GROUP-4

MACHINE LEARNING PROJECT TO PREDICT THE ACTUAL PRODUCTIVITY OF GARMENT WORKERS

J.SaiDwarak -107222546016
B.Poojashree -107222546007
G.Shyam -107222546014
S.Jayavardhan -107222546017

ABSTRACT

This study applies regression analysis to investigate the factors influencing productivity among garment workers, using a comprehensive dataset. By utilizing several Machine Learning Algorithms such as Linear and Multiple regression, KNN, SVM and Decision Tree, Random Forest ,bagging and Boosting the study aims to establish relationships between different variables.

Objective

Determine the relationships between various independent variables (e.g., department, idle time, team, targeted productivity)

Develop a predictive model that can be used by garment manufacturers to improve operational efficiency, and enhance overall production output.

Develop a predictive model that can be used by garment manufacturers to optimize workforce management, improve operational efficiency, and enhance overall production output.

CONTENTS

1. Introduction
2. literature review
3. data Pre-processing
4. Exploratory data Analysis
5. Data Modelling and Evaluation
6. Summary
7. Appendix

INTRODUCTION

Garment workers are individuals engaged in the production of clothing and textiles, forming the backbone of the global fashion and apparel industry. They are employed in a variety of roles, ranging from sewing, cutting, and assembling fabrics to finishing, quality control, and packaging. Garment workers are found in factories, workshops, and home-based setups across the world, with a significant concentration in developing countries such as Bangladesh, India, Vietnam, China, and Cambodia.

LITERATURE REVIEW-1

AUTHORS: MD EMRAN BISWAS et.al

TITLE: MACHINE LEARNING APPROACH TO ESTIMATE REQUIREMENTS FOR TARGET PRODUCTIVITY OF GARMENTS EMPLOYEES

SUMMARY: This study explores data-driven approaches to enhance productivity in the garment industry by predicting employee-level productivity and estimating team operating factors such as incentives and overtime. using three months of industry data, including department id, standard minute value, incentive, overtime, worker count, and actual productivity, the researchers combined machine learning techniques with optimization algorithms to recommend strategies for achieving target productivity.

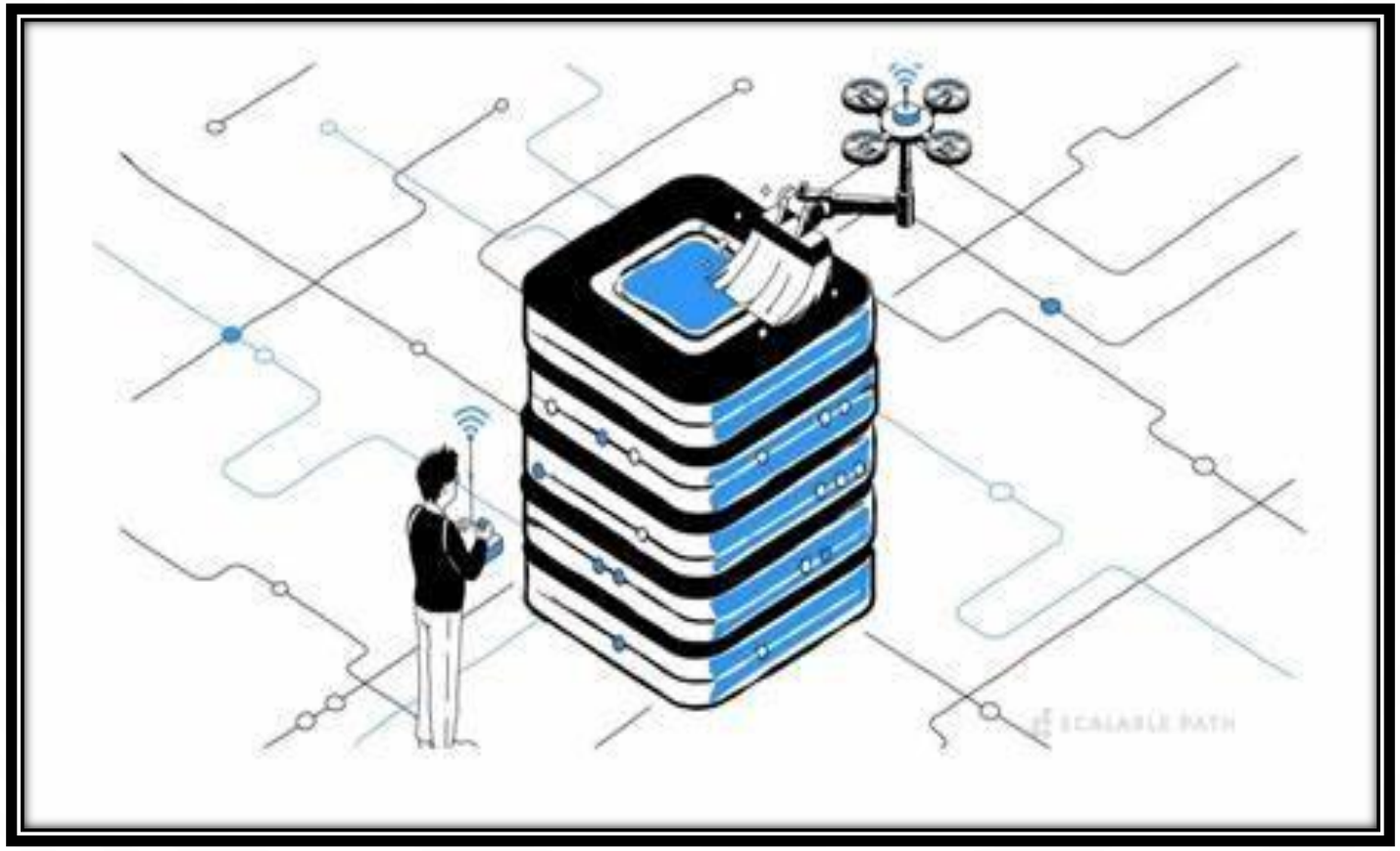
LITERATURE REVIEW - 2

AUTHORS: RUBA OBIEDAT et.al

TITLE: A COMBINED APPROACH FOR PREDICTING EMPLOYEES' PRODUCTIVITY BASED ON ENSEMBLE MACHINE LEARNING METHODS

SUMMARY: This research focuses on predicting garment employee productivity using a hybrid approach that combines classification algorithms with ensemble learning techniques. The study evaluates algorithms including Random forest with 0.983 Accuracy and MAE as 0.0097 , Adaboost with 0.991 Accuracy and MAE as 0.0101.

DATA PRE-PROCESSING



DATA

- **Dataset –** Our dataset consists of 15 variables and 1198 records
- **Source –** https://archive.ics.uci.edu/dataset/597/productivity+prediction+of+garment+employees
- **Categorical variables –** Quarter, Department, Day
- **Continuous variables –** Date ,Team , targerted productivity, smv , wip, overtime, incentive , idle time, idle men, no of style, no of workers, actual productivity.

date	quarter	departme	day	team	targeted_	smv	wip	over_time	incentive	idle_time	idle_men	no_of_sty	no_of_wc	actual_productivity
########	Quarter1	sweing	Thursday	8	0.8	26.16	1108	7080	98	0	0	0	59	0.940725
########	Quarter1	finishing	Thursday	1	0.75	3.94		960	0	0	0	0	8	0.8865
########	Quarter1	sweing	Thursday	11	0.8	11.41	968	3660	50	0	0	0	30.5	0.80057
########	Quarter1	sweing	Thursday	12	0.8	11.41	968	3660	50	0	0	0	30.5	0.80057
########	Quarter1	sweing	Thursday	6	0.8	25.9	1170	1920	50	0	0	0	56	0.800382
########	Quarter1	sweing	Thursday	7	0.8	25.9	984	6720	38	0	0	0	56	0.800125
########	Quarter1	finishing	Thursday	2	0.75	3.94		960	0	0	0	0	8	0.755167
########	Quarter1	sweing	Thursday	3	0.75	28.08	795	6900	45	0	0	0	57.5	0.753683
########	Quarter1	sweing	Thursday	2	0.75	19.87	733	6000	34	0	0	0	55	0.753098
########	Quarter1	sweing	Thursday	1	0.75	28.08	681	6900	45	0	0	0	57.5	0.750428
########	Quarter1	sweing	Thursday	9	0.7	28.08	872	6900	44	0	0	0	57.5	0.721127
########	Quarter1	sweing	Thursday	10	0.75	19.31	578	6480	45	0	0	0	54	0.712205
########	Quarter1	sweing	Thursday	5	0.8	11.41	668	3660	50	0	0	0	30.5	0.707046
########	Quarter1	finishing	Thursday	10	0.65	3.94		960	0	0	0	0	8	0.705917
########	Quarter1	finishing	Thursday	8	0.75	2.9		960	0	0	0	0	8	0.676667
########	Quarter1	finishing	Thursday	4	0.75	3.94		2160	0	0	0	0	18	0.593056
########	Quarter1	finishing	Thursday	7	0.8	2.9		960	0	0	0	0	8	0.540729
########	Quarter1	sweing	Thursday	4	0.65	23.69	861	7200	0	0	0	0	60	0.52118
########	Quarter1	finishing	Thursday	11	0.7	4.15		1440	0	0	0	0	12	0.426326

