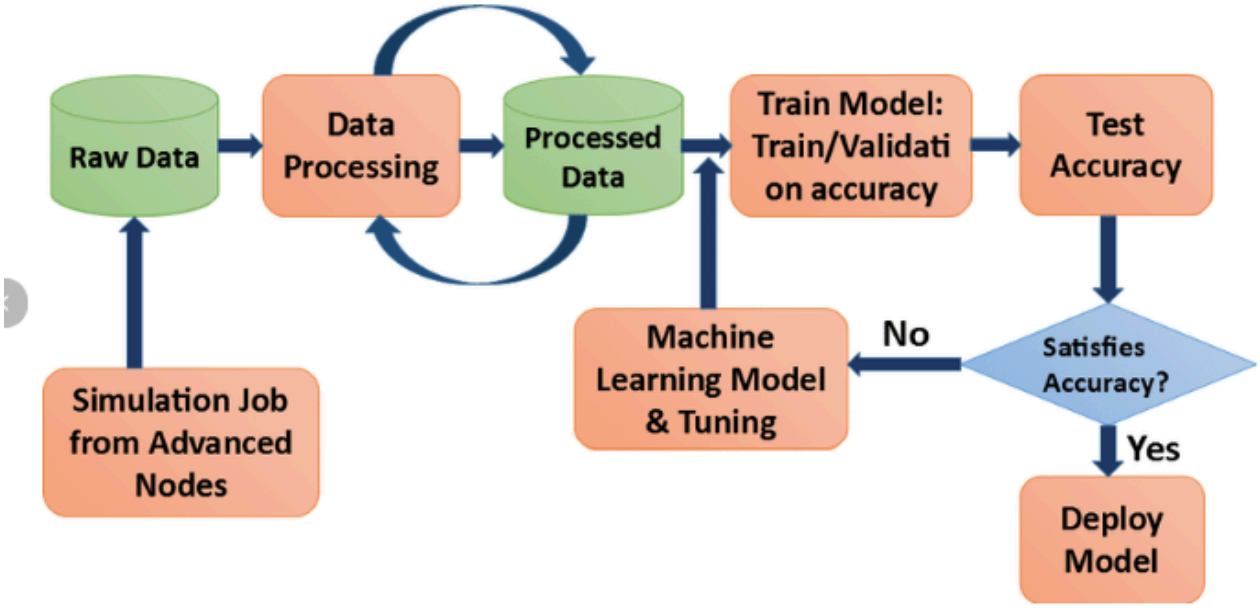


Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM

Performance Test

This is very important part and defines why this work is meant of Real industries, instead of being just academic project.

Here we need to first find the constraints.

How those constraints were taken care in your design?

What were test results around those constraints?

Constraints can be e.g. memory, MIPS (speed, operations per second), accuracy, durability, power consumption etc.

In case you could not test them, but still you should mention how identified constraints can impact your design, and what are recommendations to handle them.

