ANALYSIS ON
GARMENT WORKER'S PRODUCTIVITY

Presented by:

Group - 3

BHAVANS VIVEKANANDA COLLEGE

group members:

SRAVAN KUMAR / VAMSHI / SRI VANI / NITHIN CHANDRA

Abstract:

This project focuses on analyzing and modeling productivity among garment factory workers using a detailed dataset comprising various production and worker-related metrics.

The study focuses on multiple machine learning algorithms, including: Regression,KNN,SVM,Decision trees, and ensemble algorithms.

Objective:

The primary objective of this project is to analyze and predict the productivity of garment factory workers by examining various production and workforce factors

Content:

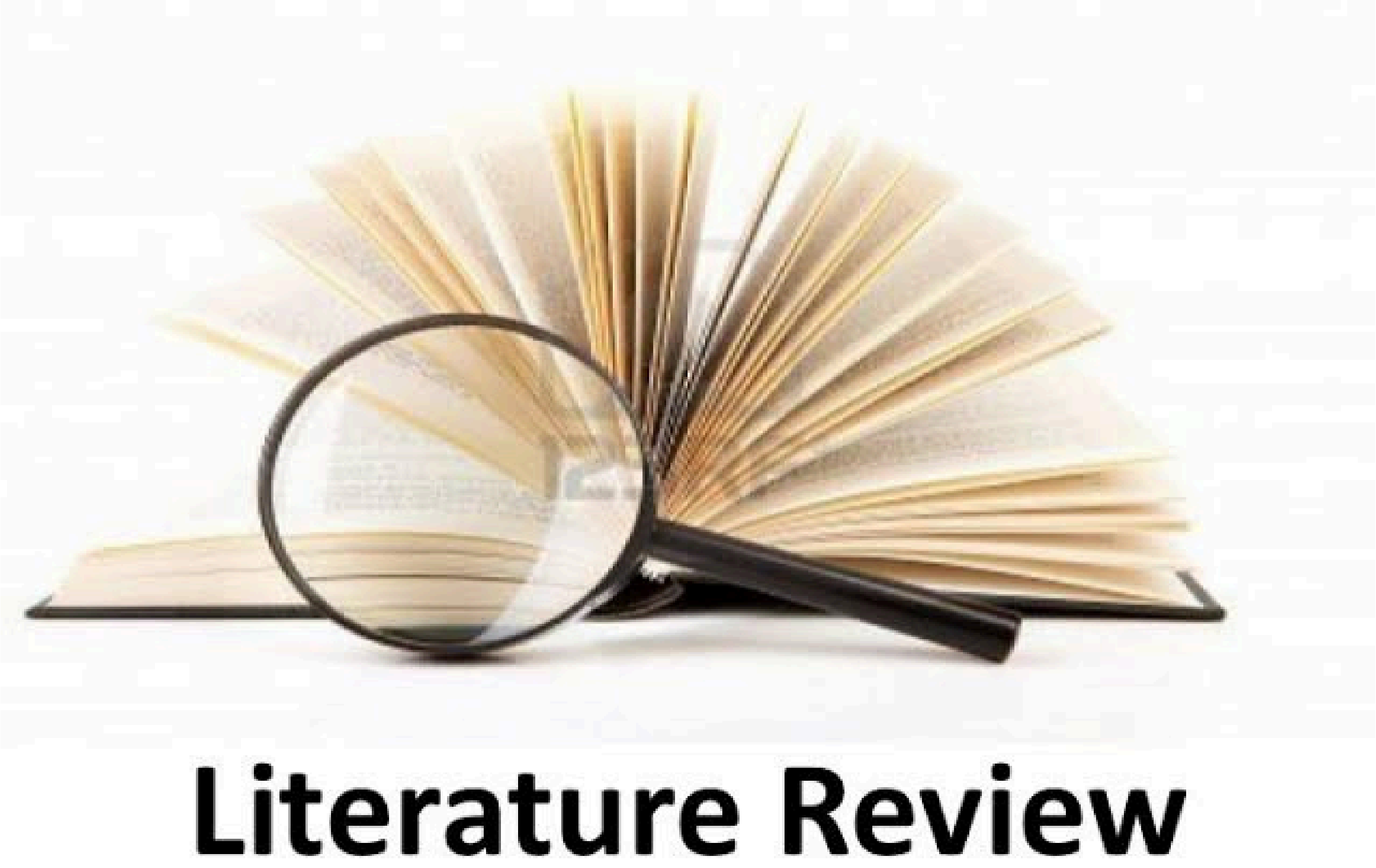
Introduction:

- Purpose: Analyze productivity trends, identify influential factors, and use machine learning models to predict productivity.

The garment manufacturing industry is characterized by complex processes and various factors that influence worker productivity.
This project focuses on analyzing a dataset related to garment workers' productivity, which includes variables such as working hours, incentives, idle time, work-in-progress (WIP), and productivity targets.

- Conducted exploratory data analysis (EDA) to understand distributions and correlations.
- Checked and addressed multicollinearity using VIF.
- Evaluated various machine learning models, including regression, KNN, SVM, decision trees, and ensemble techniques (bagging and boosting).
- Compared model performance before and after handling multicollinearity to identify the most effective predictive approach.

LITERATURE REVIEW



Literature review - 1:

Kumar , Patel & Saha

Technological and Incentive Impacts: Studies such as Kumar and Saha (2019) have examined how the use of technology and performance incentives (reflected in your dataset's incentive column) affect productivity.

Patel emphasizes the importance of visual tools like scatter plots, box plots, and histograms for uncovering patterns and relationships between variables before applying predictive models.

Published in International Journal of Commerce and Management Research.

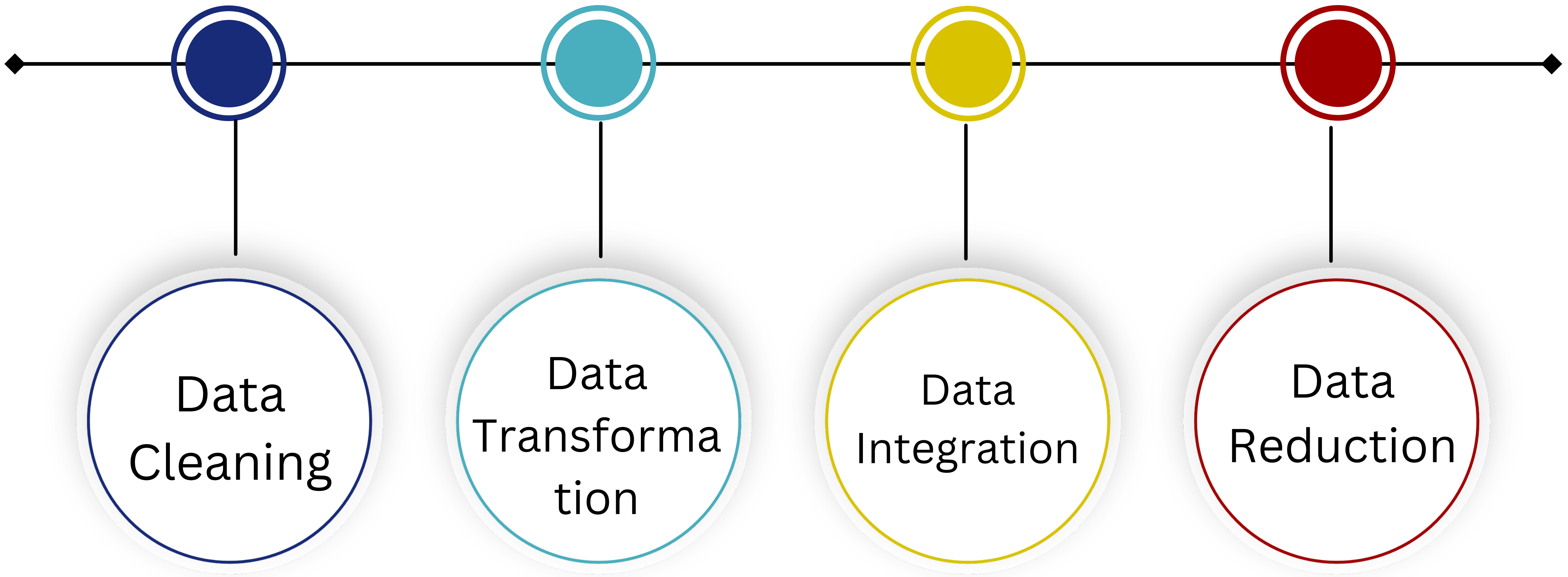
Literature Review - 2

John
&
Smith

Statistical and Machine Learning Techniques for Analysis: To analyze the factors influencing productivity, Smith and Jones (2020) suggest regression models and machine learning approaches as valuable tools.

he **Journal of Management Studies** published by **Wiley-Blackwell**
on behalf of the **Society for the Advancement of Management Studies**

Data Pre -Processing



Dataset Review:

Dataset: The dataset is a regression dataset with 1197 rows and 15 columns

Source Code: https://archive.ics.uci.edu/dataset/597/productivity+prediction+of+garment+employees

Variable:

CATEGORICAL VARIABLES	CONTINUOUS VARIABLES	CONTINOUS VARIABLES
day	team	Idle_time
quarter	targeted_productivity	Idle_men
department	Smv	no_of_workers
	Wip	actual_productivity
	over_time	no_of_style_change
	incentive	

About the dataset:

garment workers' productivity dataset is a detailed collection of production-related data from a garment factory

dataset is used for regression analysis because the target variable, actual_productivity, is a continuous numeric variable.

index	date	quarter	department	day	team	targeted_productivity	smv	wip
0	1/1/2015	Quarter1	sweing	Thursday	8	0.8	26.16	1108.0
1	1/1/2015	Quarter1	finishing	Thursday	1	0.75	3.94	NaN
2	1/1/2015	Quarter1	sweing	Thursday	11	0.8	11.41	968.0
3	1/1/2015	Quarter1	sweing	Thursday	12	0.8	11.41	968.0
4	1/1/2015	Quarter1	sweing	Thursday	6	0.8	25.9	1170.0
5	1/1/2015	Quarter1	sweing	Thursday	7	0.8	25.9	984.0
6	1/1/2015	Quarter1	finishing	Thursday	2	0.75	3.94	NaN
7	1/1/2015	Quarter1	sweing	Thursday	3	0.75	28.08	795.0
8	1/1/2015	Quarter1	sweing	Thursday	2	0.75	19.87	733.0
9	1/1/2015	Quarter1	sweing	Thursday	1	0.75	28.08	681.0
10	1/1/2015	Quarter1	sweing	Thursday	9	0.7	28.08	872.0
11	1/1/2015	Quarter1	sweing	Thursday	10	0.75	19.31	578.0
12	1/1/2015	Quarter1	sweing	Thursday	5	0.8	11.41	668.0
13	1/1/2015	Quarter1	finishing	Thursday	10	0.65	3.94	NaN
14	1/1/2015	Quarter1	finishing	Thursday	8	0.75	2.9	NaN
15	1/1/2015	Quarter1	finishing	Thursday	4	0.75	3.94	NaN
16	1/1/2015	Quarter1	finishing	Thursday	7	0.8	2.9	NaN
17	1/1/2015	Quarter1	sweing	Thursday	4	0.65	23.69	861.0
18	1/1/2015	Quarter1	finishing	Thursday	11	0.7	4.15	NaN
19	1/3/2015	Quarter1	finishing	Saturday	4	0.8	4.15	NaN
20	1/3/2015	Quarter1	finishing	Saturday	11	0.75	2.9	NaN
21	1/3/2015	Quarter1	finishing	Saturday	9	0.8	4.15	NaN
22	1/3/2015	Quarter1	finishing	Saturday	3	0.75	3.94	NaN
23	1/3/2015	Quarter1	finishing	Saturday	1	0.8	3.94	NaN
24	1/3/2015	Quarter1	sweing	Saturday	1	0.8	28.08	772.0

over_time	incentive	idle_time	idle_men	no_of_style_change	no_of_workers	actual_productivity
7080	98	0.0	0	0	59.0	0.940725424
960	0	0.0	0	0	8.0	0.8865
3660	50	0.0	0	0	30.5	0.800570492
3660	50	0.0	0	0	30.5	0.800570492
1920	50	0.0	0	0	56.0	0.800381944
6720	38	0.0	0	0	56.0	0.800125
960	0	0.0	0	0	8.0	0.755166667
6900	45	0.0	0	0	57.5	0.753683478
6000	34	0.0	0	0	55.0	0.753097531
6900	45	0.0	0	0	57.5	0.750427826
6900	44	0.0	0	0	57.5	0.721126957
6480	45	0.0	0	0	54.0	0.712205247
3660	50	0.0	0	0	30.5	0.707045902
960	0	0.0	0	0	8.0	0.705916667
960	0	0.0	0	0	8.0	0.676666667
2160	0	0.0	0	0	18.0	0.593055556
960	0	0.0	0	0	8.0	0.540729167
7200	0	0.0	0	0	60.0	0.52118
1440	0	0.0	0	0	12.0	0.436326389
6600	0	0.0	0	0	20.0	0.988024691
5640	0	0.0	0	0	17.0	0.987880435
960	0	0.0	0	0	8.0	0.956270833
1560	0	0.0	0	0	8.0	0.945277778
960	0	0.0	0	0	8.0	0.902916667
6300	50	0.0	0	0	56.5	0.800725314