DATA CLEANING

- Checking the Null values of every column
- Dropping the wip and date columns as the wip column consists

  many null values and date column is not necessary for

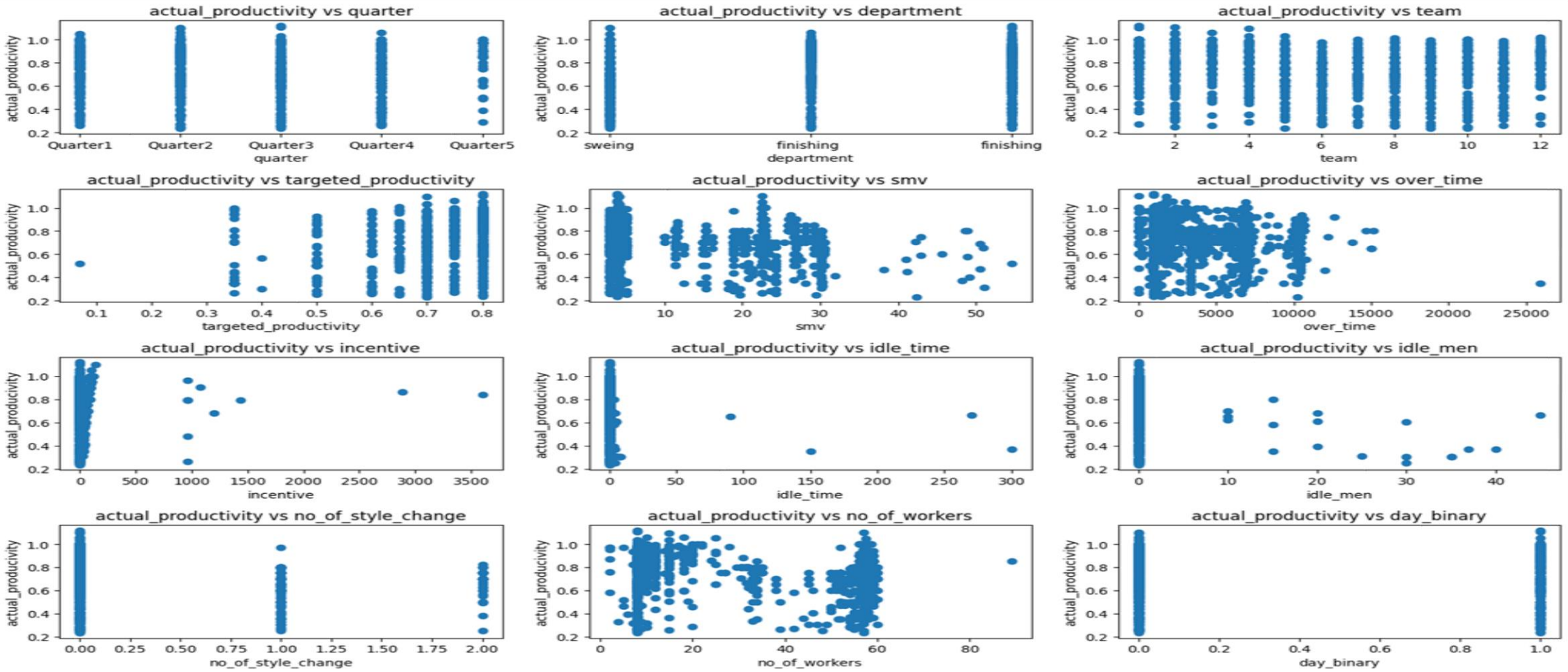
  our analysis
- Identifying the unique values of every column for further

  analysis
- Converting the day column into binary categorical column

EXPLORATORY DATA ANALYSIS

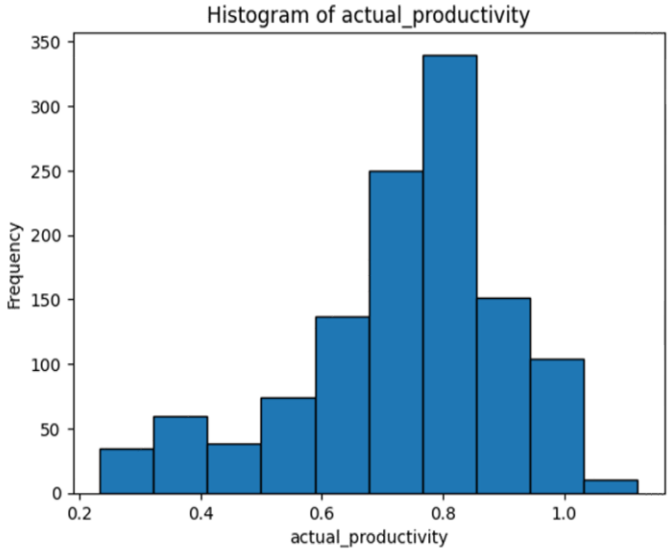
SCATTER-PLOT

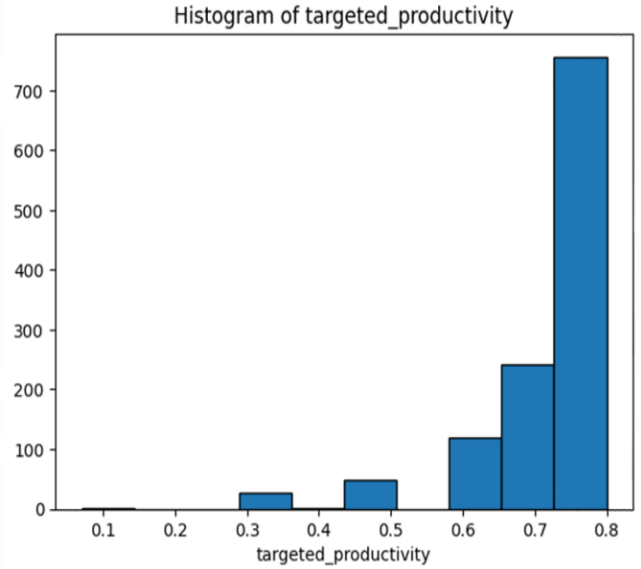
- The target variable ‘actual productivity’ is plotted against the independent variables on the scatterplot shown below