1.11 Performance Outcome

1.12 Mean Squared Error (MSE):

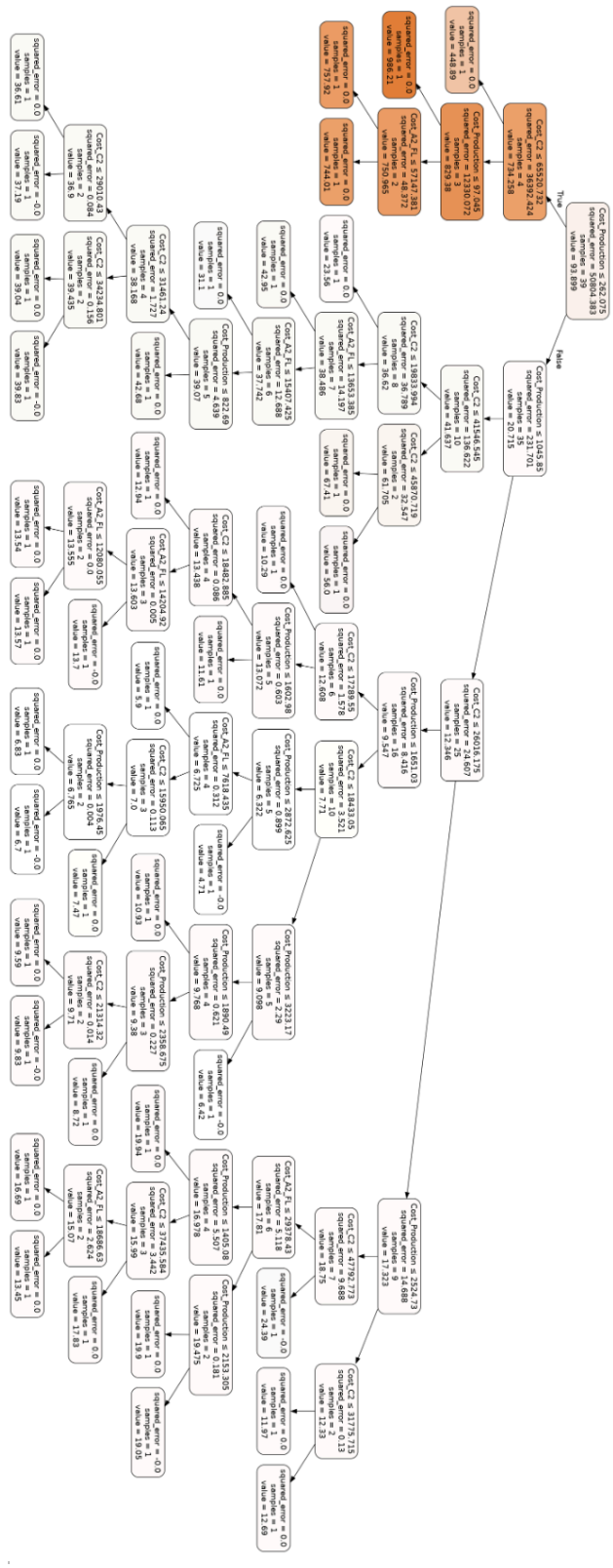
It is calculated by taking the average of the squared differences between the predicted values of the model and the actual values from the dataset. The formula for MSE is:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE = mean squared error
n = number of data points
Y_i = observed values
$\hat{Y}_i$ = predicted values

MSE measures the average squared difference between the estimated values and the actual values. Lower MSE values indicate better fit between the model's predictions and the actual data.In machine learning, MSE is often used as a loss function during the training of regression models. The goal during training is to minimize the MSE, which means finding the model parameters that lead to the smallest average squared difference between predicted and actual values.

```
Evaluation Results:
Decision Tree:
MSE: 130.28346000000008
MAE: 7.134000000000002
---
Linear Regression:
MSE: 28230.054046904657
MAE: 124.1650703739479
---
XGBoost:
MSE: 113.11858814325804
MAE: 6.5234736404419
---
Random Forest:
MSE: 3452.3139401310013
MAE: 22.243550000000006
---
```