Data Cleaning

1

Converting the ‘day’ column:

day
Thursday
Thursday
Thursday
Thursday
Thursday
Thursday
Thursday
Thursday
Thursday
Thursday
Thursday
Thursday
Thursday
Thursday
Thursday
Thursday
Thursday
Thursday
Thursday
Saturday
Saturday
Saturday
Saturday
Saturday
Saturday

Here we converted the day column into binary values
Monday - Friday as 0
Saturday and Sunday as 1

Categorical to Binary

day_binary
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
1
1
1
1
1
1

2

Checking For Null Values:

Here we check for any null values in the dataset and replace them using mean , median or mode or remove the column if necessary

	0
quarter	0
department	0
team	0
targeted_productivity	0
smv	0
wip	506
over_time	0
incentive	0
idle_time	0
idle_men	0
no_of_style_change	0
no_of_workers	0
actual_productivity	0
day_binary	0

dtype: int64

Here we removed the ‘wip’ column since the Null values are more

	0
quarter	0
department	0
team	0
targeted_productivity	0
smv	0
over_time	0
incentive	0
idle_time	0
idle_men	0
no_of_style_change	0
no_of_workers	0
actual_productivity	0
day_binary	0

dtype: int64

3

Assigning Dummy Variables

-- To perform dummy variable encoding we divided the data into two sets continuous and categorical data
-- The categorical variables are encoded into continuous variablesusing the dummy variables
-- Remaining the columns are

Original Variables	Renamed Variables
quarter	quarter_Quarter1
	quarter_Quarter2
	quarter_Quarter3
	quarter_Quarter4
	quarter_Quarter5

Original Variables	Renamed Variables
department	department_finishing
	department_sweing

Converting the ‘quarter’ and ‘department’ columns into binary

Here we converted the boolean data which we got after dummy variable encoding , into binary value

Quarter column:

quarter
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1
Quarter1

After dummy variables encoding

quarter_Quarter2	quarter_Quarter3	quarter_Quarter4	quarter_Quarter5
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false
false	false	false	false

Boolean to Binary

quarter_Quarter2	quarter_Quarter3	quarter_Quarter4	quarter_Quarter5
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Department column:

department
sweing
finishing
sweing
sweing
sweing
sweing
finishing
sweing
sweing
sweing
sweing
sweing
sweing
finishing
finishing
finishing
finishing
sweing
finishing
finishing
finishing
finishing
finishing
finishing
sweing

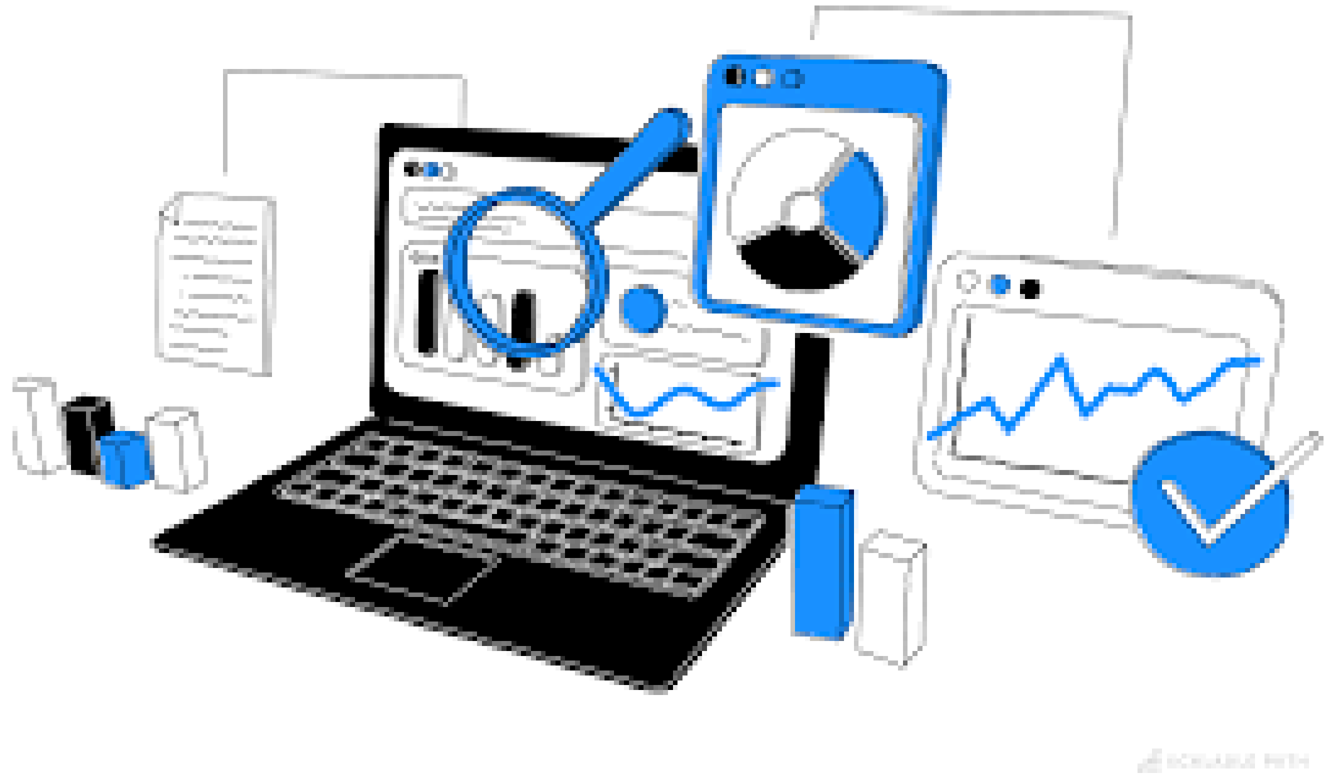
After Dummy variables encoding

department_finishing	department_sweing
false	true
true	false
false	true
false	true
false	true
false	true
true	false
false	true
false	true
false	true
false	true
false	true
false	true
true	false
true	false
true	false
true	false
false	true
false	false
true	false
true	false
true	false
true	false
true	false
false	true

Boolean to binary

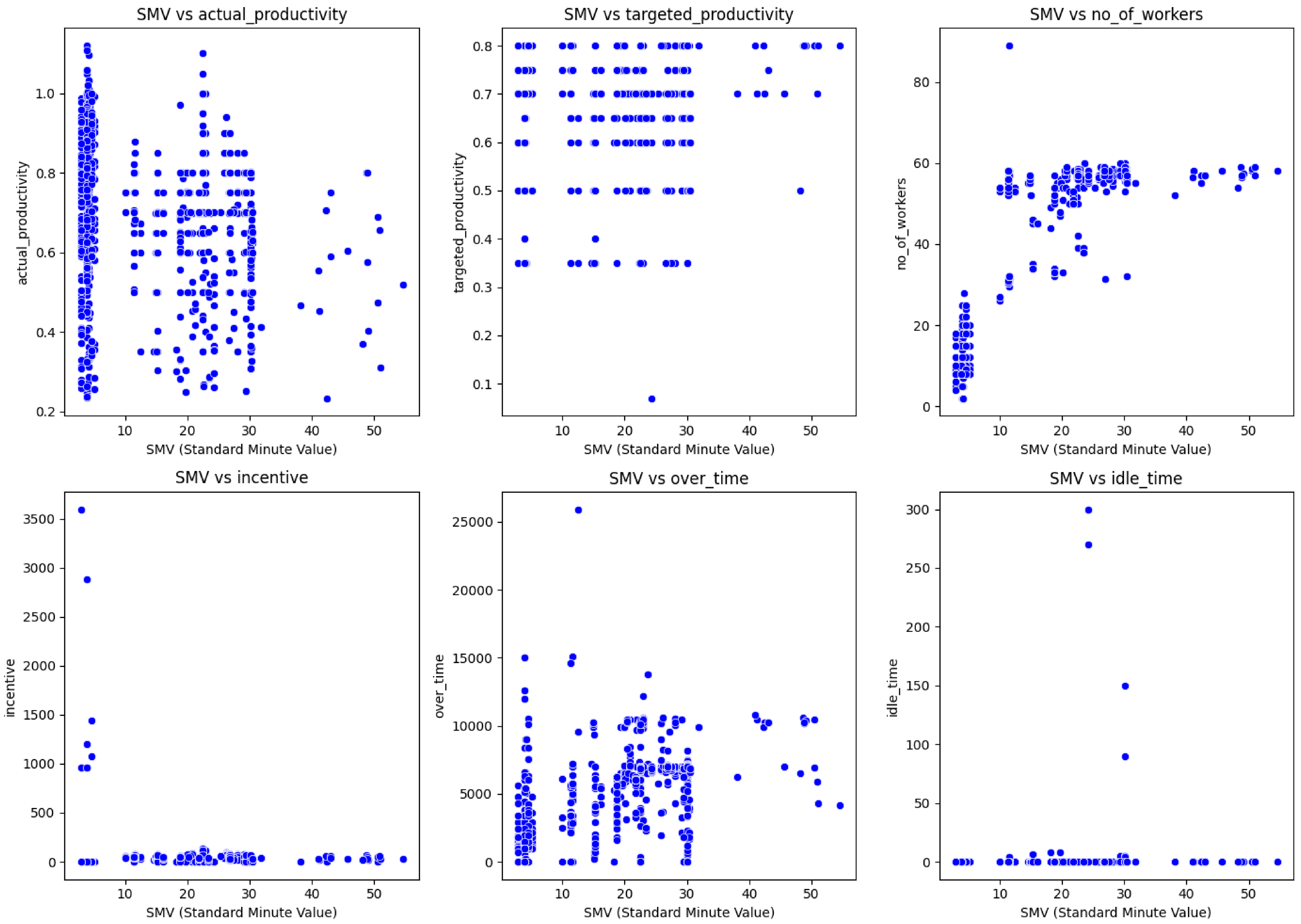
department_finishing	department_sweing
0	1
1	0
0	1
0	1
0	1
0	1
1	0
0	1
0	1
0	1
0	1
0	1
0	1
1	0
1	0
1	0
1	0
0	1
0	0
1	0
1	0
1	0
1	0
1	0
0	1

Exploratory Data Analysis



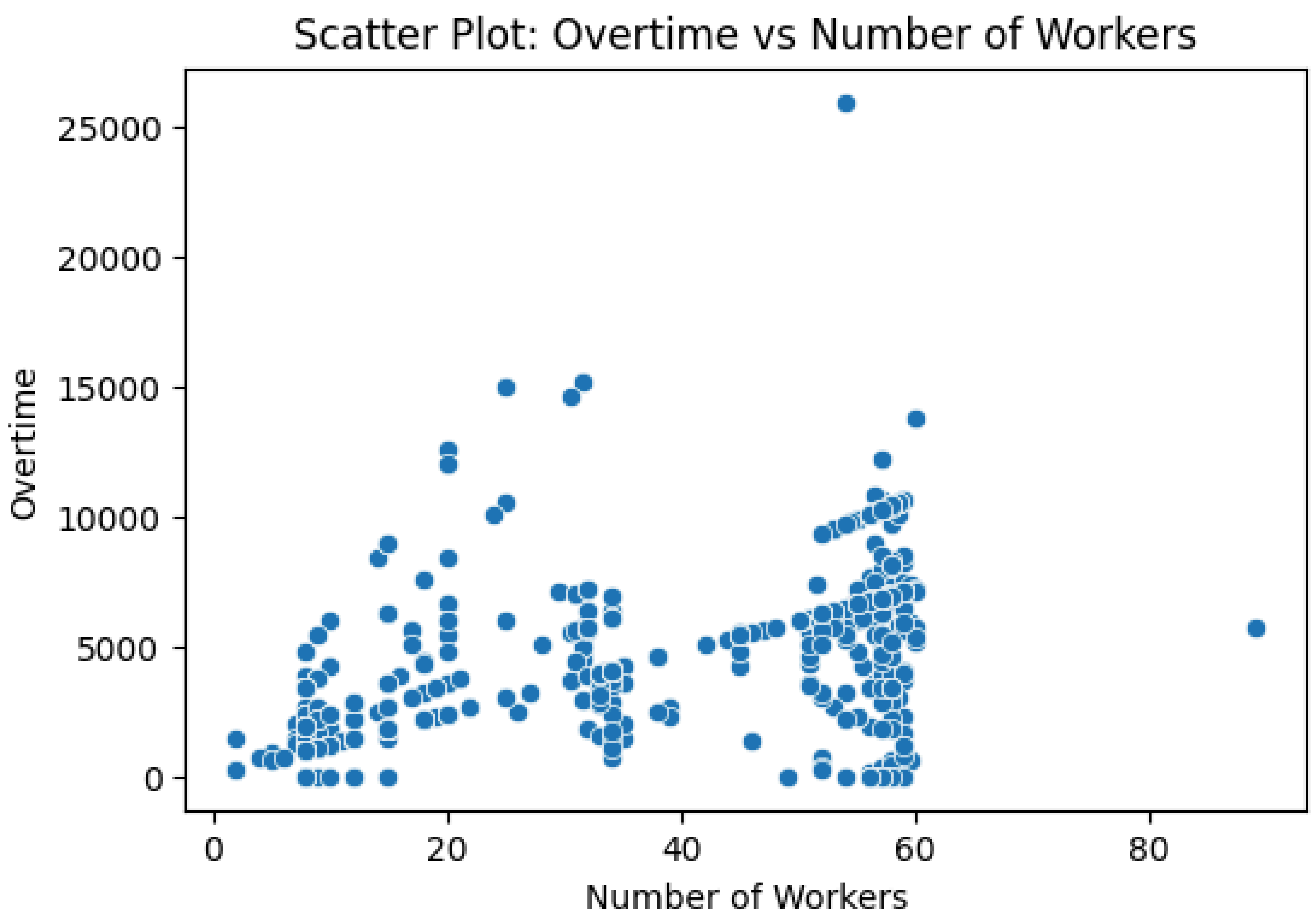
Scatter plot:

Analysis of Standard Minute value(SMV) based on each attribute , SMV as a target variable comparing with each of the columns



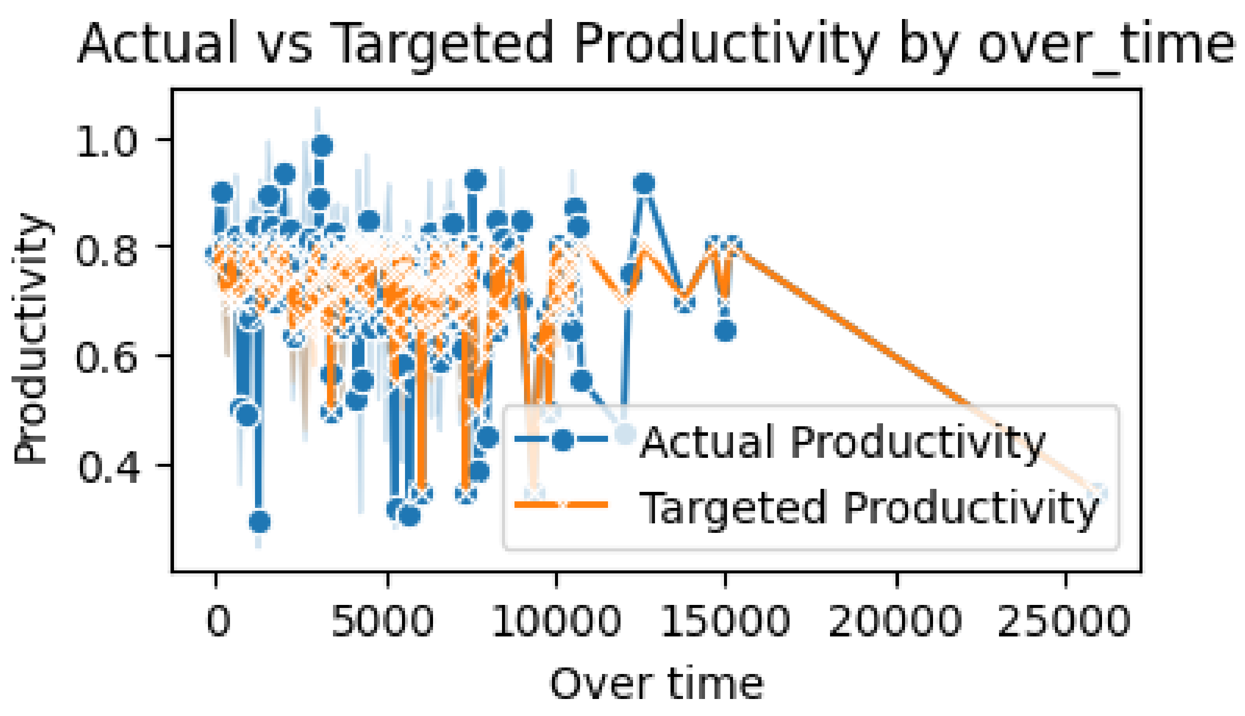
Scatter plot:

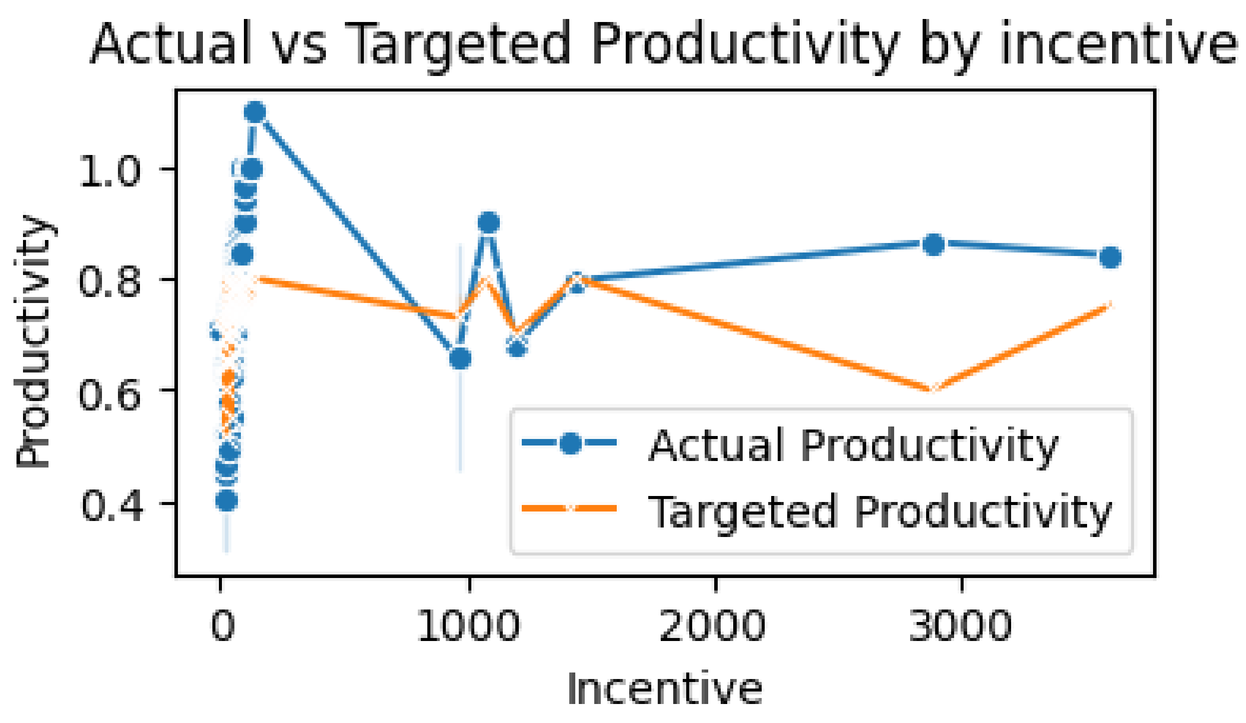
Checking the productivity based on the no of workers and the overtime

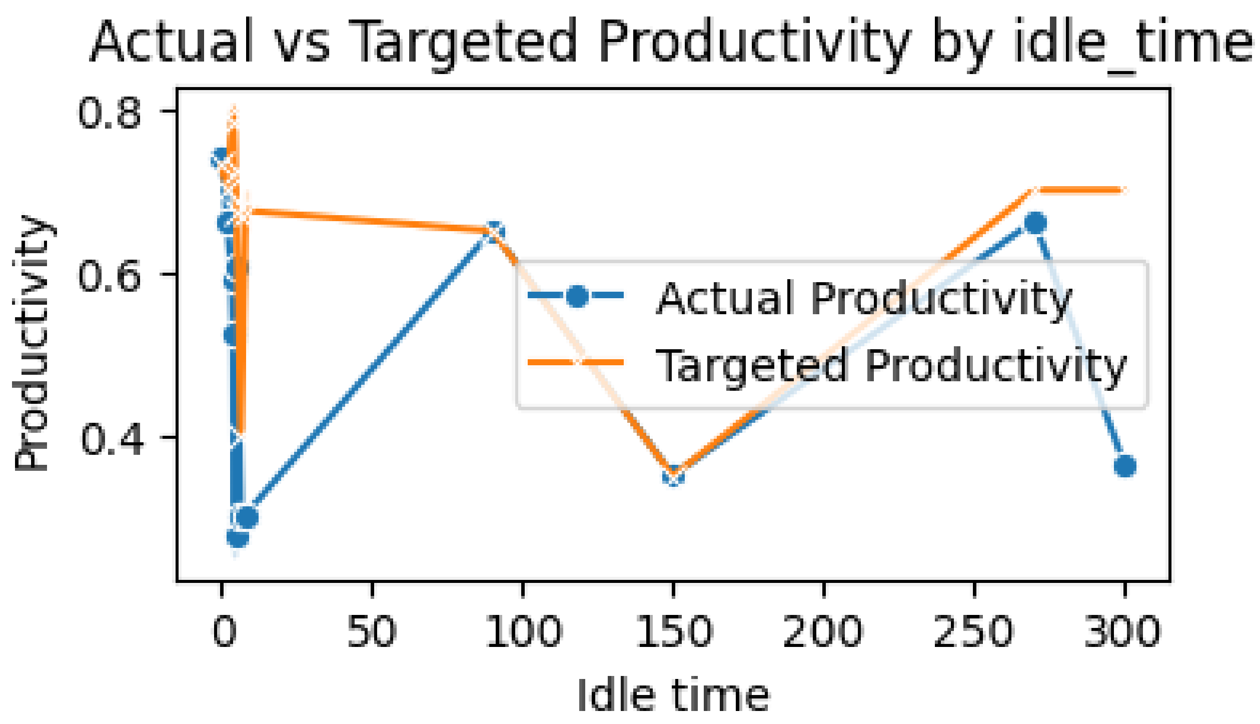


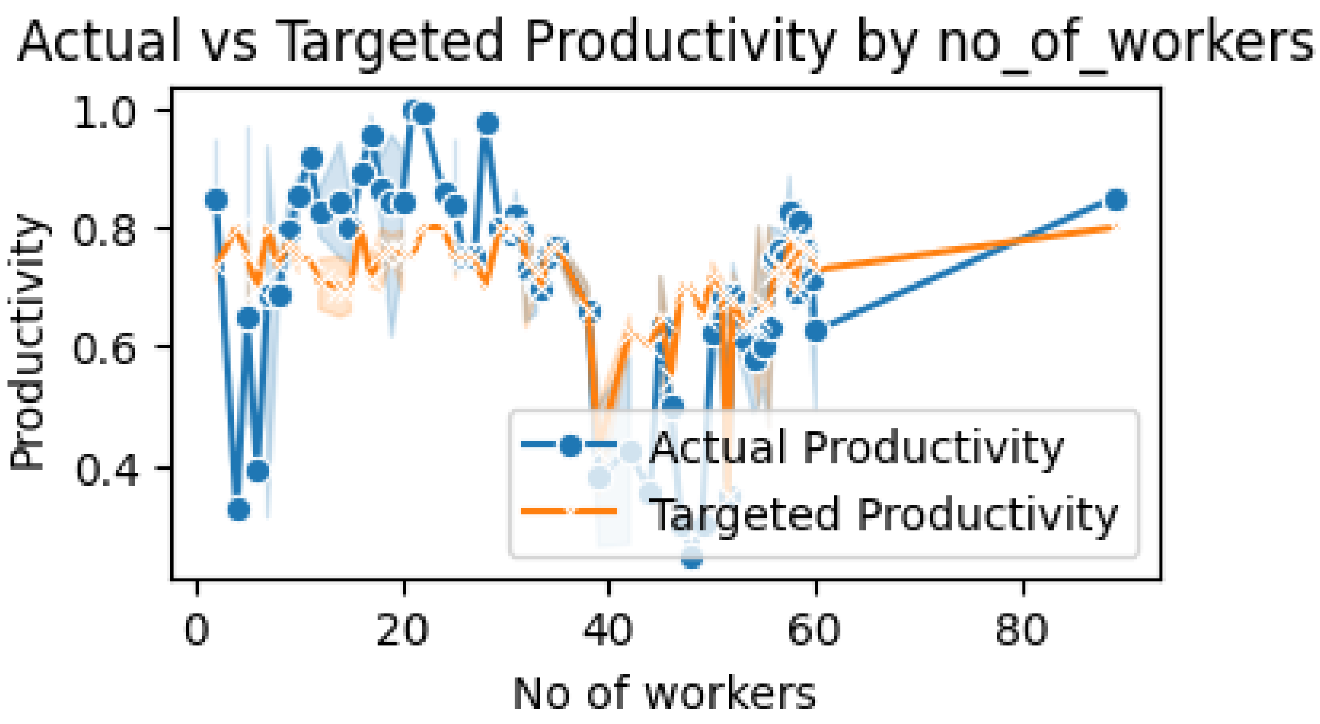
Line Plot:

Comparing Actual productivity and targeted productivitywith each of the columns using line plots



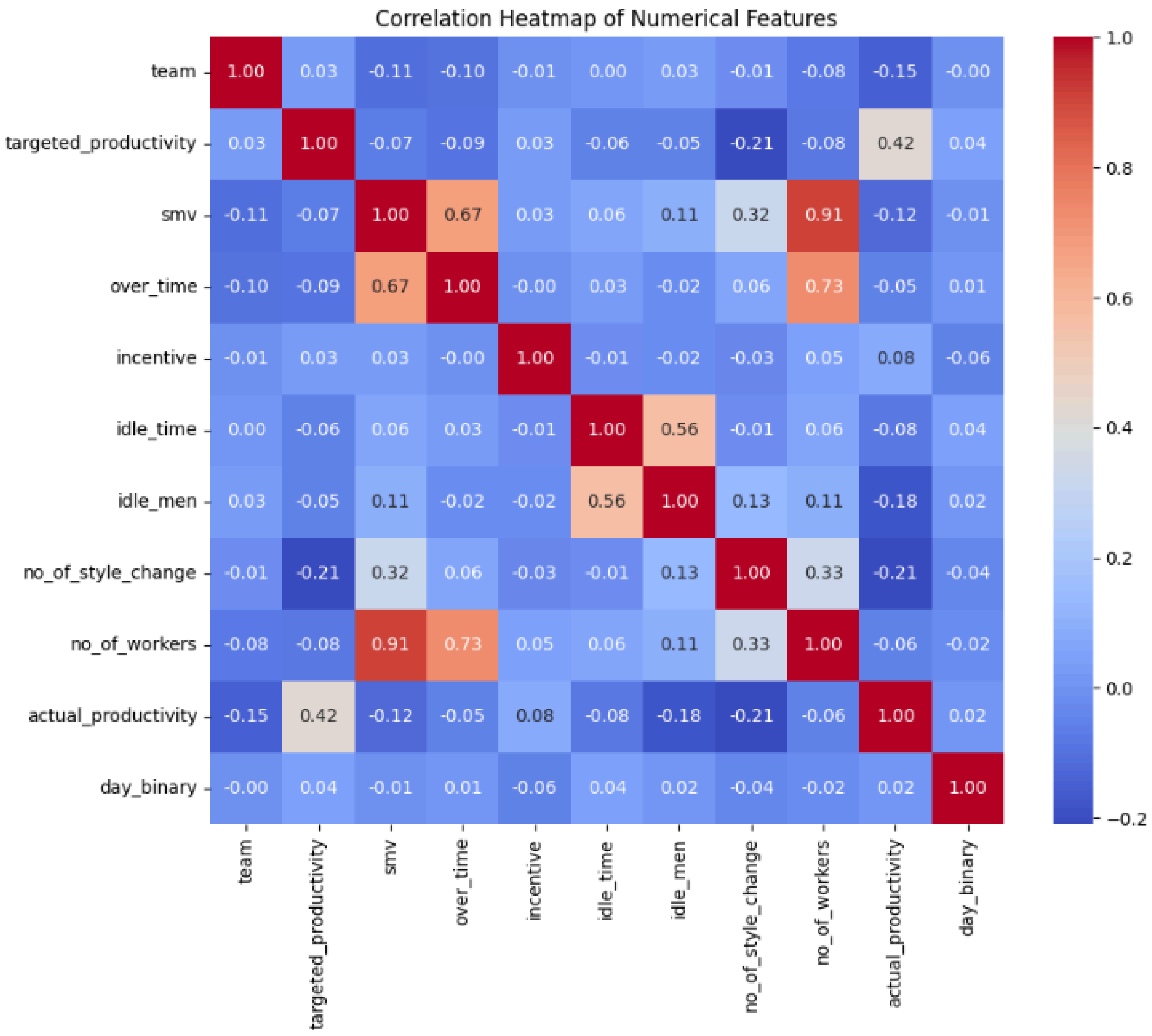




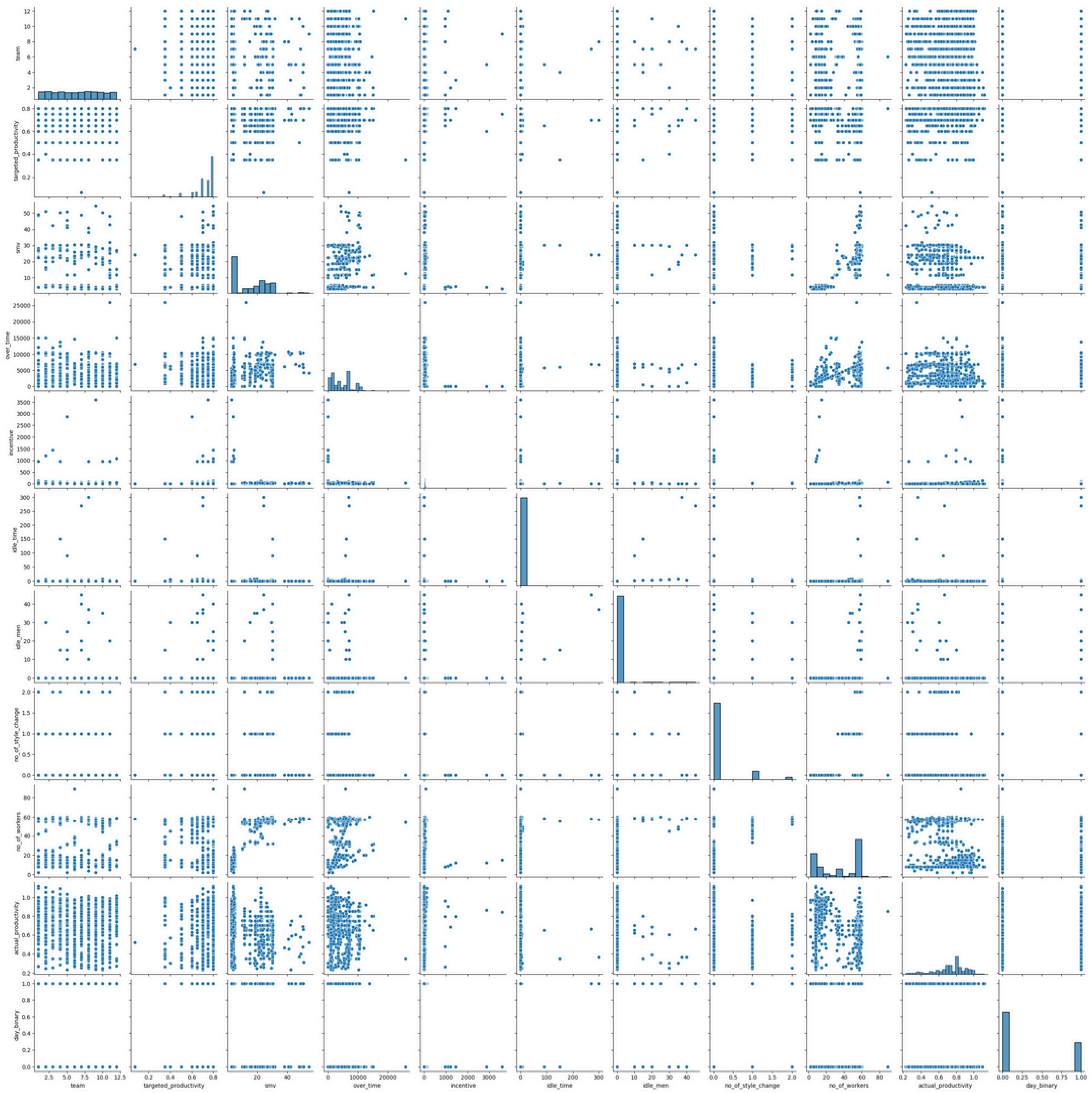


Correlation Matrix:

The correlation gives the overview of the complete dataset, here actual productivity have high positive correlation



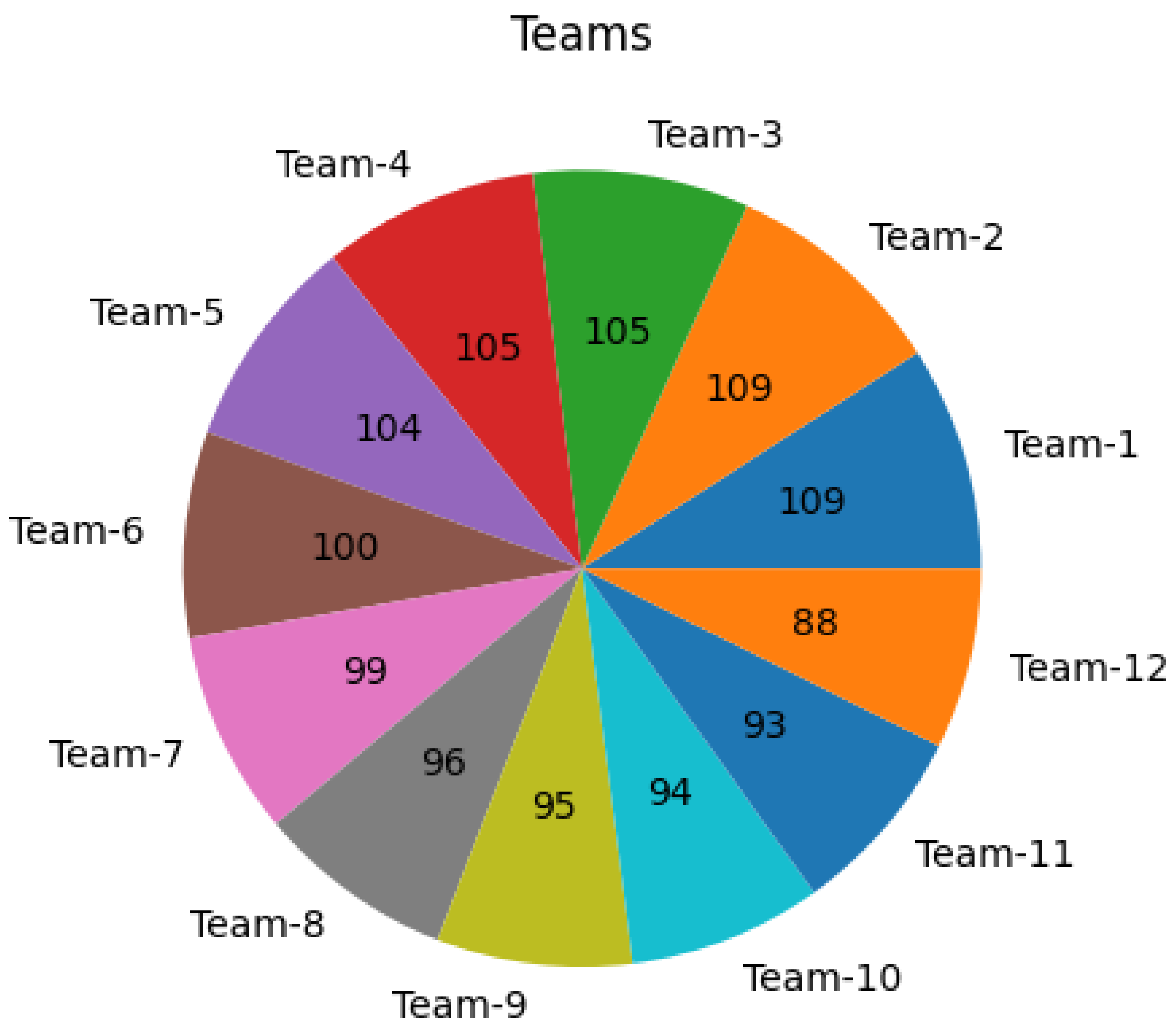
Pair plot:



- The scatterplot matrix provides an overview of the relationships between pairs of numerical features in your dataset