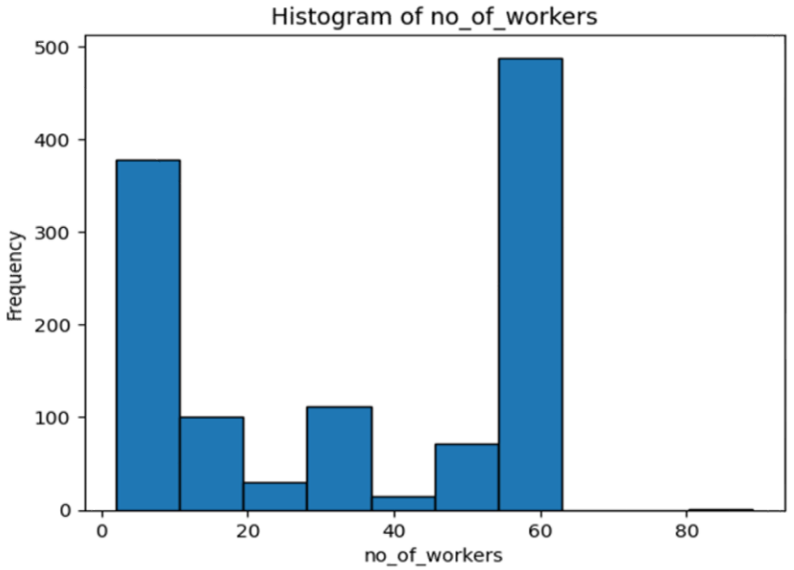


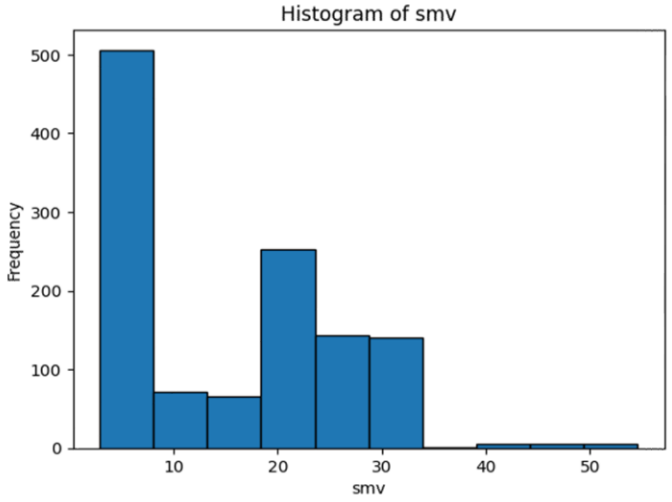
HISTOGRAM

- The histogram of the continuous variables is plotted as shown below



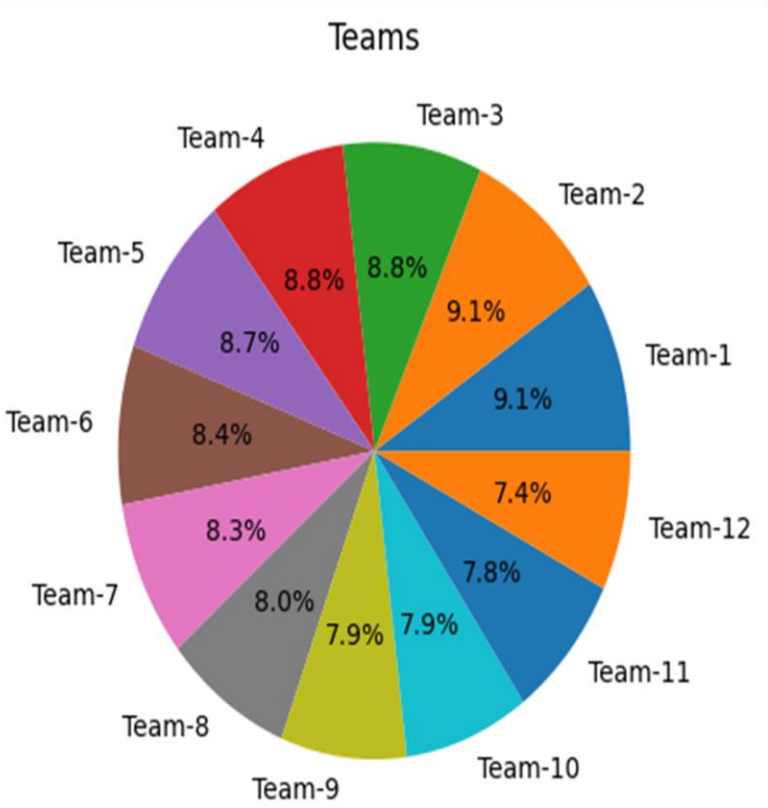


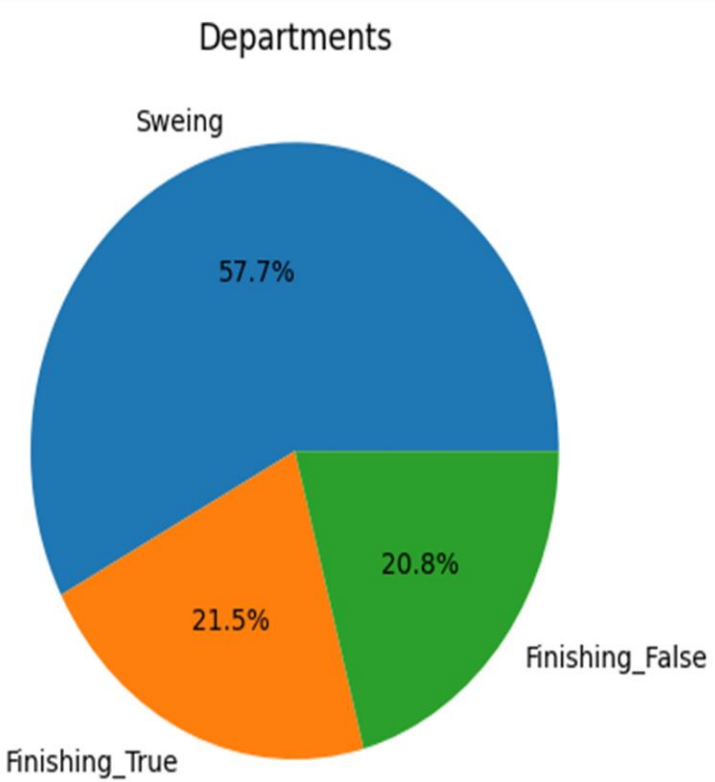


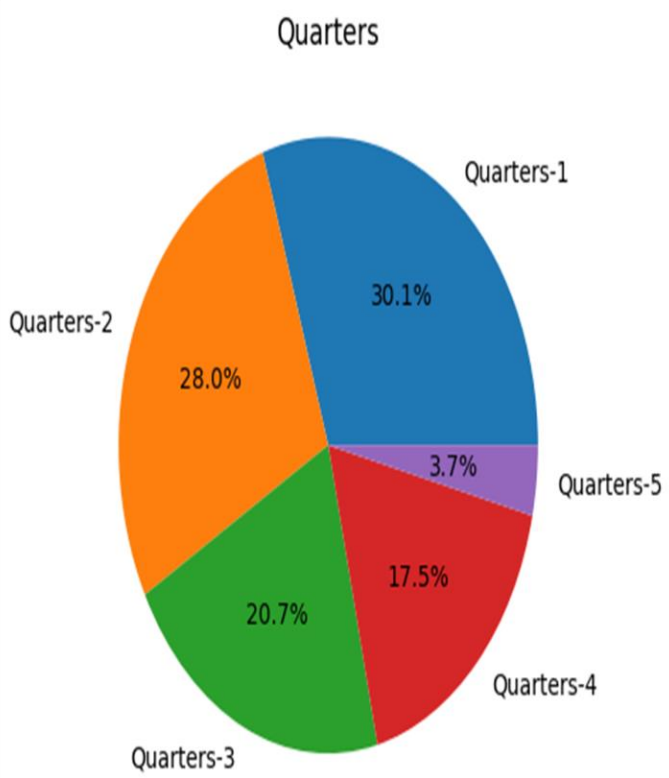


PIE CHARTS

The pie charts of the categorical columns are distributed as shown below

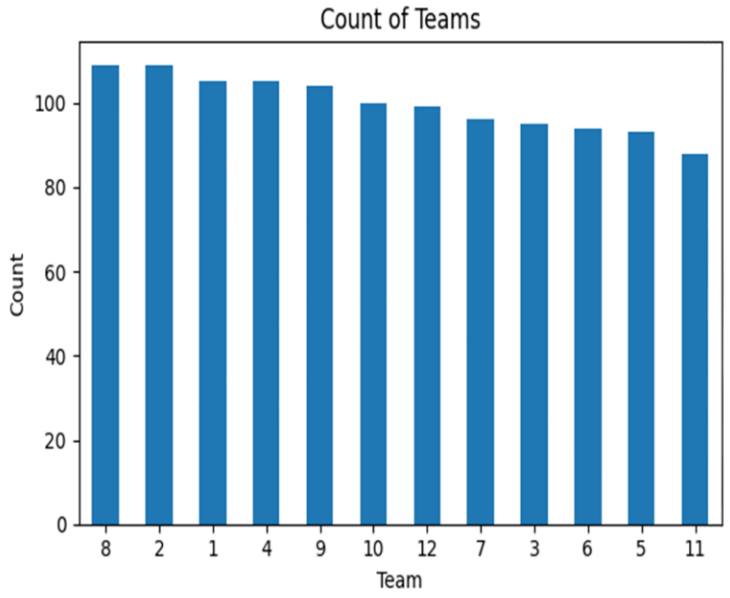


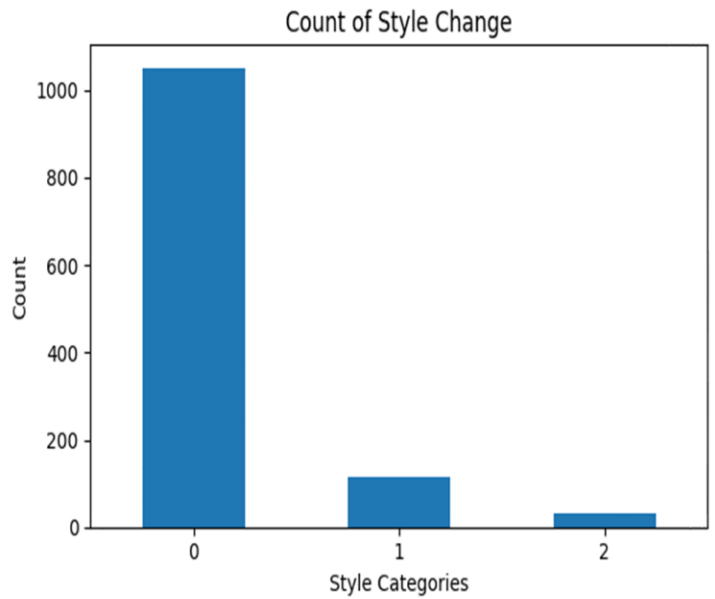


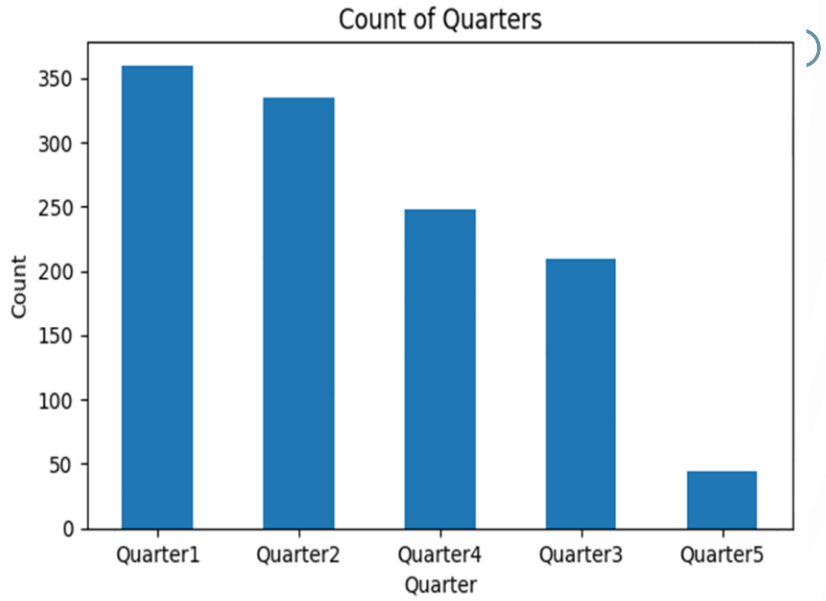


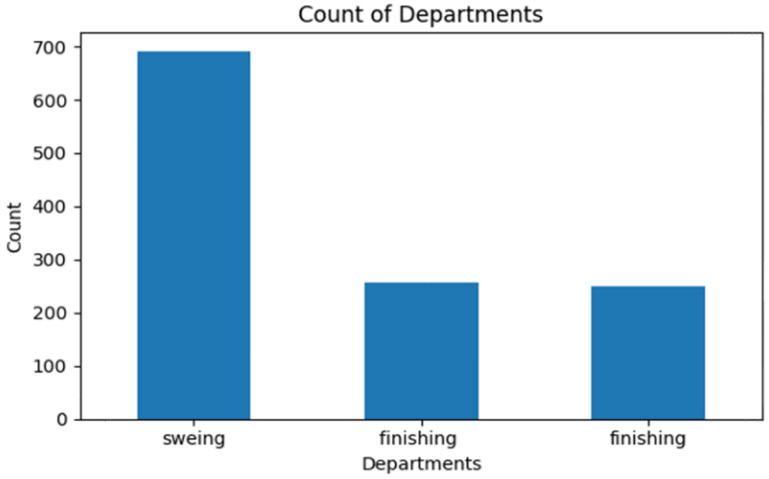
BAR PLOTS

- Bar plot is generally used to plot categorical data. The bar plot for the categorical variables is plotted as shown below