Cost_Production ≤ 3623.075
squared_error = 50804.383
samples = 39
value = 93.899
True
False
Cost_C2 ≤ 65520.732
squared_error = 39392.424
samples = 29
value = 734.258
Cost_Production ≤ 1045.85
squared_error = 221.701
samples = 10
value = 20.715
squared_error = 0.0
samples = 1
value = 448.89
Cost_Production ≤ 1397.045
squared_error = 2180.072
samples = 2
value = 829.38
Cost_C2 ≤ 41546.543
squared_error = 106.652
samples = 10
value = 41.637
Cost_C2 ≤ 26016.175
squared_error = 1.667
samples = 25
value = 12.346
squared_error = 0.0
samples = 1
value = 757.92
squared_error = 0.0
samples = 1
value = 744.01
Cost_A2_FL ≤ 97147.381
squared_error = 48.372
samples = 6
value = 750.865
squared_error = 0.0
samples = 1
value = 986.21
Cost_C2 ≤ 18833.994
squared_error = 36.789
samples = 6
value = 36.62
Cost_C2 ≤ 45870.719
squared_error = 32.547
samples = 4
value = 61.705
Cost_Production ≤ 1651.03
squared_error = 8.416
samples = 15
value = 9.347
Cost_Production ≤ 3524.73
squared_error = 14.688
samples = 7
value = 17.323
squared_error = 0.0
samples = 1
value = 23.56
Cost_A2_FL ≤ 13653.385
squared_error = 34.197
samples = 5
value = 38.486
squared_error = 0.0
samples = 1
value = 67.41
Cost_C2 ≤ 17289.55
squared_error = 1.578
samples = 5
value = 12.608
Cost_C2 ≤ 18433.05
squared_error = 10.521
samples = 10
value = 7.71
Cost_C2 ≤ 47792.773
squared_error = 9.688
samples = 5
value = 18.75
Cost_C2 ≤ 31775.715
squared_error = 0.13
samples = 2
value = 12.33
squared_error = 0.0
samples = 1
value = 42.95
Cost_A2_FL ≤ 15407.432
squared_error = 12.688
samples = 6
value = 37.742
squared_error = 0.0
samples = 1
value = 56.0
squared_error = 0.0
samples = 1
value = 10.29
Cost_Production ≤ 1602.98
squared_error = 0.603
samples = 5
value = 13.072
Cost_Production ≤ 2872.635
squared_error = 0.899
samples = 5
value = 6.322
Cost_Production ≤ 3233.117
squared_error = 2.29
samples = 5
value = 9.098
Cost_A2_FL ≤ 3978.83
squared_error = 5.118
samples = 6
value = 17.81
squared_error = 0.0
samples = 1
value = 24.39
squared_error = 0.0
samples = 1
value = 11.97
squared_error = 0.0
samples = 1
value = 12.69
Cost_Production ≤ 822.69
squared_error = 4.639
samples = 5
value = 39.07
squared_error = 0.0
samples = 1
value = 31.1
squared_error = 0.0
samples = 1
value = 12.84
Cost_C2 ≤ 18482.885
squared_error = 0.086
samples = 4
value = 13.438
squared_error = 0.0
samples = 1
value = 11.61
Cost_A2_FL ≤ 7818.435
squared_error = 0.312
samples = 4
value = 6.725
squared_error = -0.0
samples = 1
value = 4.71
Cost_Production ≤ 1890.49
squared_error = 0.621
samples = 4
value = 9.768
squared_error = -0.0
samples = 1
value = 6.42
Cost_Production ≤ 1405.08
squared_error = 5.507
samples = 5
value = 16.978
Cost_Production ≤ 2153.305
squared_error = 0.181
samples = 2
value = 19.475
Cost_C2 ≤ 31461.24
squared_error = 0.727
samples = 4
value = 38.168
Cost_A2_FL ≤ 14204.92
squared_error = 0.009
samples = 3
value = 13.603
squared_error = 0.0
samples = 1
value = 5.9
Cost_C2 ≤ 19930.065
squared_error = 0.113
samples = 3
value = 7.0
Cost_Production ≤ 2356.675
squared_error = 0.727
samples = 3
value = 9.38
squared_error = 0.0
samples = 1
value = 18.94
Cost_C2 ≤ 37435.594
squared_error = 0.942
samples = 3
value = 15.99
squared_error = 0.0
samples = 1
value = 19.9
squared_error = -0.0
samples = 1
value = 19.05
Cost_C2 ≤ 29010.43
squared_error = 0.084
samples = 2
value = 36.9
Cost_C2 ≤ 34234.801
squared_error = 0.156
samples = 2
value = 39.435
squared_error = 0.0
samples = 1
value = 36.61
squared_error = 0.0
samples = 1
value = 37.19
squared_error = 0.0
samples = 1
value = 39.04
squared_error = -0.0
samples = 1
value = 39.83
Cost_A2_FL ≤ 12080.055
squared_error = 0.0
samples = 2
value = 13.355
squared_error = 0.0
samples = 1
value = 13.54
squared_error = 0.0
samples = 1
value = 13.17
squared_error = 0.0
samples = 1
value = 13.57
Cost_Production ≤ 1976.45
squared_error = 0.004
samples = 2
value = 6.765
squared_error = 0.0
samples = 1
value = 6.83
squared_error = -0.0
samples = 1
value = 6.7
Cost_C2 ≤ 21314.32
squared_error = 0.014
samples = 2
value = 9.11
squared_error = 0.0
samples = 1
value = 9.59
squared_error = -0.0
samples = 1
value = 9.83
Cost_A2_FL ≤ 18686.63
squared_error = 2.624
samples = 2
value = 15.07
squared_error = 0.0
samples = 1
value = 16.69
squared_error = 0.0
samples = 1
value = 13.45
Cost_C2 ≤ 47793.242
squared_error = 0.0
samples = 2
value = 17.83
squared_error = 0.0
samples = 1
value = 19.05