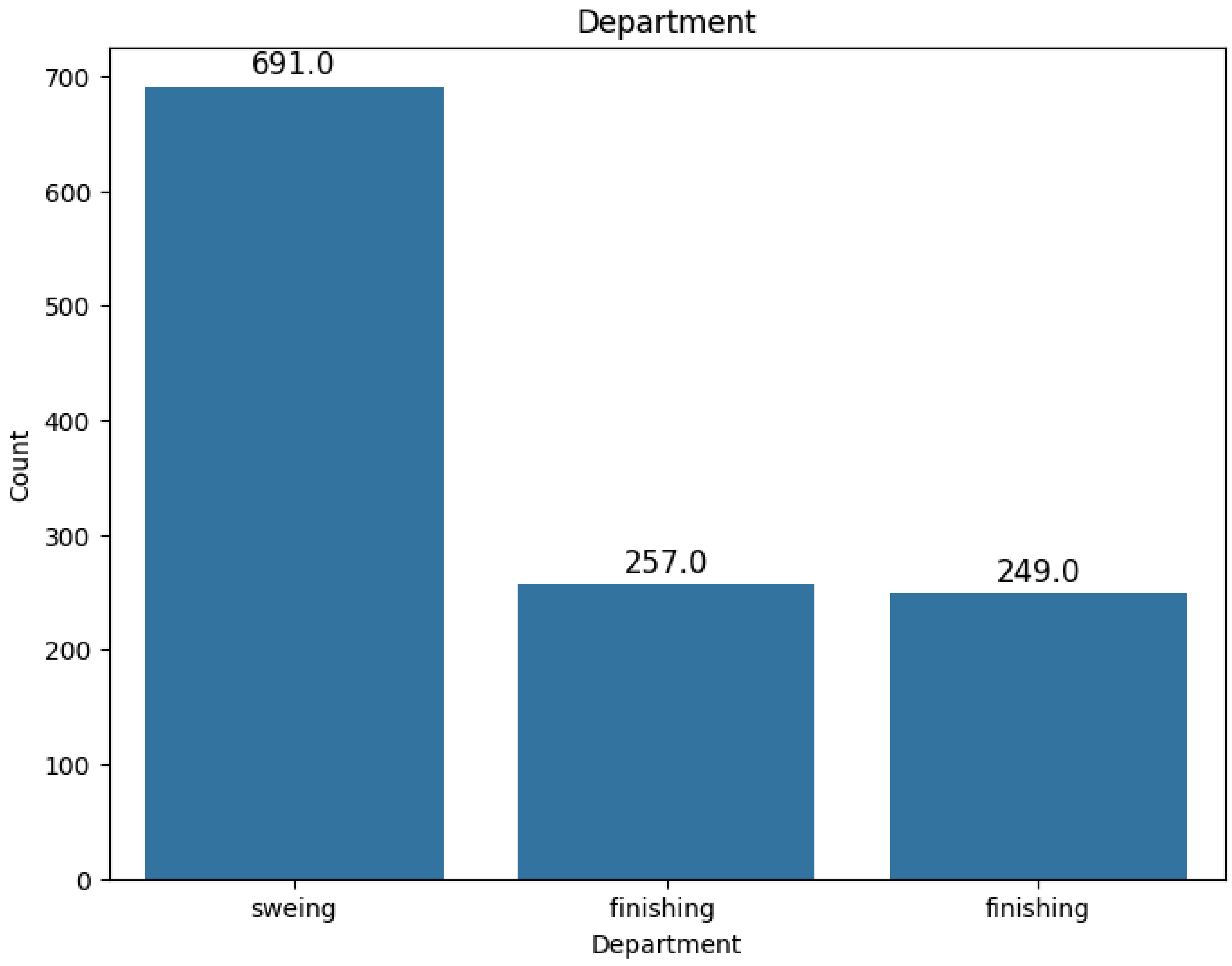
It shows potential correlations and patterns among variables, such as targeted_productivity, actual_productivity, smv, over_time, and other factors.

Pie Charts:

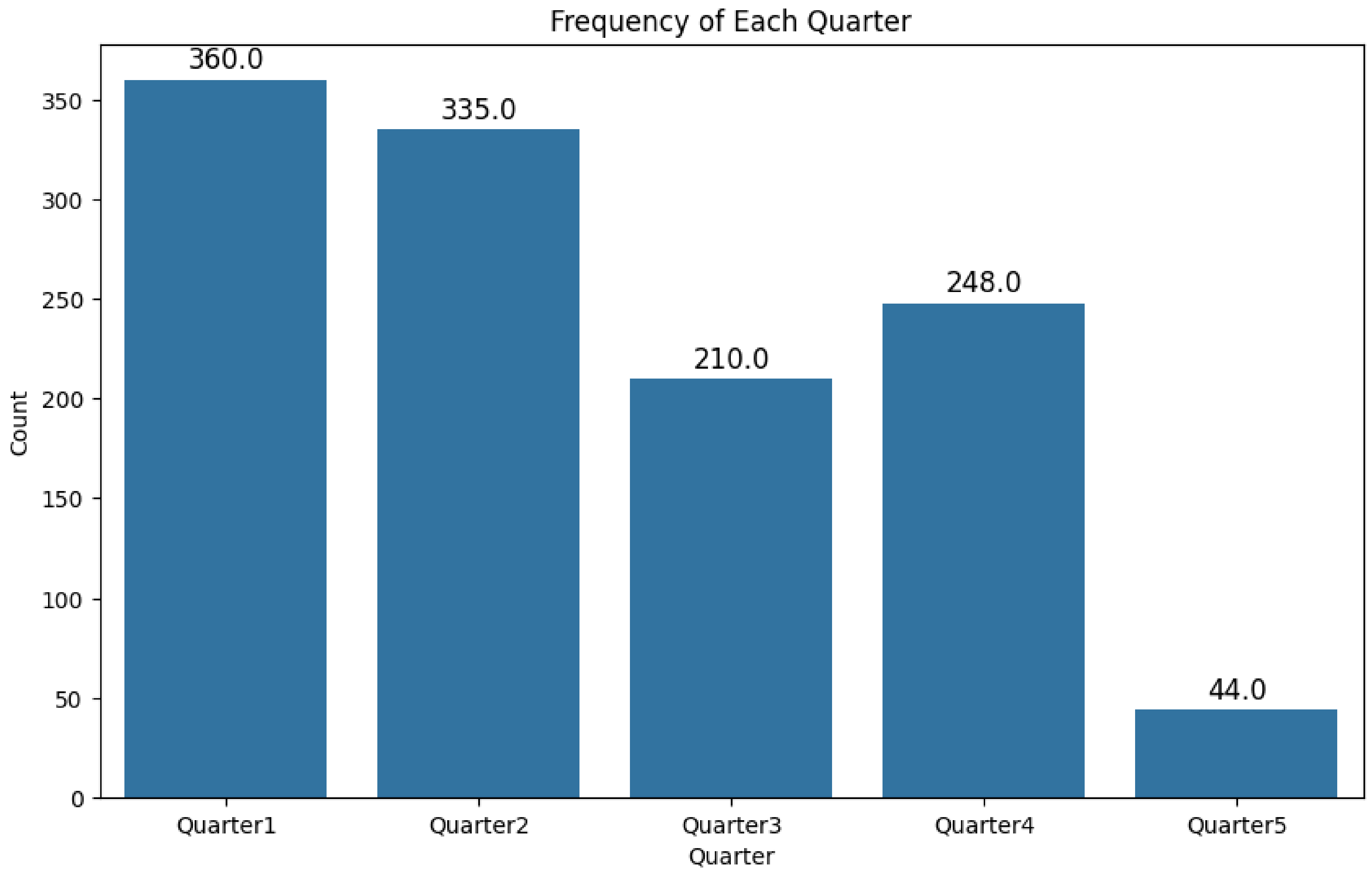

Teams
Team-1
109
Team-2
109
Team-3
105
Team-4
105
Team-5
104
Team-6
100
Team-7
99
Team-8
96
Team-9
95
Team-10
94
Team-11
93
Team-12
88

Bar plots:

Department Column



Quarter column



Multicollinearity Checking

Checking the multicollinearity of the dataset using variance inflation factor . here the columns with high correlation are removed,the columns are :no_of_workers','smv','targeted_productivity','department_sweing

	variables	VIF
0	team	4.732500
1	targeted_productivity	12.350808
2	smv	18.036447
3	over_time	7.166048
4	incentive	1.103324
5	idle_time	1.492213
6	idle_men	1.563800
7	no_of_style_change	1.566141
8	no_of_workers	48.515423
9	day_binary	1.513058
10	quarter_Quarter2	1.935032
11	quarter_Quarter3	1.610072
12	quarter_Quarter4	1.795143
13	quarter_Quarter5	1.148189
14	department_finishing	2.080977
15	department_sweing	23.559402

no _of_workers column is removed

	variables	VIF
0	team	4.655391
1	targeted_productivity	10.906711
2	smv	14.244128
3	over_time	6.154119
4	incentive	1.099404
5	idle_time	1.492139
6	idle_men	1.559137
7	no_of_style_change	1.497347
8	day_binary	1.511988
9	quarter_Quarter2	1.934861
10	quarter_Quarter3	1.609993
11	quarter_Quarter4	1.794113
12	quarter_Quarter5	1.147719
13	department_finishing	2.080922
14	department_sweing	13.271302

smv column is removed

	variables	VIF
0	team	4.394002
1	targeted_productivity	9.970066
2	over_time	5.747457
3	incentive	1.099397
4	idle_time	1.492062
5	idle_men	1.554649
6	no_of_style_change	1.449999
7	day_binary	1.511863
8	quarter_Quarter2	1.933154
9	quarter_Quarter3	1.609669
10	quarter_Quarter4	1.793995
11	quarter_Quarter5	1.146136
12	department_finishing	2.080529
13	department_sweing	6.950013

