PROJECT REPORT

ON

# "GARMENT EMPLOYEE PRODUCTIVITY VISUALIZATION AND PREDICTION USING MACHINE LEARNING MODEL"

*For the partial fulfillment of*

**M.TECH IN DECISION SCIENCES AND ENGINEERING,**

IIT (BHU), Varanasi

**2021**

**UNDER THE GUIDANCE OF**

**Dr. Cherian Samuel**

**BY**

**Mahesh Purbia**

**DEPARTMENT OF MECHANICAL ENGINEERING**

**Indian Institute of Technology (BHU)**

**VARANASI, UTTAR PRADESH**

**INDIA**

# DECLARATION

This is to certify that the minor project work presented on the topic "**GARMENT EMPLOYEE PRODUCTIVITY VISUALIZATION AND PREDICTION USING MACHINE LEARNING MODEL**" is an original work. The work has been carried out by:

- **MAHESH PURBIA**

1st Semester student (M. Tech in Decision Sciences and Engineering) of Department of Mechanical Engineering, Indian Institute of Technology (BHU), Varanasi, under the guidance of **DR. CHERIAN SAMUEL** for the partial fulfillment of the course curriculum of M. Tech degree in Decision Sciences and Engineering.

(DR.CHERIAN.SAMUEL)

**(MAHESH PURBIA)**

# ACKNOWLEDGEMENT

It is very True that every mission needs a spirit of dedication and hard work. But more than anything else, it requires appropriate guidance to reach the goal at right time.

I Sincerely thank my advisor**, Dr. Cherian Samuel,** Professor in Mechanical Engineering for his perennial guidance, suggestion and continuous support throughout my Project. I greatly appreciate all the support that he has been given to me.

The Purpose to do this project and making the report was to observe practically about the courses that I am Studying while pursuing my masters at IIT BHU, Varanasi in Decision Sciences and engineering.

**Mahesh Purbia**

# CONTENT

# INTRODUCTION

The words clothing, food, and shelter are basic human needs that are familiar to the ears. Clothing is one of the basic needs for the human. It is inconceivable if these needs were not met. Besides the need for clothing in terms of function, clothing sales or business is also very potent

Apart from an economic standpoint, the garment industry is one of the most labor-intensive industries in the world. About 75 million people are directly involved in the textile, clothing, and footwear sectors worldwide (United, 2021).

The Garment Industry is one of the key examples of the industrial globalization of this modern era. It is a **highly labour-intensive industry** with lots of manual processes.

Satisfying the huge global demand for garment products is mostly **dependent on the production and delivery performance** of the employees in the garment manufacturing companies.

A common problem in Textile/Garment industry is that the actual productivity of apparel employees sometimes fails to reach the productivity targets set by the authorities to meet production targets on time, resulting in huge losses.
So, it is highly desirable among the decision makers in the garments industry to track, analyse and predict the productivity performance of the working teams in their factories.

The ready-to-wear clothing industry is a major part of manufacturing production, employment, and trade in many developing countries, for example, Bangladesh, which is now the second-largest exporter of apparel in the world after China (Chaerani, 2018).

According to the recently released Bureau of Export Promotion Data, Bangladesh's export revenue from the ready-to-wear sector is around $ 30.61 billion and ready-to-wear clothing holds nearly 14.07% of Bangladesh's GDP as well as 81% of total export revenue. (M Saiful Islam et al., 2019).

# *Literature Review*

With the increasing demand for clothing needs around the world, the increase in the quality of production in the garment industry must be maintained and improved.
One of the business performance measurement tools used is productivity, where the definition of productivity itself is a comparison between output and all sources used (input) (Sri & Margareta, 2020). Technology is a tool used to accelerate productivity (Afani, Utari Nur. dan Solovida, 2019).

With better employee productivity with employee productivity getting better, the quality and quantity of the product produced will not be doubted every time it is produced.

But before increasing the productivity of employee performance, it is **necessary to know in advance what factors affect and how to predict employee productivity**, especially garment employees that are being discussed

# *Problem Solution - Approach*

In past many researchers have studied such issues related to employee productivity. Some of them are mentioned as under.