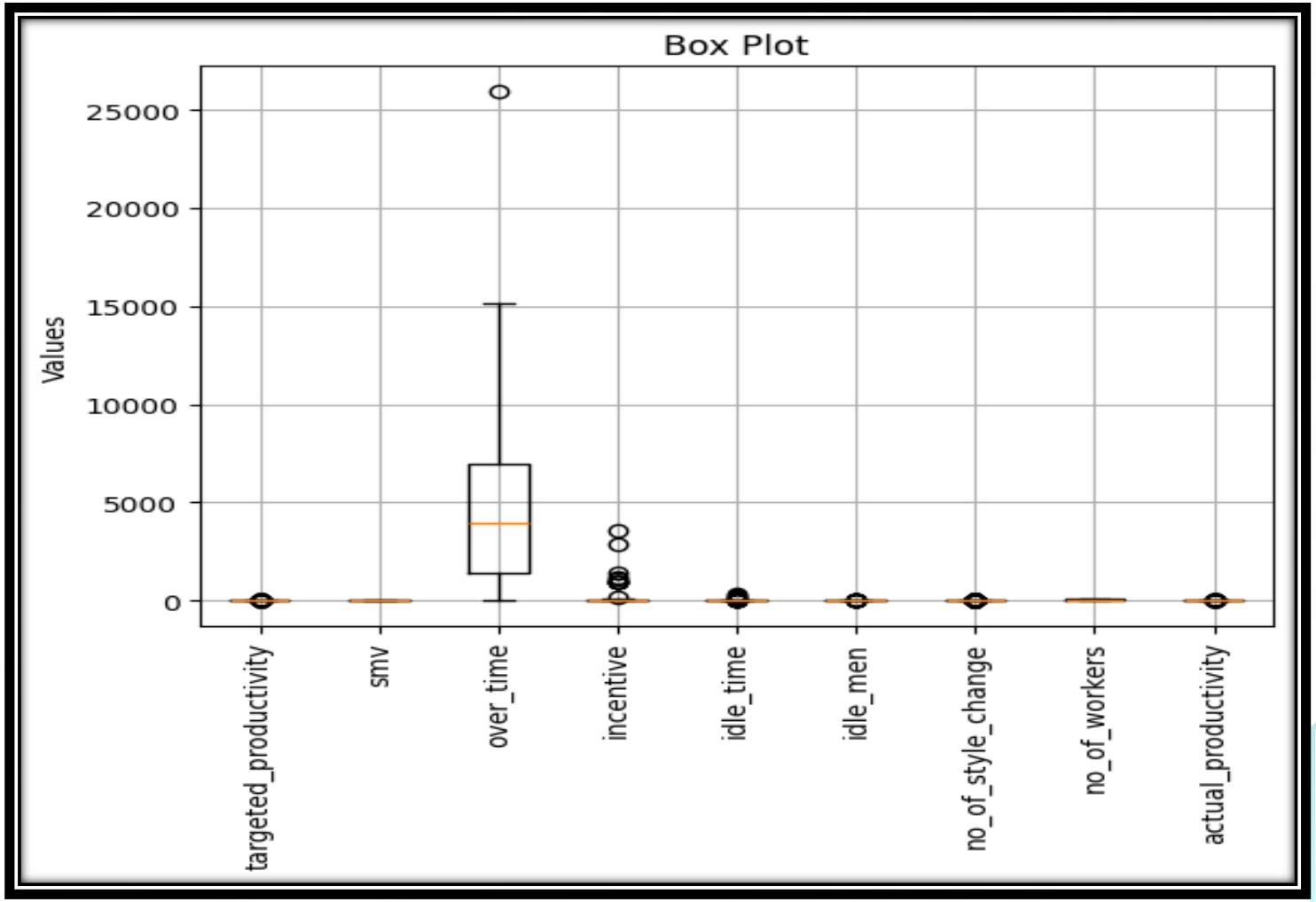




BOX PLOT

Box Plot:
This plot is used to detect outliers in the data.

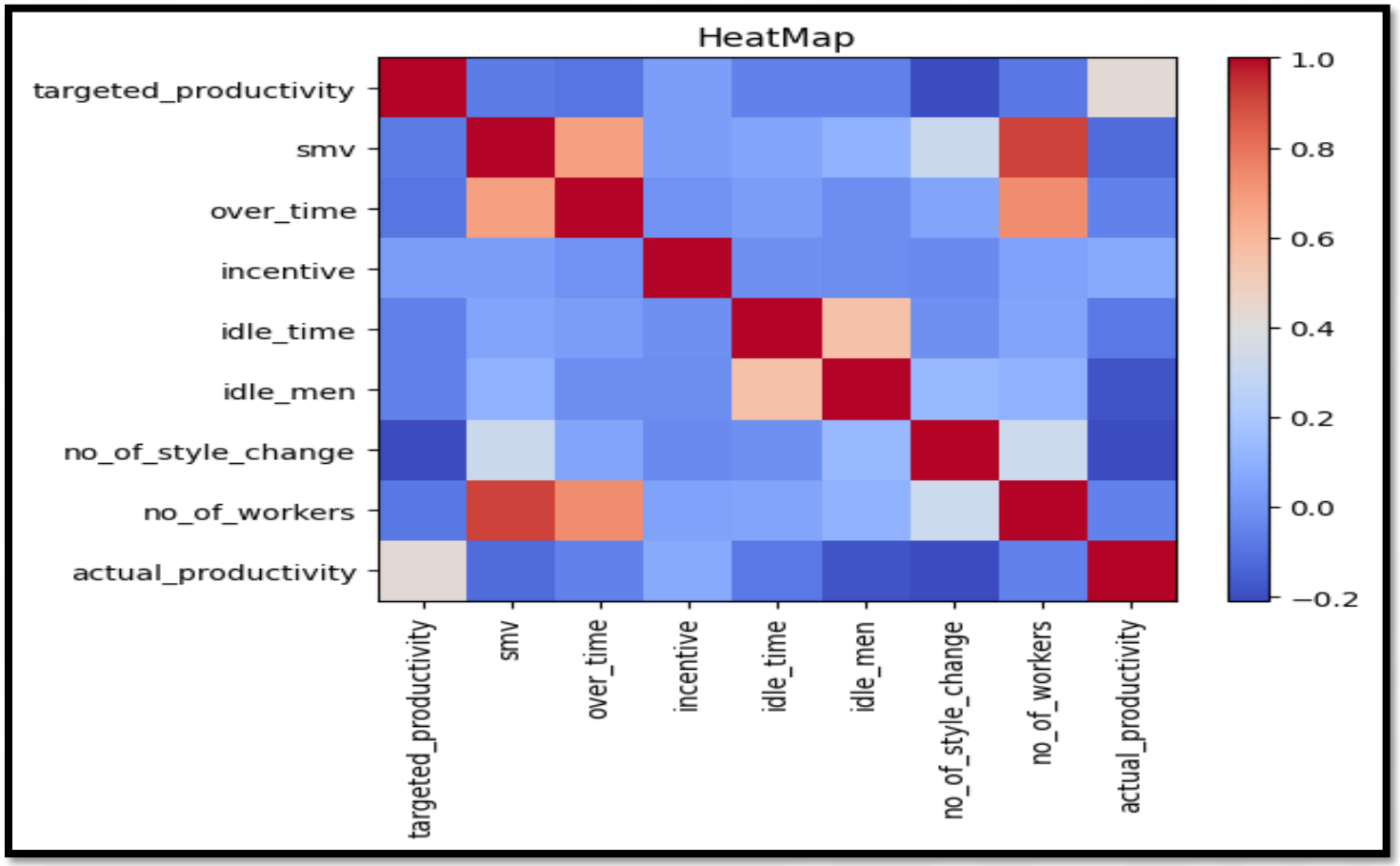
In this case the column overtime has one outlier.



HEAT MAP

Heat Maps: Heat Maps are used to check for high multi-collinearity in the data

Here we can observe that smv and no_of_workers are having high positive correlation.