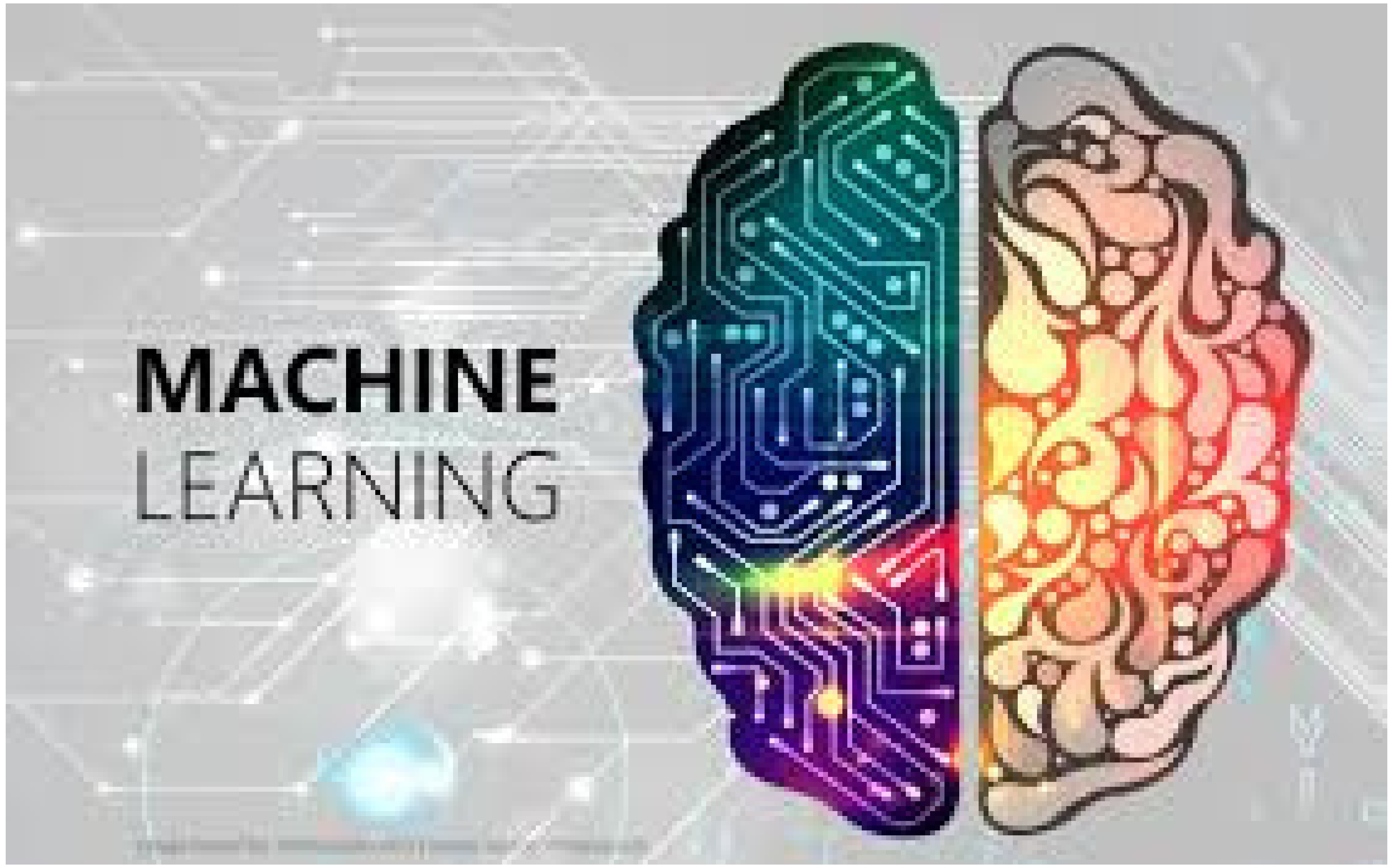
targeted_productivity column is remoned

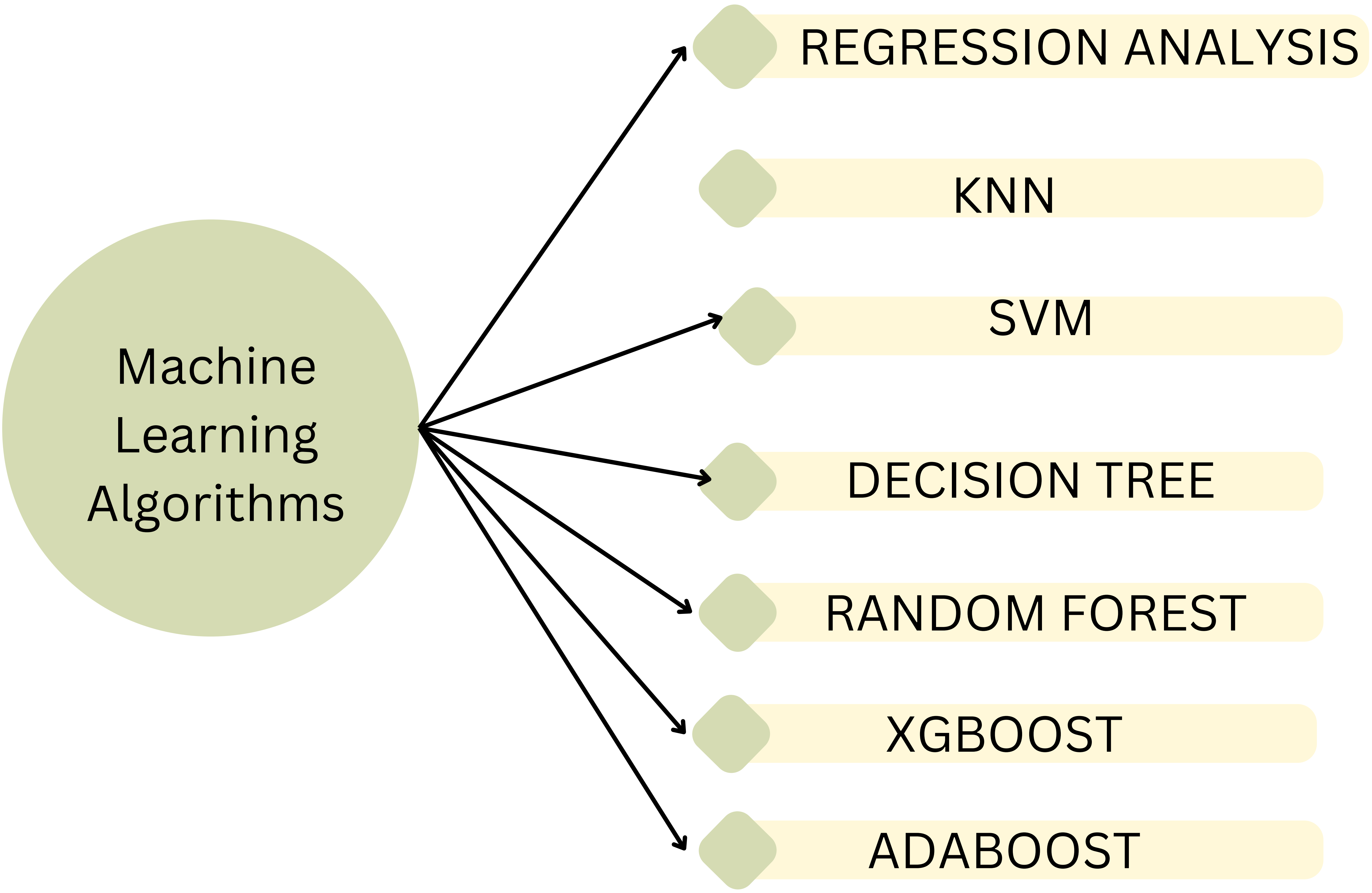
	variables	VIF
0	team	3.166119
1	over_time	5.486969
2	incentive	1.084794
3	idle_time	1.491642
4	idle_men	1.553609
5	no_of_style_change	1.439650
6	day_binary	1.456817
7	quarter_Quarter2	1.709498
8	quarter_Quarter3	1.511013
9	quarter_Quarter4	1.678291
10	quarter_Quarter5	1.139899
11	department_finishing	1.679275
12	department_sweing	6.511808

department_sweing column is removed

	variables	VIF
0	team	2.824925
1	over_time	2.356929
2	incentive	1.082418
3	idle_time	1.491067
4	idle_men	1.539909
5	no_of_style_change	1.272883
6	day_binary	1.454919
7	quarter_Quarter2	1.708352
8	quarter_Quarter3	1.510918
9	quarter_Quarter4	1.671193
10	quarter_Quarter5	1.127056
11	department_finishing	1.377880

MACHINE LEARNING ALGORITHMS




Machine Learning Algorithms
REGRESSION ANALYSIS
KNN
SVM
DECISION TREE
RANDOM FOREST
XGBOOST
ADABOOST

80 - 20 Train_test_split

Algorithm	model1 (r2 score)	model 2 (r2 score)	model 1 MAE	model 2 MAE
Multipe Regreson	-0.3144	-0.0596	0.1281	0.1662
KNN	0.2676	0.1599	0.1057	0.1117
SVM	0.0255	0.0368	0.1323	0.1228
Decision Tree	0.4478	0.3747	0.0777	0.0846
Random Forest	0.5900	0.4023	0.0689	0.0884
XGBoost	0.5785	0.3928	0.0692	0.0826
ADABoost	0.2815	0.2615	0.1018	0.1020

- r2 score: r_squared is a statistical measure used to evaluate the performance of a regression model
- $R^2 = 1$: Indicates that the regression model perfectly fits the data
- $R^2 = 0$: Suggests that the model does not explain any variability in the dependent variable

- MAE - “Mean Absolute Error” is a metric used to evaluate the accuracy of a regression model

model 1 before applying VIF
model 2 after applying VIF

75 - 25 Train_test_Split

Algorithm	model1 (r2 score)	model 2 (r2 score)	model 1 MAE	model 2 MAE
Multiple Regression	-0.4605	-0.0858	0.1595	0.1320
KNN	0.2145	0.1730	0.1089	0.1121
SVM	0.0290	0.02736	0.1228	0.1236
Decision Tree	0.2099	0.3258	0.0955	0.0879
Random Forest	0.5092	0.3210	0.0715	0.0884
XGBoost	0.5019	0.3325	0.0719	0.0872
ADABoost	0.4215	0.2497	0.0928	0.1020

- r2 score: r_squared is a statistical measure used to evaluate the performance of a regression model
- $R^2 = 1$: Indicates that the regression model perfectly fits the data
- $R^2 = 0$: Suggests that the model does not explain any variability in the dependent variable

- MAE - “Mean Absolute Error” is a metric used to evaluate the accuracy of a regression model

model 1 before applying VIF
model 2 after applying VIF

70 - 30 Train_test_split

Algorithm	model1 (r2 score)	model 2 (r2 score)	model 1 MAE	model 2 MAE
Multiple Regression	-0.2538	-0.0201	0.1628	0.1312
KNN	0.2123	0.1353	0.112	0.1175
SVM	0.0354	0.0213	0.1300	0.1277
Decision Tree	0.2320	0.1315	0.1010	0.1077
Random Forest	0.4885	0.3546	0.0746	0.0879
XGBoost	0.4470	0.3654	0.0824	0.0872
ADABoost	0.3394	0.2670	0.1018	0.1068

- r2 score: r_squared is a statistical measure used to evaluate the performance of a regression model
- $R^2 = 1$: Indicates that the regression model perfectly fits the data
- $R^2 = 0$: Suggests that the model does not explain any variability in the dependent variable

- MAE - “Mean Absolute Error” is a metric used to evaluate the accuracy of a regression model

model 1 before applying VIF
model 2 after applying VIF

60 - 40 Train_test_split

Algorithm	model1 (r2 score)	model 2 (r2 score)	model 1 MAE	model 2 MAE
Multiple Regression	-0.2161	-0.8512	0.1518	0.1578
KNN	0.1595	0.1162	0.1264	0.1171
SVM	0.0195	0.0097	0.1415	0.1288
Decision Tree	0.2749	0.2784	0.0899	0.0896
Random Forest	0.5094	0.3180	0.0773	0.0891
XGBoost	0.4276	0.3215	0.0796	0.0891
ADABoost	0.4215	0.2092	0.1073	0.1102

- r2 score: r_squared is a statistical measure used to evaluate the performance of a regression model
- $R^2 = 1$: Indicates that the regression model perfectly fits the data
- $R^2 = 0$: Suggests that the model does not explain any variability in the dependent variable

- MAE - "Mean Absolute Error" is a metric used to evaluate the accuracy of a regression model

model 1 before applying VIF
model 2 after applying VIF

Algorithms Comparison

60 - 40 Train_test_split before vif

Algorithm	model1 (r2 score)	model 1 MAE
Multiple Regression	-0.2161	0.1518
KNN	0.1595	0.1264
SVM	0.0195	0.1415
Decision Tree	0.2749	0.0899
Random Forest	0.5094	0.0773
XGBoost	0.4276	0.0796
ADABoost	0.4215	0.1073

60 - 40 Train_test_split after vif

Algorithm	model 2 (r2 score)	model 2 MAE
Multiple Regression	-0.8512	0.1578
KNN	0.1162	0.1171
SVM	0.0097	0.1288
Decision Tree	0.2784	0.0896
Random Forest	0.3280	0.0891
XGBoost	0.3115	0.0891
ADABoost	0.2092	0.1102

NEURAL NETWORKS

Before Multicollinearity Checking

Train_test	Architecture	Optimizer	Epochs	MAE	loss	Precision	Re
80 - 20	30-20-18-8-1	Adam	150	0.1241	0.1241	0.1250	0.1
75 - 25	30-20-18-8-1	Adam	150	0.1231	0.1231	0.1255	0.1
70 - 30	30-20-18-8-1	Adam	150	0.1258	0.1258	0.13101	0.13
60 - 40	30-20-18-8-1	Adam	150	0.1293	0.1293	0.1321	0.1

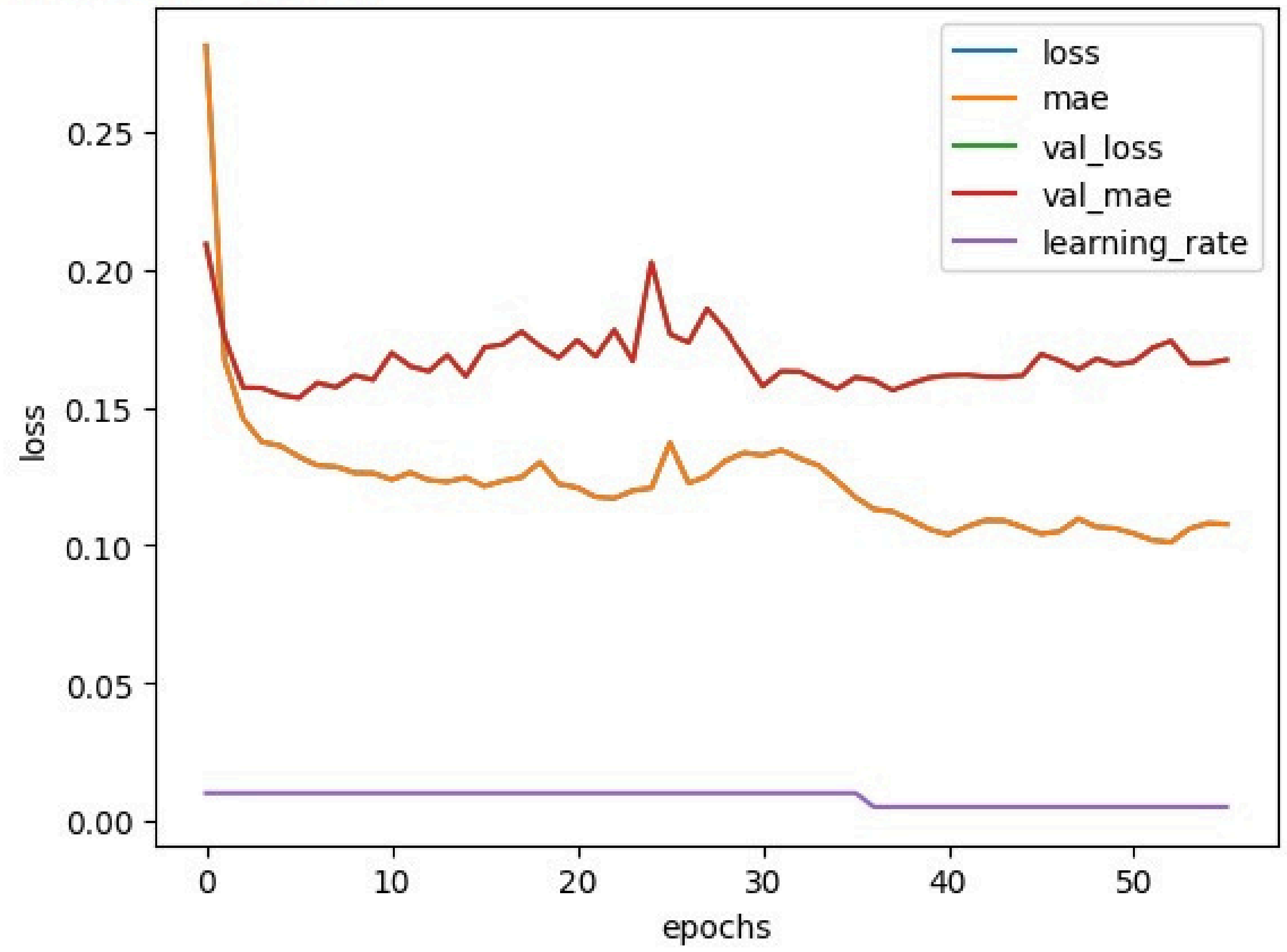
NEURAL NETWORKS

After Multicollinearity Checking

Train_test	Architecture	Optimizer	Epochs	MAE	loss	Precision	Recall
80 - 20	30-20-18-8-1	Adam	150	0.1231	0.1231	0.12472	0.12472
75 - 25	30-20-18-8-1	Adam	150	0.1747	0.1747	0.1753	0.1753
70 - 30	30-20-18-8-1	Adam	150	0.1672	0.1672	0.1709	0.17093
60 - 40	30-20-18-8-1	Adam	150	0.1493	0.1493	0.1522	0.1522

Neural Network plot for observation

Text(0.5, 0, 'epochs')



Train_test_split	**75 - 25**
Architecture	30-20-18-8-1
Optimizer	Adam
Epochs	150

Summary

- This project aimed to analyze and predict the productivity of garment workers by using a comprehensive dataset containing various features related to worker output and operational conditions.