**Table 1.** ***Research Literature***

| **Author** | ***Research Problem*** | ***Literature Supports*** |
|---|---|---|
| Abu Hamja, Malek Maalouf & Peter Hasle (Hamja et al., 2019) | Lean is increasingly being used by garment manufacturers to increase productivity to reduce costs and waiting times. However, it has not been documented in the review whether lean increases productivity, and it is an open question whether lean can increase productivity without compromising health and safety work. | The solution is to explore and collect a systematic review of the available review literature Research on Lean in Clothing and Its Associated Effects on Productivity and Safety |
| Lila Ayu Ratna Winanda (Winanda, 2010) | The problems of construction workers cannot be separated from the resulting productivity. Therefore, this article discusses the methods of factors affecting the productivity of construction workers, so that in the end we can estimate the productivity of construction workers themselves. | The solution is to take a probabilistic approach to the neural networks used for mapping the productivity and productivity factors of the workers themselves. |
| Doni Efriza & Iswandi Idris (Doni Efriza, 2018) | Measurement of employee productivity is used as a management tool to analyze and encourage efficiency so that increased productivity will provide a greater ability for companies to increase employee wages, which in turn will stimulate employee morale and morale. | The solution is to distribute questionnaires to bank employees interviewed in Medan City to see the variables of motivation, knowledge, skills, and income levels of the factors that affect work efficiency |

In addition to the cases described in the previous paper on solving productivity problems, a researcher from Indonesia, (Gunawan et al., 2010) has identified that increased managerial ability to monitor finances and evaluate activities in medium-scale Indonesian garment factories is essential to sustain the Indonesian economy.

However, it is not enough to increase managerial abilities, because of the limitations of each person in the managerial ranks itself.

Need help with data mining processing or data mining that can make predictions in this case.

**Reference Articles :**

1. Deep Neural Network Approach for Predicting the Productivity of Garment Employees. Imran, A. A., Amin, M. N., Islam Rifat, M. R., & Mehreen, S. (2019).

**Result**

| Title | Measurement Units |
|---|---|
| Deep Neural Network Approach for Predicting the Productivity of Garment Employees (2019) | ➢ **MSE** = 0.086<br>➢ **MAE** = 0.018<br>➢ **MAPE** = 15.932 |

By using the Deep Neural Network (DNN) model, the experimental results of this study have shown that the proposed model produces promising predictive performance with a minimum Mean Absolute Error of 0.086 which is less than the basic performance error of 0.15. Such predictive performance can help producers to set accurate targets, minimize production losses and maximize profits.

2. Garment Employee Productivity Prediction Using Random Forest (Imanuel Balla 1; Sri Rahayu 2) ; Jajang Jaya Purnama, 2021)

**Result**

**Title -** Garment Employee Productivity Prediction Using Random Forest (2021)

| **Algorithms** | **Correlation Coefficients** | **MAE** | **RMSE** |
|---|---|---|---|
| Random Forest | 0.7071 | 0.0787 | 0.1236 |
| Linear Regression | 0.5173 | 0.1081 | 0.1494 |
| Neural Network | 0.4169 | 0.1218 | 0.1763 |

**Conclusion:**

Study produces the correlation coefficient, MAE, and RMSE values for each of the models applied. Because the aim is to find the smallest MAE value, the random forest model, in this case, is most appropriate

# *MATERIALS AND METHODS*

In a study, of course, the main material is a dataset to be used as machine learning material using algorithms. In this study, the dataset used is garments worker productivity, which is a public dataset because it is taken from the **UCI repository** website.

The dataset used in this study was published in 2020 with 15 attributes including date, day, quarter, department, team_no, no_of_workers, no_of_style_change, targeted_productivity, SMV, wip, over_time, incentive, idle_time, idle_men, actual_productivity with continuous actual_productivity classes. , has 1197 instances.

From the specification of the garments worker productivity dataset, which has 15 attributes and 1197 data as described and shown below.

**Attribute Information:**

1. **date** : Date in MM-DD-YYYY
2. **day** : Day of the Week
3. **quarter** : A portion of the month. A month was divided into four quarters
4. **department** : Associated department with the instance
5. **teamno** : Associated team number with the instance
6. **noofworkers** : Number of workers in each team
7. **no of style change** : Number of changes in the style of a particular product
8. **targetedproductivity** : Targeted productivity set by the Authority for each team for each day.
9. **smv** : Standard Minute Value, it is the allocated time for a task
10. **wip** : Work in progress. Includes the number of unfinished items for products
11. **overtime** : Represents the amount of overtime by each team in minutes
12. **incentive** : Represents the amount of financial incentive (in BDT) that enables or motivates a particular course of action.
13. **idletime** : The amount of time when the production was interrupted due to several reasons