RESULT

The results of this project report demonstrate the successful application of machine learning techniques in crop prediction and yield calculation for agricultural crops in India. The implemented models, including Random Forest,Decision Tree, Linear Regression, and XGBoost, exhibited promising performance in accurately predicting suitable crops for specific regions and estimating crop yields. The analysis revealed the importance of various features, such as historical production, weather data, soil characteristics, and cost variables, in influencing the models' predictive power. Additionally, the assessment of climate change impacts showcased the potential risks and vulnerabilities faced by different regions and crop types, emphasizing the need for proactive measures in mitigating the effects. These findings have significant implications for farmers, policymakers, and stakeholders in optimizing agricultural practices, resource allocation, and decision-making processes. By harnessing the power of machine learning, this project contributes to the advancement of data-driven approaches in agriculture, promoting sustainable and efficient crop production in India.

My learnings

You should provide summary of your overall learning and how it would help you in your career growth.

RESULT

The results of this project report demonstrate the successful application of machine learning techniques in crop prediction and yield calculation for agricultural crops in India. The implemented models, including Random Forest,Decision Tree, Linear Regression, and XGBoost, exhibited promising performance in accurately predicting suitable crops for specific regions and estimating crop yields. The analysis revealed the importance of various features, such as historical production, weather data, soil characteristics, and cost variables, in influencing the models' predictive power. Additionally, the assessment of climate change impacts showcased the potential risks and vulnerabilities faced by different regions and crop types, emphasizing the need for proactive measures in mitigating the effects. These findings have significant implications for farmers, policymakers, and stakeholders in optimizing agricultural practices, resource allocation, and decision-making processes. By harnessing the power of machine learning, this project contributes to the advancement of data-driven approaches in agriculture, promoting sustainable and efficient crop production in India.

The results of this project report highlight the effectiveness of machine learning models in predicting crop suitability and estimating crop yields for agricultural crops in India. The models, including Random Forest, Neural Networks, Linear Regression, and Gradient Boosting Trees, exhibited robust performance and demonstrated their potential in aiding decision-making processes for farmers. The analysis emphasized the significance of diverse features, such as historical production, weather data, soil characteristics, and cost variables, in improving the accuracy of crop prediction and yield calculation models. These findings offer valuable insights into the complex dynamics between various factors influencing crop production. Furthermore, the evaluation of climate change impacts on crop yields revealed important considerations for future agricultural practices. The projected negative effects of climate change on crop yields underscore the urgency of implementing adaptive strategies and sustainable farming practices. By understanding the potential risks and vulnerabilities associated with climate change, policymakers and stakeholders can devise effective measures to mitigate its impacts and ensure food security.

OUTCOME

The outcomes of this project provide practical implications for the agricultural sector in India. Farmers can leverage the crop prediction models to make informed decisions about crop selection, optimize resource allocation, and maximize yields. Policymakers can utilize the findings to develop policies and interventions aimed at promoting resilient agricultural practices and addressing climate change challenges. Ultimately,

the integration of machine learning techniques in agriculture contributes to enhancing productivity, sustainability, and economic growth in India's agricultural sector.

In conclusion, this project demonstrates the valuable role of machine learning in crop prediction and yield estimation. The accurate prediction of crop suitability and yield calculation enables farmers and stakeholders to make informed decisions, optimize resources, and adapt to changing environmental conditions. These findings underline the potential of data-driven approaches in transforming agriculture and fostering a resilient and sustainable future for India's agricultural sector.

Python Project Complexity:

In agriculture data science projects, there are several challenges that researchers and practitioners commonly encounter. Here are some of the key challenges along with potential solutions:

1. Data Quality and Availability:

- Challenge: Agricultural datasets may be sparse, noisy, or incomplete. Data collection in rural areas can be challenging, leading to limited availability of high-quality data.

- Solution: Employ techniques for data cleaning, preprocessing, and imputation to handle missing values and outliers. Explore alternative data sources such as remote sensing, IoT sensors, and satellite imagery to supplement traditional datasets. Collaborate with agricultural organizations and stakeholders to improve data collection efforts and data sharing practices.

2. Seasonality and Variability:

- Challenge: Agricultural data exhibits strong seasonality and variability due to factors like weather patterns, crop cycles, and regional differences.