- So for the actual data we have the Random Forest showing the best result with a r2 score of 0.5094 and MAE of 0.0773

- As for the data after multicollinearity checking , XGBoost algorithm shows the best result with r2 score of 0.3115 , MAE of 0.0891 and the Random Forest is showing the next best result

- key stages included examining multicollinearity using the variance inflation factor (VIF) to ensure the reliability of the features selected for modelling

Insights

- Analysis revealed that number of workers, overtime, targeted productivity, and incentives are significant drivers of actual productivity.
- Checking multicollinearity using VIF helped identify highly correlated features, ensuring that only reliable features were used for predictive modeling.
- The analysis suggested that adjusting workforce size and strategically applying incentives can positively impact productivity.
- These insights provide a foundation for data driven decision-making to optimize labor management and improve productivity outcomes in the garment industry.

Future Scope:

1.) Incorporating External Data: Integrating external data like weather conditions, fabric types, or economic indicators could provide additional insights that might influence worker productivity.

2.)Scalability and Deployment: Transforming the model into a deployable tool that can be scaled across multiple manufacturing plants could help in real-world decision-making.

3.)Predictive Maintenance: Leveraging predictive analytics for equipment and workflow management could help minimize idle time and improve operational efficiency.

Work Distribution

Vamshi	Collecting information about the garment worker's productivity
Nithin Chandra	Data Pre-Processing
Sri Vani	Exploratory Data Analysis
Sravan Kumar	Implement Machine Learning Algorithms

Colob Notebook

Thank You

SRAVAN KUMAR - 107222546009
VAMSHI - 107222546010
SRI VANI - 107222546011
NITHIN CHANDRA - 107222546012

Appendix


Productivity
Production
of Garment
Employees


Productivity

```
import pandas as pd
import seaborn as sns
import statsmodels.formula.api as smf
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
```

- **importing libraries**

uploading files to colab notebook

- **uploading the dataset**

```
from google.colab import files
uploaded=files.upload()
```

Choose Files garments_...oductivity.csv

- **garments_worker_productivity.csv**(text/csv) - 94933 bytes, last modified: 8/26/2024 - 100% done

```
Saving garments_worker_productivity.csv to garments_worker_productivity (2).csv
```

- **loading the dataset**

```
data=pd.read_csv('/content/garments_worker_productivity.csv')
data
```

date	quarter	department	day	team	targeted_productivity	smv	wip	over_time	incentive	idle_time	idle_men	no_of_style_change	no_of_workers	actual_productivity
1/1/2015	Quarter1	sweing	Thursday	8	0.80	26.16	1108.0	7080	98	0.0	0	0	59.0	0.940725
1/1/2015	Quarter1	finishing	Thursday	1	0.75	3.94	NaN	960	0	0.0	0	0	8.0	0.886500
1/1/2015	Quarter1	sweing	Thursday	11	0.80	11.41	968.0	3660	50	0.0	0	0	30.5	0.800570
1/1/2015	Quarter1	sweing	Thursday	12	0.80	11.41	968.0	3660	50	0.0	0	0	30.5	0.800570
1/1/2015	Quarter1	sweing	Thursday	6	0.80	25.90	1170.0	1920	50	0.0	0	0	56.0	0.800382
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
11/2015	Quarter2	finishing	Wednesday	10	0.75	2.90	NaN	960	0	0.0	0	0	8.0	0.628333
11/2015	Quarter2	finishing	Wednesday	8	0.70	3.90	NaN	960	0	0.0	0	0	8.0	0.625625
11/2015	Quarter2	finishing	Wednesday	7	0.65	3.90	NaN	960	0	0.0	0	0	8.0	0.625625
11/2015	Quarter2	finishing	Wednesday	9	0.75	2.90	NaN	1800	0	0.0	0	0	15.0	0.505889
11/2015	Quarter2	finishing	Wednesday	6	0.70	2.90	NaN	720	0	0.0	0	0	6.0	0.394722

× 15 columns

```
[ ] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1197 entries, 0 to 1196
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   date                   1197 non-null   object
 1   quarter                1197 non-null   object
 2   department             1197 non-null   object
 3   day                    1197 non-null   object
 4   team                   1197 non-null   int64
 5   targeted_productivity  1197 non-null   float64
 6   smv                    1197 non-null   float64
 7   wip                    691 non-null    float64
 8   over_time              1197 non-null   int64
 9   incentive              1197 non-null   int64
 10  idle_time              1197 non-null   float64
 11  idle_men               1197 non-null   int64
 12  no_of_style_change     1197 non-null   int64
 13  no_of_workers          1197 non-null   float64
 14  actual_productivity    1197 non-null   float64
dtypes: float64(6), int64(5), object(4)
memory usage: 140.4+ KB
```

data types of the data

converting day column to binary

```
def convert_day_to_binary(day):
    if day in ['Saturday', 'Sunday']:
        return 1
    else:
        return 0
```

```
data['day_binary'] = data['day'].apply(convert_day_to_binary)
```

```
data.drop('day', axis=1, inplace=True)
```

```
data.drop('date', axis=1, inplace=True)
```

```
[74] data.isna().sum()
```

	0
quarter	0
department	0
team	0
targeted_productivity	0
smv	0
wip	506
over_time	0
incentive	0
idle_time	0
idle_men	0
no_of_style_change	0
no_of_workers	0
actual_productivity	0
day_binary	0

dtype: int64

checking for null values

removing the null values column

```
data.drop('wip', axis=1, inplace=True)
```

```
data.isna().sum()
```

	0
quarter	0
department	0
team	0
targeted_productivity	0
smv	0
over_time	0
incentive	0
idle_time	0
idle_men	0
no_of_style_change	0
no_of_workers	0
actual_productivity	0
day_binary	0

dtype: int64

```
[81] for i in range(data.shape[1]):
       print(data.iloc[:,i].unique())
       print(data.iloc[:,i].value_counts())
```

```
 0.71441049 0.70058841 0.61836111 0.60007051 0.54175    0.53690175
 0.50790323 1.10048392 1.09663333 0.83866667 0.75548611 0.75079944
 0.68801768 0.664875   0.65666667 0.63771186 0.60194444 0.58
 0.53567797 0.50012336 0.49788506 0.46340395 0.441392   0.92927778
 0.80088889 0.80037492 0.79620833 0.75039216 0.725625   0.70020613
 0.602      0.60044751 0.55725245 0.48333333 0.23804167 1.1204375
 1.108125   0.87644444 0.80080631 0.76083333 0.72256863 0.71533333
 0.70063277 0.7005731  0.68159804 0.65022372 0.60520833 0.60416667
 0.59862745 0.47571839 0.4321229  0.28704167 0.28305449 0.96043333
 0.80224332 0.80098039 0.80031237 0.72233333 0.70045989 0.62941667
 0.62197175 0.56597222 0.35542803 0.32996488 0.258      0.92754167
 0.91375    0.9025     0.7866     0.75062135 0.74916667 0.7008882
 0.70061403 0.70060345 0.65343137 0.650134   0.60098291 0.58604167
 0.5814     0.36107143 0.30211735 0.9918     0.93686111 0.919125
 0.8211125  0.80027969 0.75053268 0.73464583 0.70009573 0.671875
 0.64025    0.54979167 0.32813158 0.30357447 0.2565     0.25139925
 0.81640625 0.80071149 0.80047051 0.80009402 0.79998285 0.7858642
 0.73327778 0.710125   0.70054044 0.7        0.68402778 0.67213542
 0.63861438 0.63135417 0.61114054 0.60958333 0.58531579 0.24941667
 0.8721     0.8319375  0.8300625  0.80555556 0.80000295 0.78375
 0.753525   0.72734954 0.70060526 0.6721408  0.62701118 0.62657778
 0.456875   0.38579167 0.30750146 0.28395833 0.95579167 0.93041667
 0.87115    0.80013725 0.75077012 0.75029394 0.700623   0.70036207
 0.60128    0.41791667 0.3715625  0.36871875 0.35645833 0.81138889
 0.80007184 0.79145833 0.75072733 0.75043727 0.75017699 0.72693333
 0.70051852 0.7002568  0.5046875  0.47110849 0.325      0.26821429
 0.97081667 0.90296296 0.90083333 0.89955556 0.80080864 0.80011582
 0.75050357 0.7502069  0.70006981 0.70005833 0.65854167 0.59879234
 0.58113095 0.440375   0.41083333 0.9217037  0.92160494 0.80051667
 0.76884722 0.75047368 0.7503719  0.70025177 0.66237931 0.59061728
 0.5565625  0.49561751 0.44996491 0.4078125  0.37889515 0.37659722
 0.271875   0.80077902 0.80026082 0.75071698 0.75042593 0.75039551
```

finding unique values

dummy variable encoding

assaigning dummies

```
X= pd.get_dummies(data, drop_first=True)
X
```

:eam	targeted_productivity	smv	over_time	incentive	idle_time	idle_men	no_of_style_change	no_of_workers	actual_productivity	day_binary	quarter_Quarter2	quarter_Quarter3	qua
8	0.80	26.16	7080	98	0.0	0	0	59.0	0.940725	0	False	False	
1	0.75	3.94	960	0	0.0	0	0	8.0	0.886500	0	False	False	
11	0.80	11.41	3660	50	0.0	0	0	30.5	0.800570	0	False	False	
12	0.80	11.41	3660	50	0.0	0	0	30.5	0.800570	0	False	False	
6	0.80	25.90	1920	50	0.0	0	0	56.0	0.800382	0	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	
10	0.75	2.90	960	0	0.0	0	0	8.0	0.628333	0	True	False	
8	0.70	3.90	960	0	0.0	0	0	8.0	0.625625	0	True	False	
7	0.65	3.90	960	0	0.0	0	0	8.0	0.625625	0	True	False	
9	0.75	2.90	1800	0	0.0	0	0	15.0	0.505889	0	True	False	
6	0.70	2.90	720	0	0.0	0	0	6.0	0.394722	0	True	False	