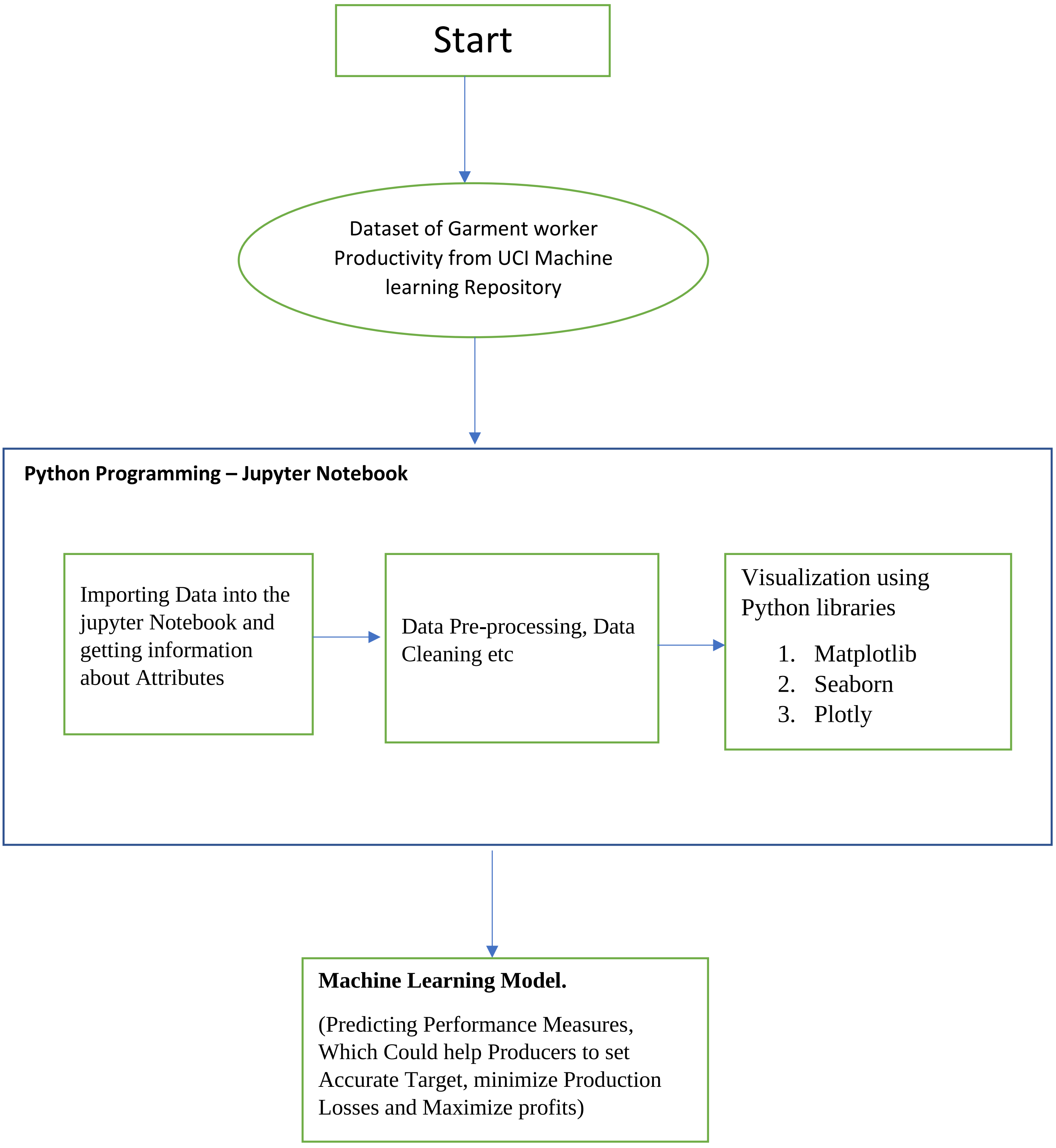
**14.idlemen** : The number of workers who were idle due to production interruption

**15.actual_productivity** : The actual % of productivity that was delivered by the workers. It ranges from 0-1.

By using above attributes provided in the dataset, we will use python programming language to visualise the insights using different visualization library such as Matplotlib, seaborn, plotly and prediction to be carried out using Machine learning Model.