MULTICOLLINEARITY CHECK

VIF<3

	variables	VIF
0	team	1.1
1	targeted_productivity	1.1
2	smv	6.2
3	over_time	2.5
4	incentive	1.0
5	idle_time	1.5
6	idle_men	1.5
7	no_of_style_change	1.4
8	no_of_workers	14.5
9	day_binary	1.0
10	quarter_Quarter1	22.5
11	quarter_Quarter2	20.6
12	quarter_Quarter3	13.3
13	quarter_Quarter4	15.4
14	quarter_Quarter5	3.5
15	department_sweing	9.6

Quarter_quarter1 has been removed

	variables	VIF
0	team	4.7
1	targeted_productivity	10.6
2	smv	18.6
3	over_time	7.2
4	incentive	1.1
5	idle_time	1.3
6	idle_men	1.4
7	no_of_style_change	1.6
8	no_of_workers	49.9
9	day_binary	1.5
10	quarter_Quarter2	2.0
11	quarter_Quarter3	1.6
12	quarter_Quarter4	1.8
13	quarter_Quarter5	1.1
14	department_sweing	22.7

No_of_workers has been removed

	variables	VIF
0	team	4.6
1	targeted_productivity	9.0
2	smv	14.7
3	over_time	6.3
4	incentive	1.1
5	idle_time	1.3
6	idle_men	1.4
7	no_of_style_change	1.5
8	day_binary	1.5
9	quarter_Quarter2	2.0
10	quarter_Quarter3	1.6
11	quarter_Quarter4	1.8
12	quarter_Quarter5	1.1
13	department_sweing	12.1

Smv has been removed

	variables	VIF
0	team	4.3
1	targeted_productivity	8.2
2	over_time	5.8
3	incentive	1.1
4	idle_time	1.3
5	idle_men	1.4
6	no_of_style_change	1.5
7	day_binary	1.5
8	quarter_Quarter2	1.9
9	quarter_Quarter3	1.6
10	quarter_Quarter4	1.8
11	quarter_Quarter5	1.1
12	department_sweing	5.5

Department sewing has been removed

	variables	VIF
0	team	4.2
1	targeted_productivity	8.1
2	over_time	2.8
3	incentive	1.1
4	idle_time	1.3
5	idle_men	1.3
6	no_of_style_change	1.2
7	day_binary	1.5
8	quarter_Quarter2	1.9
9	quarter_Quarter3	1.6
10	quarter_Quarter4	1.8
11	quarter_Quarter5	1.1

Targeted productivity has been removed

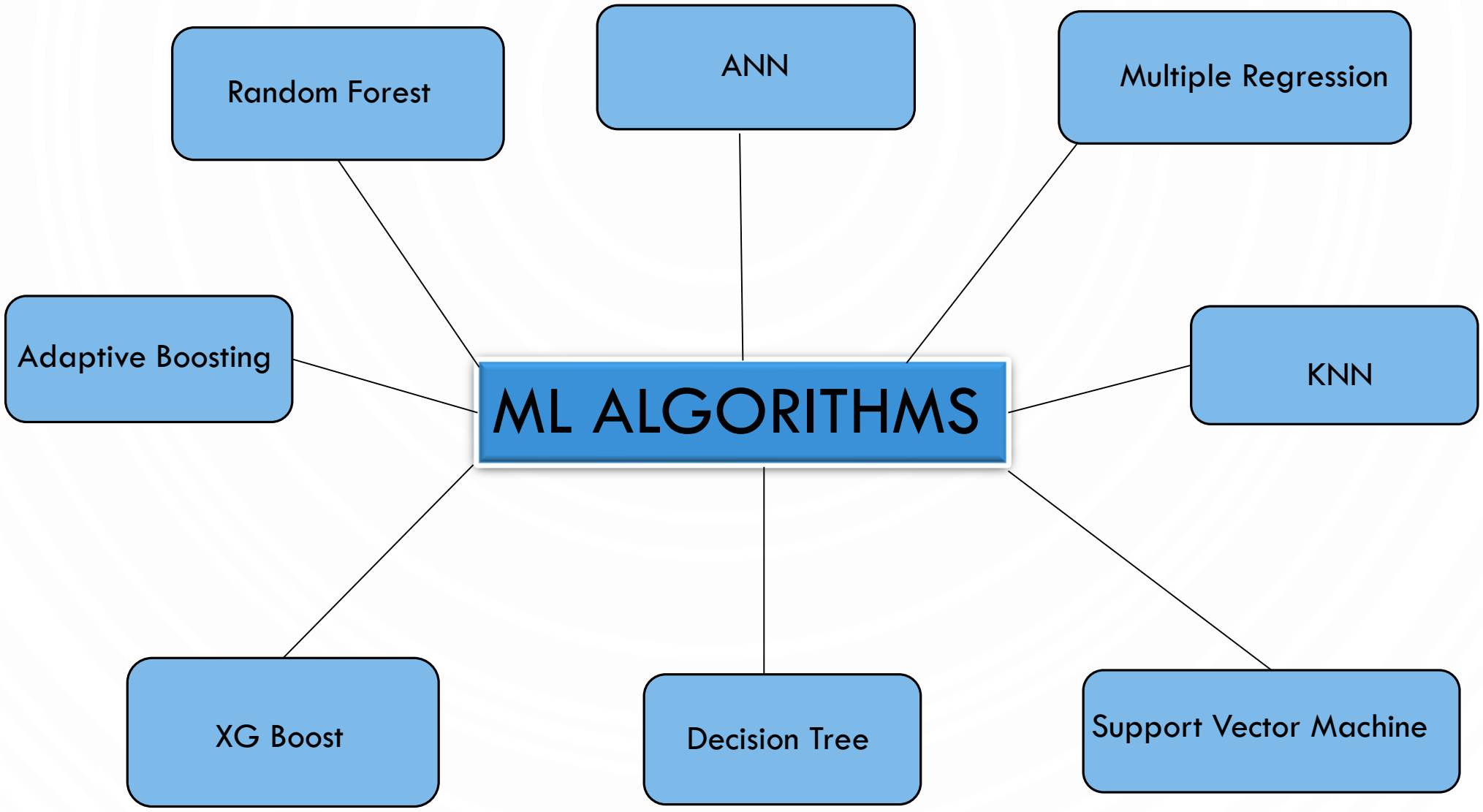
	variables	VIF
0	team	2.6
1	over_time	2.2
2	incentive	1.1
3	idle_time	1.3
4	idle_men	1.3
5	no_of_style_change	1.2
6	day_binary	1.4
7	quarter_Quarter2	1.7
8	quarter_Quarter3	1.5
9	quarter_Quarter4	1.6
10	quarter_Quarter5	1.1

Team has been removed

	variables	VIF
0	over_time	2.0
1	incentive	1.1
2	idle_time	1.3
3	idle_men	1.3
4	no_of_style_change	1.2
5	day_binary	1.4
6	quarter_Quarter2	1.4
7	quarter_Quarter3	1.3
8	quarter_Quarter4	1.4
9	quarter_Quarter5	1.0

	variables	VIF
0	incentive	1.1
1	idle_time	1.3
2	idle_men	1.3
3	no_of_style_change	1.2
4	day_binary	1.3
5	quarter_Quarter2	1.2
6	quarter_Quarter3	1.1
7	quarter_Quarter4	1.2
8	quarter_Quarter5	1.0

Over time has been removed


ML ALGORITHMS
Random Forest
ANN
Multiple Regression
Adaptive Boosting
KNN
XG Boost
Decision Tree
Support Vector Machine

MULTIPLE REGRESSION

SPLIT	r^2_score (Before vif)	MAE (Before vif)	r^2_score (After vif)	MAE (After vif)
80-20	0.165	0.119	0.010	0.132
72-25	0.204	0.115	0.011	0.128
70-30	0.199	0.122	0.010	0.126
60-40	0.225	0.114	-0.017	0.129

K NEAREST NEIGHBOUR

SPLIT	r^2_score (Before vif)	MAE (Before vif)	r^2_score (After vif)	MAE (After vif)
80-20	0.221	0.109	-0.200	0.126
75-25	0.409	0.085	-0.044	0.113
70-30	0.403	0.846	0.030	0.108
60-40	0.008	0.135	0.105	0.109

SUPPORT VECTOR MACHINE

SPLIT	r^2_score (Before vif)	MAE (Before vif)	r^2_score (After vif)	MAE (After vif)
80-20	0.049	0.135	0.119	0.118
75-25	0.040	0.136	0.131	0.113
70-30	0.261	0.106	0.117	0.112
60-40	0.008	0.135	0.125	0.114

DECISION TREE

SPLIT	r^2_score (Before vif)	MAE (Before vif)	r^2_score (After vif)	MAE (After vif)
80-20	0.315	0.099	0.111	0.113
75-25	0.270	0.103	0.123	0.108
70-30	0.261	0.100	0.136	0.104
60-40	0.262	0.101	0.136	0.105

XG BOOST

SPLIT	r^2_score (Before vif)	MAE (Before vif)	r^2_score (After vif)	MAE (After vif)
80-20	0.132	0.109	0.027	0.117
75-25	0.160	0.110	0.025	0.112
70-30	0.201	0.105	0.090	0.108
60-40	0.430	0.082	0.095	0.111

ADAPTIVE BOOSTING

SPLIT	r^2_score (Before vif)	MAE (Before vif)	r^2_score (After vif)	MAE (After vif)
80-20	0.315	0.099	0.119	0.118
75-25	0.409	0.085	0.157	0.109
70-30	0.403	0.085	0.162	0.108
60-40	0.008	0.135	0.082	0.114