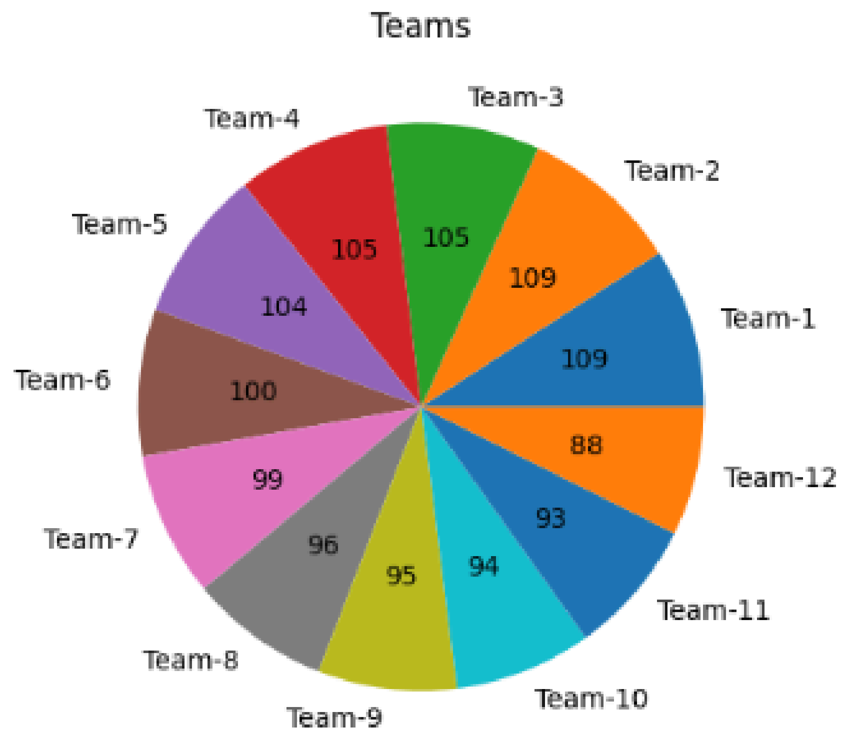
s × 17 columns

pie chart

```
[90] teams = data['team']
     sizes = teams.value_counts()

     # Custom function to display counts instead of percentages
     def autopct_format(pct, all_vals):
         absolute = int(round(pct/100.*sum(all_vals)))
         return f'{absolute}'

     plt.pie(sizes, labels=["Team-1", "Team-2", "Team-3", "Team-4", "Team-5", "Team-6", "Team-7", "Team-8", "Team-9", "Team-10", "Team-11", "Team-12"],
             autopct=lambda pct: autopct_format(pct, sizes))
     plt.title('Teams')
     plt.show()
```



```
attributes = ['over_time', 'incentive', 'idle_time', 'no_of_workers']

for column in attributes:
    plt.figure(figsize=(4,2))

    sns.lineplot(x=column, y='actual_productivity', data=data, label='Actual Productivity', marker='o')

    sns.lineplot(x=column, y='targeted_productivity', data=data, label='Targeted Productivity', marker='x')

    plt.title(f'Actual vs Targeted Productivity by {column}')
    plt.xlabel(column.capitalize().replace('_', ' '))
    plt.ylabel('Productivity')
    plt.legend()

    plt.show()
```

Line plot

Correlation heatmap

```
corr_matrix = data.corr(numeric_only=True)
```

```
[86] plt.figure(figsize=(10, 8))
     sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
     plt.title("Correlation Heatmap of Numerical Features")
     plt.show()
```

Analysis of Standard Minute value(SMV) based on each attribute

```
[84] variables = ['actual_productivity', 'targeted_productivity', 'no_of_workers',
                  'incentive', 'over_time', 'idle_time']

     plt.figure(figsize=(14, 10))

     for i, var in enumerate(variables, 1):
         plt.subplot(2, 3, i)
         sns.scatterplot(x='smv', y=var, data=data, marker='o', color='blue')
         plt.title(f'SMV vs {var}')
         plt.xlabel('SMV (Standard Minute Value)')
         plt.ylabel(var)


     plt.tight_layout()
     plt.show()
```

Scatter plot

Multicollinearity Checking

```
[ ] from statsmodels.stats.outliers_influence import variance_inflation_factor
```

```
[ ] def calc_vif(X):
        vif = pd.DataFrame()
        vif["variables"] = X.columns
        vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

        return(vif)

    calc_vif(X)
```

```
[ ] calc_vif(X.drop('no_of_workers', axis=1))
```

```
[ ] calc_vif(X.drop(['no_of_workers','smv'], axis=1))
```

```
[ ] calc_vif(X.drop(['no_of_workers','smv','targeted_productivity'], axis=1))
```

```
[ ] calc_vif(X.drop(['no_of_workers','smv','targeted_productivity','department_sweing'], axis=1))
```

```
[ ] import statsmodels.api as sm
```

```
[ ] X_train_nomulti = X.drop(['no_of_workers','smv','targeted_productivity','department_sweing'], axis=1)
```

Multiple regression

Loading...

```
lm2 = LinearRegression()
lm2.fit(X_train2, y_train2)
y_pred = lm2.predict(X_test)
print(np.sqrt(metrics.mean_squared_error(y_test2, y_pred)))
```

```
[ ] r2_score(y_test2,y_pred2)
```

```
[ ] metrics.mean_absolute_error(y_test2,y_pred2)
```

```
[ ] lm2 = LinearRegression()
    lm2.fit(X_train4, y_train4)
    y_pred = lm2.predict(X_test4)
    print(np.sqrt(metrics.mean_squared_error(y_test4, y_pred)))
```

```
[ ] r2_score(y_test4,y_pred4)
```

```
[ ] metrics.mean_absolute_error(y_test4,y_pred4)
```

```
[ ] lm2 = LinearRegression()
    lm2.fit(X_train2, y_train2)
    y_pred = lm2.predict(X_test)
    print(np.sqrt(metrics.mean_squared_error(y_test2, y_pred)))
```

```
[ ] r2_score(y_test2,y_pred2)
```

```
[ ] metrics.mean_absolute_error(y_test2,y_pred2)
```

Loading...

```
X_train1,X_test1,y_train1,y_test1=train_test_split(X,y,test_size=0.20,train_size=0.80)
X_train2,X_test2,y_train2,y_test2=train_test_split(X,y,test_size=0.25,train_size=0.75)
X_train3,X_test3,y_train3,y_test3=train_test_split(X,y,test_size=0.30,train_size=0.70)
X_train4,X_test4,y_train4,y_test4=train_test_split(X,y,test_size=0.40,train_size=0.60)
```

80 20 SPLIT

```
[ ] from sklearn.neighbors import KNeighborsRegressor
    from sklearn.metrics import r2_score
    from sklearn import metrics
    from sklearn.metrics import mean_squared_error
```

```
[ ] model=KNeighborsRegressor(n_neighbors=20)
```

80 - 20 SPLIT

```
[ ] model.fit(X_train1,y_train1)
```

```
[ ] y_pred1=model.predict(X_test1)
    y_pred1
```

KNN alogorithm

SVM algorithm

```
[ ] from sklearn.svm import SVR
    from sklearn import metrics
    from sklearn.metrics import mean_squared_error
    from sklearn.metrics import mean_absolute_error
    from sklearn.metrics import mean_absolute_percentage_error
    from sklearn.metrics import r2_score
```

```
[ ] model = SVR(kernel='rbf')
```

80 - 20 SPLIT

```
[ ] model.fit(X_train1, y_train1)
```

```
[ ] y_pred1= model.predict(X_test1)
    y_pred1
```

```
[ ] svm = pd.DataFrame({'Predicted':y_pred1,'Actual':y_test1})
    svm
```

```
[ ] from sklearn.tree import DecisionTreeRegressor
    from sklearn.tree import plot_tree
```

```
[ ] clf = DecisionTreeRegressor()
```

80 - 20 SPLIT

```
[ ] clf = clf.fit(X_train1,y_train1)
```

```
[ ] y_pred1 = clf.predict(X_test1)
    y_pred1
```

```
[ ] print("R-squared:", metrics.r2_score(y_test1, y_pred1))
    print("Mean Absolute Error:", metrics.mean_absolute_error(y_test1, y_pred1))
    print("Mean Squared Error:", metrics.mean_squared_error(y_test1, y_pred1))
```

```
[ ] clf = DecisionTreeRegressor(criterion="absolute_error", max_depth=3)

    clf = clf.fit(X_train1,y_train1)
    y_pred1 = clf.predict(X_test1)
    print("R-squared:", metrics.r2_score(y_test1, y_pred1))
    print("Mean Absolute Error:", metrics.mean_absolute_error(y_test1, y_pred1))
    print("Mean Squared Error:", metrics.mean_squared_error(y_test1, y_pred1))
```

Decision Tree

Random Forest

```
[ ] from sklearn.ensemble import RandomForestRegressor
```

```
[ ] rf = RandomForestRegressor()
```

80 - 20 SPLIT

```
[ ] rf.fit(X_train1, y_train1)
```

```
[ ] y_pred = rf.predict(X_test1)
    y_pred
```

```
[ ] mse = mean_squared_error(y_test1, y_pred)
    mse
```

```
[ ] r2 = r2_score(y_test1, y_pred)
    r2
```

```
[ ] mae = mean_absolute_error(y_test1, y_pred)
    mae
```

75 - 25 SPLIT

```
[ ] import xgboost as xgb
    from sklearn.ensemble import RandomForestRegressor
```

```
[ ] model1 = xgb.XGBRegressor()
    model2 = xgb.XGBRegressor(n_estimators=100, max_depth=8, learning_rate=0.1, subsample=0.5)
```

80 - 20 SPLIT

```
[ ] train_model1 = model1.fit(X_train1, y_train1)
    train_model2 = model2.fit(X_train1, y_train1)
```

```
[ ] pred1 = train_model1.predict(X_test1)
    pred2 = train_model2.predict(X_test1)
```

```
[ ] # Evaluate the models using regression metrics
    mse1 = mean_squared_error(y_test1, pred1)
    r2_1 = r2_score(y_test1, pred1)
    mae1 = mean_absolute_error(y_test1, pred1)
```

```
[ ] mse2 = mean_squared_error(y_test1, pred2)
    r2_2 = r2_score(y_test1, pred2)
    mae2 = mean_absolute_error(y_test1, pred2)
```

XGBoost

ADABoost

```
[ ] from sklearn.ensemble import AdaBoostRegressor
```

```
[ ] estimator = DecisionTreeRegressor(max_depth=3, random_state=0)
    adaboost = AdaBoostRegressor(estimator=estimator, n_estimators=3, random_state=0)
```

80 - 20 SPLIT

```
[ ] adaboost.fit(X_train1, y_train1)
```

```
[ ] y_pred1 = adaboost.predict(X_test1)
    y_pred1
```

```
[ ] mse = mean_squared_error(y_test1, y_pred1)
    print("MSE:", mse)
    r2 = r2_score(y_test1, y_pred1)
    print("R-squared:", r2)
    mae = mean_absolute_error(y_test1, y_pred1)
    print("MAE:", mae)
```

75 - 25 SPLIT

```
[ ] adaboost.fit(X_train2, y_train2)
```

NEURAL NETWORKS

75-25 TRAIN_TEST_SPLIT

BEFORE MULTICOLLINEARITY CHECKING

Epochs=300

```python
# STEP 1: Data Preprocessing - Normalize the input data
scaler = StandardScaler()
X_train2_nomulti_scaled = scaler.fit_transform(X_train2_nomulti)  # Assuming X_train2 is a NumPy array

# STEP 2: Updated Model Architecture
model = tf.keras.Sequential([
    tf.keras.layers.Dense(40, activation="relu", input_shape=(X_train2_nomulti_scaled.shape[1],)),
    tf.keras.layers.Dense(35, activation="relu"),
    tf.keras.layers.Dense(22, activation="relu"),
    tf.keras.layers.Dense(16, activation="tanh"),
    tf.keras.layers.Dense(8, activation="tanh"),
    tf.keras.layers.Dense(4, activation="tanh"),
    tf.keras.layers.Dense(1, activation="linear")  # No activation for regression
])

# STEP 3: Compile the model with a lower learning rate and optimizer tuning
model.compile(loss=tf.keras.losses.MeanAbsoluteError(),
              optimizer=tf.keras.optimizers.Adam(learning_rate=0.01),  # Decreased learning rate
              metrics=["mae"])

# STEP 4: Callbacks for Early Stopping and ReduceLROnPlateau
early_stopping = EarlyStopping(monitor="val_loss", patience=50, restore_best_weights=True, verbose=1)
reduce_lr = ReduceLROnPlateau(monitor="val_loss", factor=0.5, patience=30, verbose=1)

# STEP 5: Fit the model with callbacks
history = model.fit(X_train2_nomulti_scaled, y_train2, epochs=300, batch_size=68, verbose=1,
                    validation_split=0.2, callbacks=[early_stopping, reduce_lr])
```

```
Epoch 1/300
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
11/11 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - loss: 0.3070 - mae: 0.3070 - val_loss: 0.1741 - val_mae: 0.1741 - learning_rate: 0.0100
Epoch 2/300
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.1528 - mae: 0.1528 - val_loss: 0.1419 - val_mae: 0.1419 - learning_rate: 0.0100
Epoch 3/300
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.1347 - mae: 0.1347 - val_loss: 0.1384 - val_mae: 0.1384 - learning_rate: 0.0100
Epoch 4/300
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.1296 - mae: 0.1296 - val_loss: 0.1326 - val_mae: 0.1326 - learning_rate: 0.0100
Epoch 5/300
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.1242 - mae: 0.1242 - val_loss: 0.1225 - val_mae: 0.1225 - learning_rate: 0.0100
Epoch 6/300
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.1154 - mae: 0.1154 - val_loss: 0.1164 - val_mae: 0.1164 - learning_rate: 0.0100
Epoch 7/300
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.1116 - mae: 0.1116 - val_loss: 0.1133 - val_mae: 0.1133 - learning_rate: 0.0100
Epoch 8/300
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.1073 - mae: 0.1073 - val_loss: 0.1129 - val_mae: 0.1129 - learning_rate: 0.0100
Epoch 9/300
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.1021 - mae: 0.1021 - val_loss: 0.1091 - val_mae: 0.1091 - learning_rate: 0.0100
Epoch 10/300
```