# ***Plan Of Work***

## ➢ Flow Chart



Start
Dataset of Garment worker Productivity from UCI Machine learning Repository
Python Programming – Jupyter Notebook
Importing Data into the jupyter Notebook and getting information about Attributes
Data Pre-processing, Data Cleaning etc
Visualization using Python libraries
1. Matplotlib
2. Seaborn
3. Plotly
Machine Learning Model.
(Predicting Performance Measures, Which Could help Producers to set Accurate Target, minimize Production Losses and Maximize profits)

# Python Code

## I. Data Pre-processing

The most essential phase of solving a real-world problem using machine learning is to pre-process the dataset properly. It is one of the primary key factors for obtaining a promising result from the model.
In real world context, data can be found noisy, malformed, incomplete, missing and imbalanced.
Before fitting the data into a machine learning model, it is necessary to make the data suitable for the model so that it can learn the underlying patterns in the data precisely.
In a machine learning pipeline, the data preprocessing phase may include a bunch of tasks according to the form and structure of the data such as data cleaning, feature encoding, instance selection, normalization, transformation, feature extraction and selection.
In this study, we have structured this phase according to the form of our dataset which includes feature engineering, feature encoding, feature scaling, and data partitioning.

### (1) Feature Engineering

Feature engineering is the act of extracting features from raw data and transforming them into formats that are suitable for the machine learning model

### (2) Feature Encoding

Feature encoding is the process of representing categorical variables into numeric form. As the neural networks can only learn from numeric data, it is mandatory to encode the categorical data into numbers before feeding the data to the model.

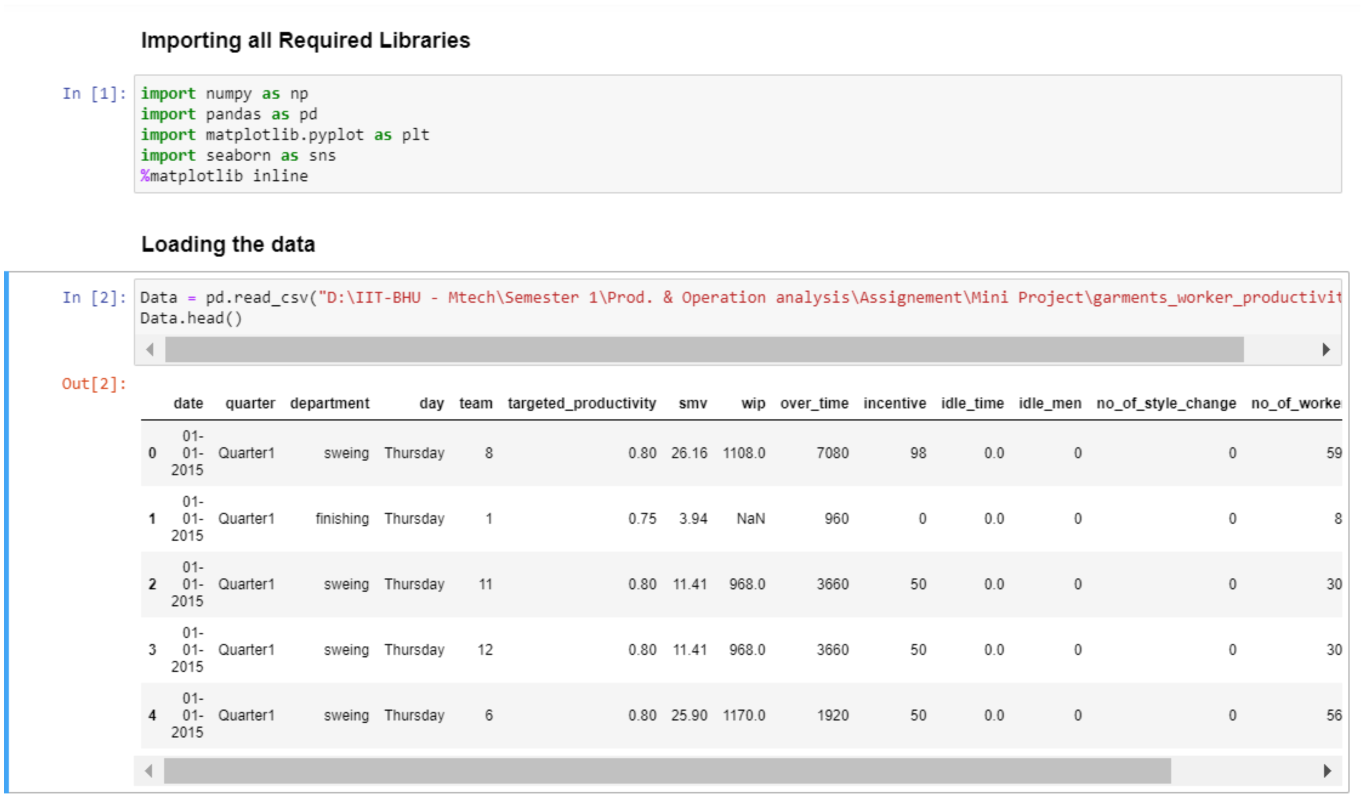
### (3) Feature Scaling

Feature scaling is a data standardization process which scales the data to a fixed range and ends up with smaller standard deviations, which suppresses the effect of outliers. It is a very essential part in terms of training a neural network based model as it helps the neural network weights converge faster.