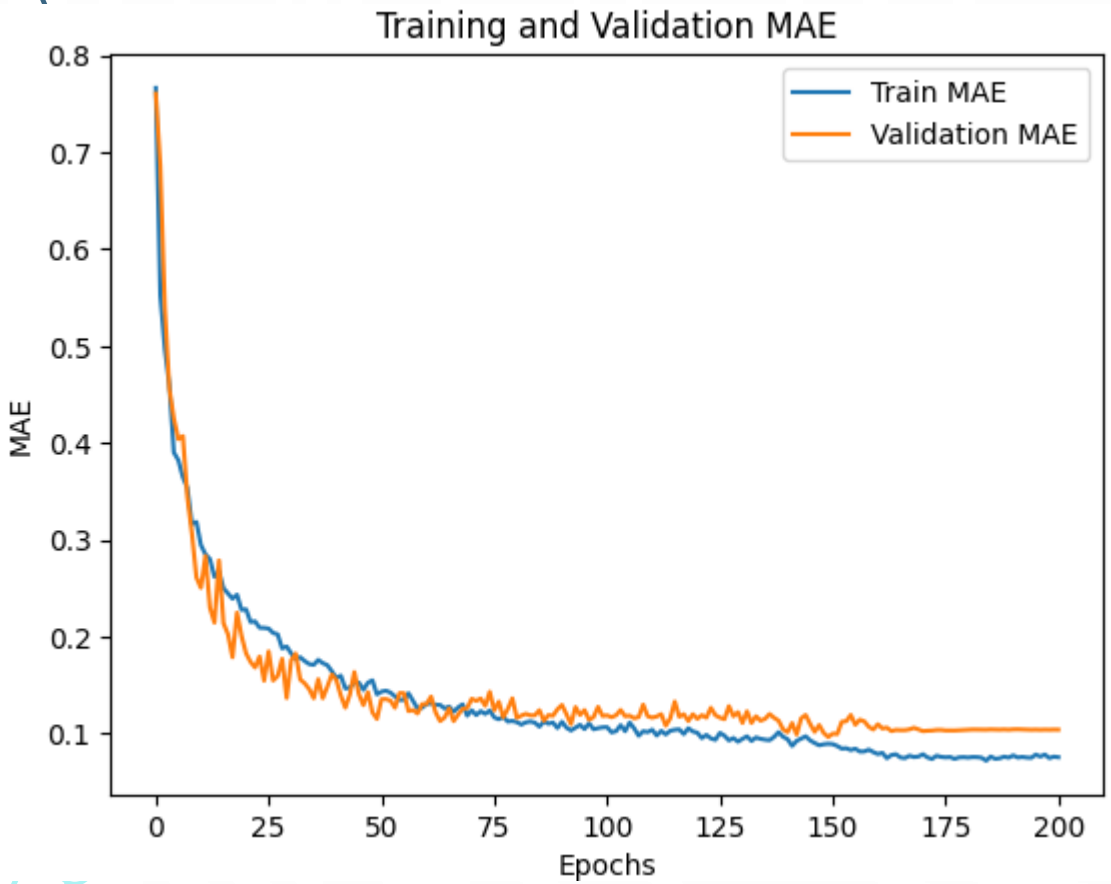
RANDOM FOREST

SPLIT	r^2_score (Before vif)	MAE (Before vif)	r^2_score (After vif)	MAE (After vif)
80-20	0.480	0.081	0.075	0.112
75-25	0.430	0.083	0.091	0.107
70-30	0.416	0.081	0.099	0.105
60-40	0.427	0.083	0.134	0.105

ARTIFICIAL NEURAL NETWORK

SPLIT	Architecture	Optimizer	Epochs	MAE
80-20	32 – 16 – 8 – 4 – 1	SDG	50	nan
80-20	32 – 16 – 8 – 4 – 1	Adam	50	5.741
80-20	64 – 32 – 16 – 8 – 4 – 1	Adam	100	4.666
80-20	128 – 64 – 32 – 16 – 8 – 4 –1	Adam	300	1.512
75-25	32 – 16 – 8 – 4 – 1	SDG	50	nan
75-25	32 – 16 – 8 – 4 – 1	Adam	50	2.947
75-25	64 – 32 – 16 – 8 – 4 – 1	Adam	100	4.018
75-25	128 – 64 – 32 – 16 – 8 – 4 –1	Adam	300	2.244
70-30	32 – 16 – 8 – 4 – 1	SDG	50	nan
70-30	32 – 16 – 8 – 4 – 1	Adam	50	3.186
70-30	64 – 32 – 16 – 8 – 4 – 1	Adam	100	10.177
70-30	128 – 64 – 32 – 16 – 8 – 4 –1	Adam	300	0.078
60-40	32 – 16 – 8 – 4 – 1	SDG	50	nan
60-40	32 – 16 – 8 – 4 – 1	Adam	50	1.120
60-40	64 – 32 – 16 – 8 – 4 – 1	Adam	100	2.796
60-40	128 – 64 – 32 – 16 – 8 – 4 –1	Adam	300	0.533

Neural Network Plot for Observation



Train Test Split	70-30
Architecture	128 – 64 – 32 – 16 – 8 – 4 – 1
Optimizer	Adam
Epochs	300

FINDING THE BEST SPLIT

SPLIT	MAE (Before vif)
60-40	0.114 (Multiple regression)
75-25	0.085 (K Nearest Neighbour)
80-20	0.099 (Decision Tree)
70-30	0.106 (Support Vector Machine)
60-40	0.082 (XG Boost)
75-25	0.085 (Adaptive Boosting)
70-30	0.081 (Random Forest)
70-30	0.078 (Artificial Neural Network)

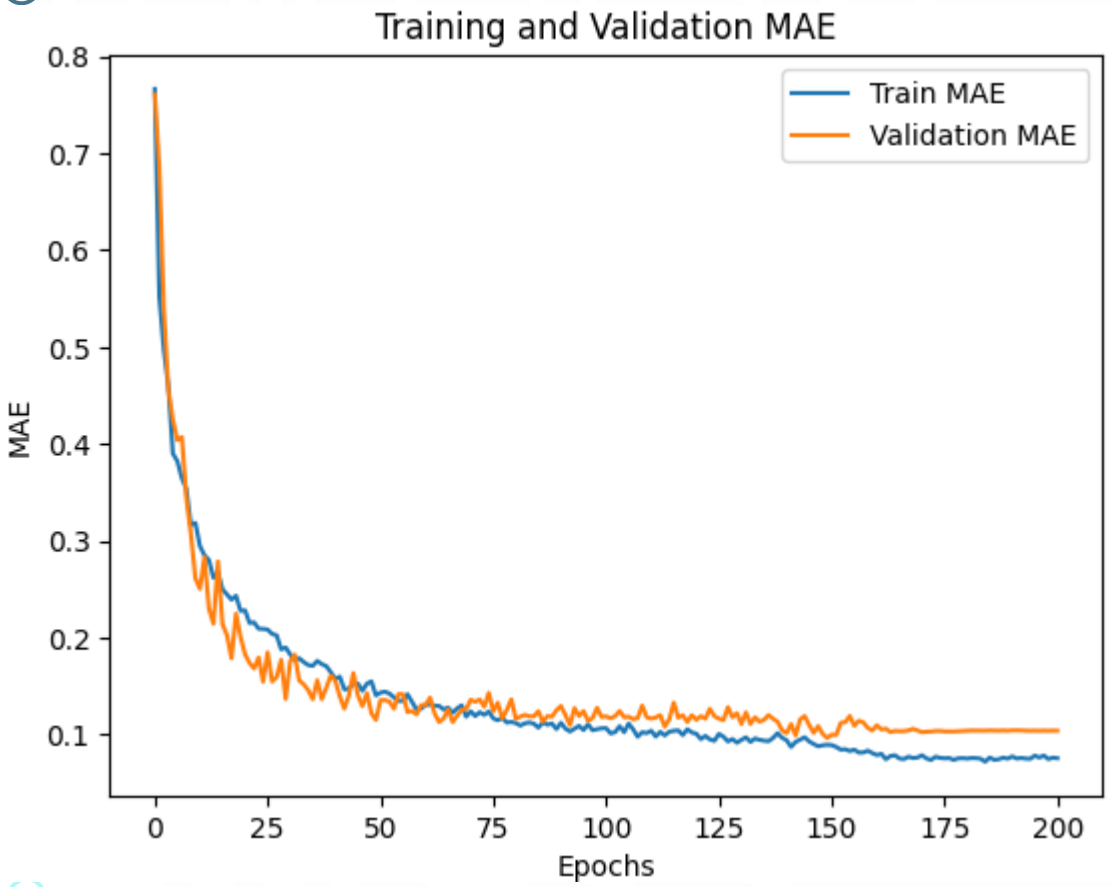
70-30 is the best split

FINDING THE BEST MODEL

SPLIT	MAE (Before vif)
70-30	0.112 (Multiple Regression)
70-30	0.846 (K Nearest Neighbour)
70-30	0.106 (Support Vector Machine)
70-30	0.100 (Decision Tree)
70-30	0.105 (XG boost)
70-30	0.085 (Adaptive Boosting)
70-30	0.081 (Random Forest)
70-30	0.036 (Artificial Neural Network)

Artificial Neural Network is the Best Model

Conclusion



From the above analysis of algorithms, it can be concluded that the **Artificial Neural Network** Algorithm has the lowest Mean Absolute Error.