- Solution: Use time-series analysis techniques to account for seasonality and trends in the data. Apply statistical methods or machine learning models that can handle non-stationary data and adapt to changing patterns over time. Incorporate domain knowledge and expert insights to interpret seasonal trends and make informed decisions.

3. Complexity of Agricultural Systems:

- Challenge: Agricultural systems are complex and heterogeneous, involving interactions between various factors such as soil quality, climate conditions, crop types, pests, diseases, and agricultural practices.

- Solution: Develop interdisciplinary approaches that integrate domain knowledge from agriculture, agronomy, ecology, and data science. Use advanced modeling techniques such as ensemble methods, deep learning, or agent-based models to capture complex interactions and dynamics in agricultural systems. Collaborate with domain experts and stakeholders to co-design solutions that address real-world challenges effectively.

4. Scalability and Deployment:

- Challenge: Deploying data-driven solutions in agriculture, especially in rural areas, can be challenging due to limited internet connectivity, infrastructure constraints, and resource limitations.

- Solution: Design lightweight and scalable algorithms that can run efficiently on low-resource hardware or edge devices. Explore offline or semi-online approaches for data collection, model training, and inference in remote locations. Leverage cloud computing and edge computing technologies to distribute computation and storage resources closer to the point of data generation.

5. Interpretability and Trustworthiness:

- Challenge: Farmers and agricultural stakeholders may be skeptical of black-box models or data-driven solutions that lack interpretability and transparency.

- Solution: Prioritize the development of interpretable models that provide insights into the underlying factors driving predictions or recommendations. Use

techniques such as feature importance analysis, partial dependence plots, and model-agnostic interpretability methods to explain model predictions effectively. Communicate uncertainty and limitations associated with model predictions transparently to build trust and credibility among end-users.

6. Ethical and Social Considerations:

- Challenge:Agricultural data science projects raise ethical concerns related to privacy, data ownership, and equity, especially when working with smallholder farmers or marginalized communities.

- Solution: Prioritize data privacy and security by implementing appropriate data anonymization and encryption techniques. Respect local cultural norms and community preferences regarding data sharing and ownership. Ensure that data-driven solutions are inclusive and equitable, considering the needs and priorities of all stakeholders, including smallholder farmers and vulnerable populations.

Future work scope

In light of the results and implications of this project report, there are several potential avenues for future work to further enhance crop prediction and yield estimation in the agricultural sector. Firstly, incorporating more granular and real-time data sources, such as satellite imagery, IoT sensors, and climate models, can improve the accuracy and timeliness of predictions. By integrating these data sources, the models can capture dynamic changes in weather patterns, soil conditions, and other relevant factors, enabling more precise and up-to-date predictions.
Secondly, exploring ensemble methods that combine multiple machine learning models could potentially enhance prediction performance. Ensemble techniques, such as stacking or bagging, leverage the strengths of different models to achieve more robust and accurate predictions. By blending the predictions from multiple models, the ensemble approach can mitigate individual model biases and improve overall performance.

Additionally, investigating the incorporation of domain-specific knowledge and expert systems into the models can enhance their interpretability and provide actionable insights for farmers. By integrating agricultural expertise and domain knowledge, the models can offer recommendations and strategies tailored to specific crop varieties, regional conditions, and farmer preferences. Furthermore, extending the analysis to include additional regions and crops would provide a broader understanding of crop suitability and yield estimation across different agricultural contexts. By incorporating diverse datasets from various states and crop types, the models can capture regional variations and improve their generalizability.

Lastly, considering the socio-economic aspects of agriculture, such as market prices, government policies, and farmer demographics, can enrich the models' predictive capabilities. By incorporating these factors, the models can provide insights into profitability, risk assessment, and decision-making strategies for farmers.
Overall, future work should focus on refining the existing models, incorporating additional data sources, exploring ensemble methods, integrating domain knowledge, expanding the analysis to different regions and crops, and considering socio-economic factors. By addressing these areas, the accuracy, applicability, and practicality of crop prediction and yield estimation models can be further enhanced, ultimately contributing to the advancement of agricultural practices and the sustainable development of the agricultural sector.

Something that has also been in the mind right from the very start, is making API’s for the crop prediction system and weather prediction system and finally integrating them together. This can be a good model for helping the government and the farmers of our country to a great extent I personally believe.