Summary

After reviewing the above comparisons, it is evident that the Artificial Neural Network has the lowest MAE. Hence, it can be considered the best machine learning algorithm for predicting the productivity of Garment workers.

Future Scope

The current model can be further improved by adding much more advanced Algorithms such as Recurrent Neural Networks or Convolutional Neural Networks as they can identify more complex patterns efficiently.

It can be trained on a much larger and more diverse dataset.

Work Distribution

Team Member	Tasks
S.Jayavardhan	Introduction and Literature Review
J.Sai Dwarak	Data Pre-Processing and EDA
B.Poojashree	Data Modelling and Evaluation
G.Shyam	Algorithm Comparison & Summary

Thank You

Done By
G.Shyam (107222546014)
J.Sai Dwarak (107222546016)
B.Poojashree (107222546007)
S.Jayavardhan (107222546017)

Colab Notebook- Regression Project

https://colab.research.google.com/drive/1Ni4to0PRkwbG6OGY_bWGrI0RRL8GFJSF#scrollTo=SXPp-lEfv5OH

APPENDIX

Importing Libraries

```
[ ] !pip install statsmodels
    import numpy as np
    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 matplotlib.pyplot as plt

    # allow plots to appear directly in the notebook
    %matplotlib inline
```

```
Requirement already satisfied: statsmodels in /usr/local/lib/python3.10/dist-packages (0.14.4)
Requirement already satisfied: numpy<3,>=1.22.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (1.26.4)
Requirement already satisfied: scipy!=1.9.2,>=1.8 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (1.13.1)
Requirement already satisfied: pandas!=2.1.0,>=1.4 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (2.2.2)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (1.0.1)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (24.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas!=2.1.0,>=1.4->statsmodels) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas!=2.1.0,>=1.4->statsmodels) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas!=2.1.0,>=1.4->statsmodels) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas!=2.1.0,>=1.4->statsmodels) (1.17.0)
```

Uploading File

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

Data Preprocessing

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

	date	quarter	department	day	team	targeted_productivity	smv	wip	over_time	incentive	idle_time	idle_men	no_of_style_change	no_of_workers
0	1/1/2015	Quarter1	sweing	Thursday	8	0.80	26.16	1108.0	7080	98	0.0	0	0	59.0
1	1/1/2015	Quarter1	finishing	Thursday	1	0.75	3.94	NaN	960	0	0.0	0	0	8.0
2	1/1/2015	Quarter1	sweing	Thursday	11	0.80	11.41	968.0	3660	50	0.0	0	0	30.5
3	1/1/2015	Quarter1	sweing	Thursday	12	0.80	11.41	968.0	3660	50	0.0	0	0	30.5
4	1/1/2015	Quarter1	sweing	Thursday	6	0.80	25.90	1170.0	1920	50	0.0	0	0	56.0

Data Preprocessing

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

	date	quarter	department	day	team	targeted_productivity	smv	wip	over_time	incentive	idle_time	idle_men	no_of_style_change	no_of_workers
0	1/1/2015	Quarter1	sweing	Thursday	8	0.80	26.16	1108.0	7080	98	0.0	0	0	59.0
1	1/1/2015	Quarter1	finishing	Thursday	1	0.75	3.94	NaN	960	0	0.0	0	0	8.0
2	1/1/2015	Quarter1	sweing	Thursday	11	0.80	11.41	968.0	3660	50	0.0	0	0	30.5
3	1/1/2015	Quarter1	sweing	Thursday	12	0.80	11.41	968.0	3660	50	0.0	0	0	30.5
4	1/1/2015	Quarter1	sweing	Thursday	6	0.80	25.90	1170.0	1920	50	0.0	0	0	56.0

Converting 'Day' column into 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)
```

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

```
quarter
Quarter1     360
Quarter2     335
Quarter4     248
Quarter3     210
Quarter5      44
Name: count, dtype: int64
department
sweing         691
finishing      257
finishing      249
```

Assigning Feature and Target Variables

```
# Feature Variable
x=data.drop(['actual_productivity'],axis=1)
print(x)

# Target Variable
y=data['actual_productivity']
print(y)
```

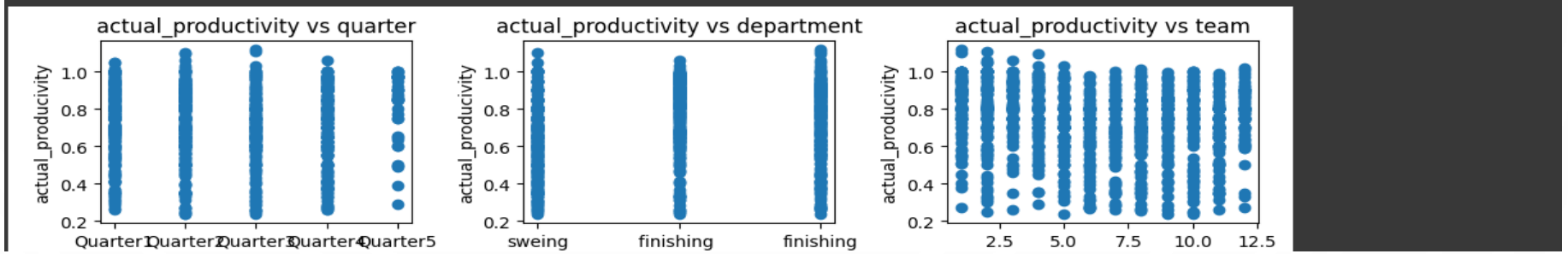
```
      quarter  department  team  targeted_productivity     smv  over_time  \
0    Quarter1       sweing     8                   0.80   26.16        7080
1    Quarter1   finishing      1                   0.75    3.94         960
2    Quarter1       sweing    11                   0.80   11.41        3660
3    Quarter1       sweing    12                   0.80   11.41        3660
4    Quarter1       sweing     6                   0.80   25.90        1920
...       ...          ...   ...                    ...     ...         ...
1192 Quarter2   finishing     10                   0.75    2.90         960
1193 Quarter2   finishing      8                   0.70    3.90         960
1194 Quarter2   finishing      7                   0.65    3.90         960
1195 Quarter2   finishing      9                   0.75    2.90        1800
1196 Quarter2   finishing      6                   0.70    2.90         720

      incentive  idle_time  idle_men  no_of_style_change  no_of_workers  \
0            98        0.0         0                   0           59.0
1             0        0.0         0                   0            8.0
2            50        0.0         0                   0           30.5
```

Exploratory Data Analysis

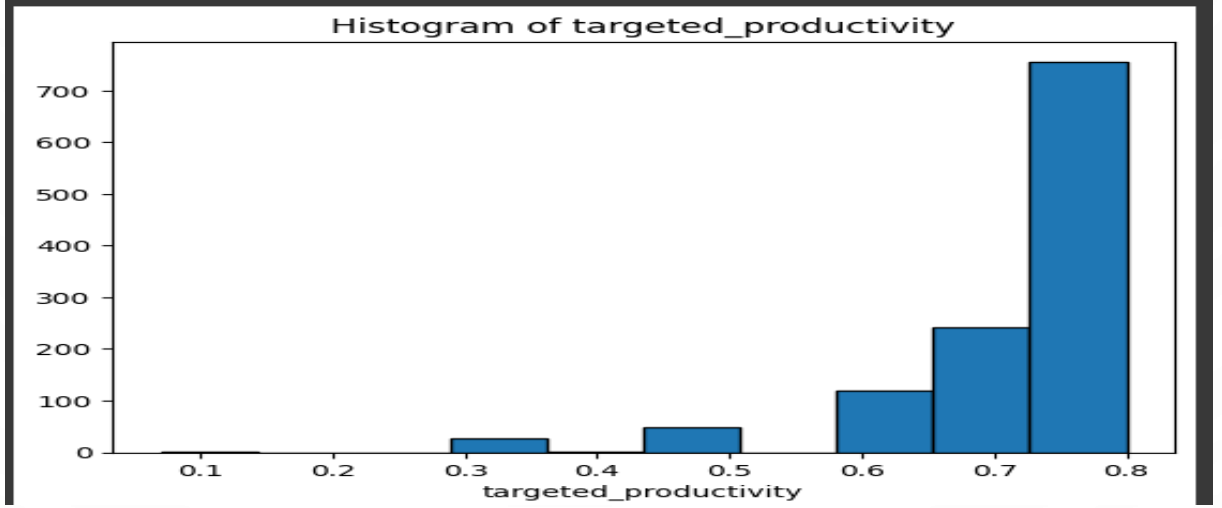
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1197 entries, 0 to 1196
Data columns (total 13 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   quarter                1197 non-null   object
 1   department             1197 non-null   object
 2   team                   1197 non-null   int64
 3   targeted_productivity  1197 non-null   float64
 4   smv                    1197 non-null   float64
 5   over_time              1197 non-null   int64
 6   incentive              1197 non-null   int64
 7   idle_time              1197 non-null   float64
 8   idle_men               1197 non-null   int64
 9   no_of_style_change     1197 non-null   int64
 10  no_of_workers          1197 non-null   float64
 11  actual_productivity    1197 non-null   float64
 12  day_binary             1197 non-null   int64
dtypes: float64(5), int64(6), object(2)
memory usage: 121.7+ KB
```

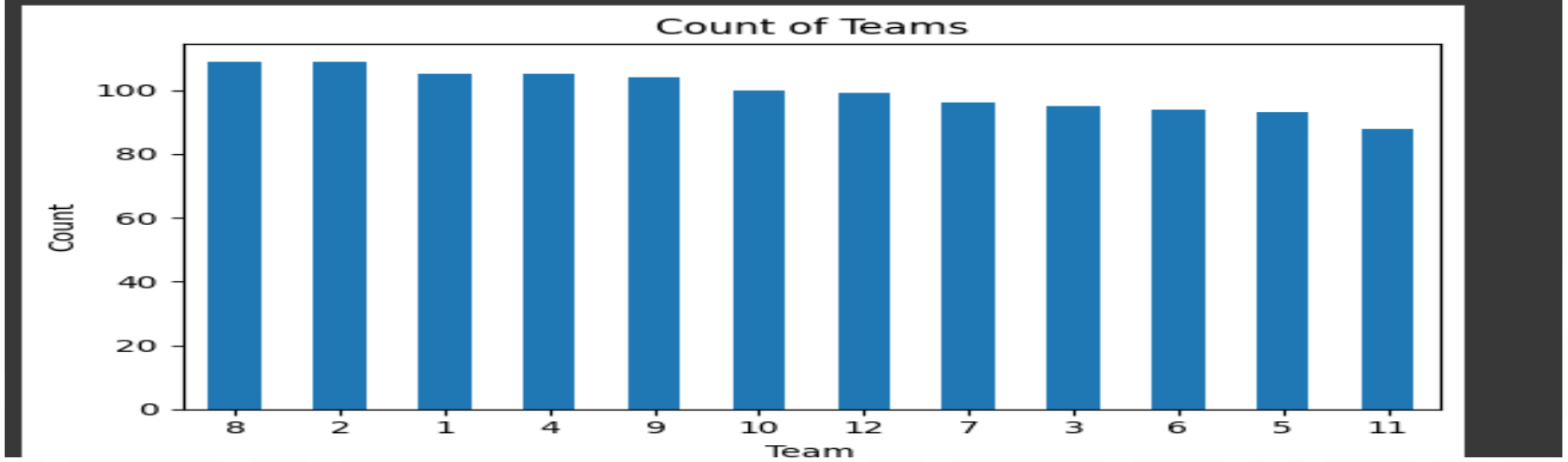
```python
# Scatter Plot

features = ["quarter","department","team","targeted_productivity","smv","over_time","incentive","idle_time","idle_men","no_of_style_change","no_of_workers","

# Create scatter plots
plt.figure(figsize=(10, 10))

for i, feature in enumerate(features):
    plt.subplot(4, 3, i+1)
    plt.scatter(data[feature],data['actual_productivity'])
    plt.xlabel(feature)
    plt.ylabel('actual_producivity')
    plt.title(f'actual_productivity vs {feature}')

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



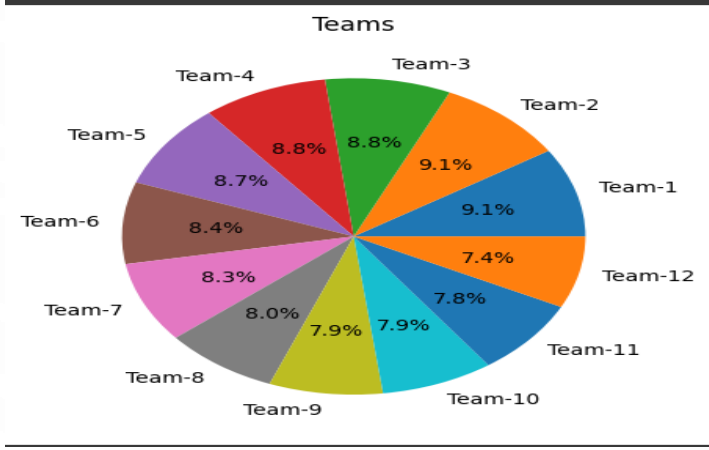

```python
targeted_productivity=data['targeted_productivity']
plt.hist(targeted_productivity,bins=10,edgecolor='black')
plt.xlabel('targeted_productivity')
plt.title('Histogram of targeted_productivity')
plt.show()
```



```
# Plotting the bar graph
plt.figure(figsize=(6, 4))
team.plot(kind='bar')
plt.title('Count of Teams')
plt.xlabel('Team')
plt.ylabel('Count')
plt.xticks(rotation=360)
plt.tight_layout()
plt.show()
```




```
# Pie chart for Teams

teams=data['team']
sizes=teams.value_counts()
plt.pie(sizes,labels=["Team-1","Team-2","Team-3","Team-4","Team-5","Team-6","Te
plt.title('Teams')
plt.show()
```



Machine Learning Algorithms

Linear Regression (Before VIF)

```
#Before the change, X was a pandas series representing one column of the dataframe.
#X = data['targeted_productivity']
#Changing it into a dataframe with double brackets [[...]] solves the problem
X = data[['targeted_productivity']]  # Use double brackets to create a DataFrame
y = data.actual_productivity
X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y, test_size=0.2,random_state=1)
lm2 = LinearRegression()
lm2.fit(X, y)
print(lm2.intercept_)
print(lm2.coef_)
```

```
0.1867874383341922
[0.7514793]
```

```
X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y, test_size=0.2,random_state=1)
X_train2,X_test2,y_train2,y_test2=train_test_split(X,y,test_size=0.25,random_state=1)
X_train3,X_test3,y_train3,y_test3=train_test_split(X,y,test_size=0.30,random_state=1)
X_train4,X_test4,y_train4,y_test4=train_test_split(X,y,test_size=0.40,random_state=1)
```

80-20 Train Test Split

```
lm1 = LinearRegression()
lm1.fit(X_train1, y_train1)
y_pred1 = lm1.predict(X_test1)
print(np.sqrt(metrics.mean_squared_error(y_test1, y_pred1)))
```

```
0.17452926704725188
```

75-25 Train Test Split

```
lm2 = LinearRegression()
lm2.fit(X_train2, y_train2)
y_pred2 = lm2.predict(X_test2)
print(np.sqrt(metrics.mean_squared_error(y_test2, y_pred2)))
```

```
0.171149235642958
```

70-30 Train Test Split

```
lm3 = LinearRegression()
lm3.fit(X_train3, y_train3)
y_pred3 = lm3.predict(X_test3)
print(np.sqrt(metrics.mean_squared_error(y_test3, y_pred3)))
```

Multiple Regression (Before VIF)

```
X = data[['team','targeted_productivity','smv','over_time','incentive','idle_time','idle_men','no_of_style_change','no_of_workers','day_binary','quarter_Quarter
y = data.actual_productivity
X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y, test_size=0.2,random_state=1)
lm2 = LinearRegression()
lm2.fit(X_train1, y_train1)
y_pred = lm2.predict(X_test1)
print(lm2.intercept_)
print(lm2.coef_)
print(np.sqrt(metrics.mean_squared_error(y_test1, y_pred)))
```

```
0.2611132888921823
[-7.68793302e-03  7.51725191e-01 -7.28277809e-03 -3.31873953e-06
  1.05884735e-04  6.07734611e-05 -9.15560764e-03 -2.92845217e-02
  4.58050443e-03  1.39640860e-03 -2.61779456e-03 -7.17313651e-03
 -2.21248721e-02 -2.05945589e-02  5.25103621e-02 -2.58408222e-02
 -2.58408222e-02 -6.36731158e-02]
0.16358553783429694
```

```
X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y, test_size=0.2,random_state=1)
X_train2,X_test2,y_train2,y_test2=train_test_split(X,y,test_size=0.25,random_state=1)
X_train3,X_test3,y_train3,y_test3=train_test_split(X,y,test_size=0.30,random_state=1)
X_train4,X_test4,y_train4,y_test4=train_test_split(X,y,test_size=0.40,random_state=1)
```

80-20 Train Test Split

```
[ ] !pip install scikit-learn # Make sure scikit-learn is installed
    from sklearn.metrics import r2_score # Import r2_score
    import numpy as np
    from sklearn.linear_model import LinearRegression
    from sklearn import metrics
    from sklearn.model_selection import train_test_split
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.5.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
```

```
lm1 = LinearRegression()
lm1.fit(X_train1, y_train1)
y_pred1 = lm1.predict(X_test1)
print(np.sqrt(metrics.mean_squared_error(y_test1, y_pred1)))
r2_score(y_test1,y_pred1)
print(metrics.mean_absolute_error(y_test1,y_pred1))
```

```
0.16358553783429694
0.11899196179151429
```

75-25 Train Test Split

```
lm2 = LinearRegression()
lm2.fit(X_train2, y_train2)
y_pred2 = lm2.predict(X_test2)
print(np.sqrt(metrics.mean_squared_error(y_test2, y_pred2)))
print(r2_score(y_test2,y_pred2))
print(metrics.mean_absolute_error(y_test2,y_pred2))
```

```
0.16094059089374735
0.20424904205514338
0.1153599868422022
```

70-30 Train Test Split

```
[ ] lm3 = LinearRegression()
    lm3.fit(X_train3, y_train3)
    y_pred3 = lm3.predict(X_test3)
    print(np.sqrt(metrics.mean_squared_error(y_test3, y_pred3)))
    print(r2_score(y_test3,y_pred3))
    print(metrics.mean_absolute_error(y_test3,y_pred3))
```

```
0.15914162981507382
0.19925723507282678
0.11223180112217462
```