### (4) Data Partitioning

Data partitioning in data mining is the division of the whole data available into two or three non-overlapping sets: the training set , the validation set , and the test set . If the data set is very large, often only a portion of it is selected for the partitions. Partitioning is normally used when the model for the data at hand is being chosen from a broad set of models.

## Step 1 Importing all Required Libraries and getting insight about the data set.



### Importing all Required Libraries

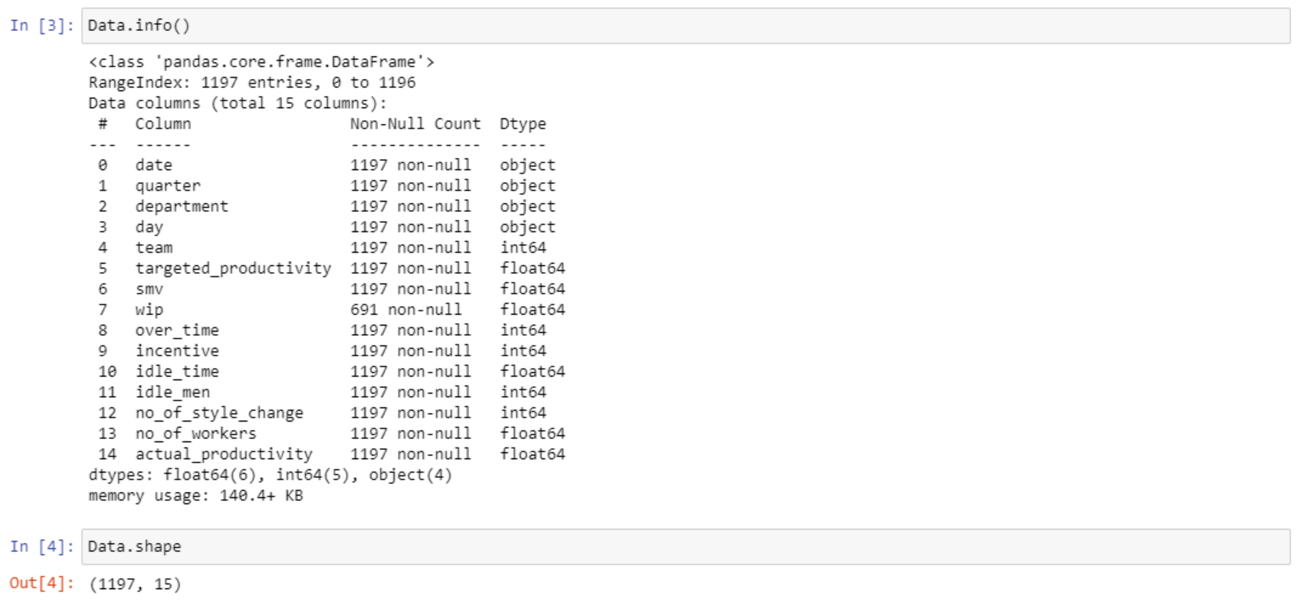
In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

### Loading the data

In [2]:
```
Data = pd.read_csv("D:\IIT-BHU - Mtech\Semester 1\Prod. & Operation analysis\Assignement\Mini Project\garments_worker_productivit
Data.head()
```

Out[2]:

| | date | quarter | department | day | team | targeted_productivity | smv | wip | over_time | incentive | idle_time | idle_men | no_of_style_change | no_of_worke |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 01-01-2015 | Quarter1 | sweing | Thursday | 8 | 0.80 | 26.16 | 1108.0 | 7080 | 98 | 0.0 | 0 | 0 | 59 |
| 1 | 01-01-2015 | Quarter1 | finishing | Thursday | 1 | 0.75 | 3.94 | NaN | 960 | 0 | 0.0 | 0 | 0 | 8 |
| 2 | 01-01-2015 | Quarter1 | sweing | Thursday | 11 | 0.80 | 11.41 | 968.0 | 3660 | 50 | 0.0 | 0 | 0 | 30 |
| 3 | 01-01-2015 | Quarter1 | sweing | Thursday | 12 | 0.80 | 11.41 | 968.0 | 3660 | 50 | 0.0 | 0 | 0 | 30 |
| 4 | 01-01-2015 | Quarter1 | sweing | Thursday | 6 | 0.80 | 25.90 | 1170.0 | 1920 | 50 | 0.0 | 0 | 0 | 56 |



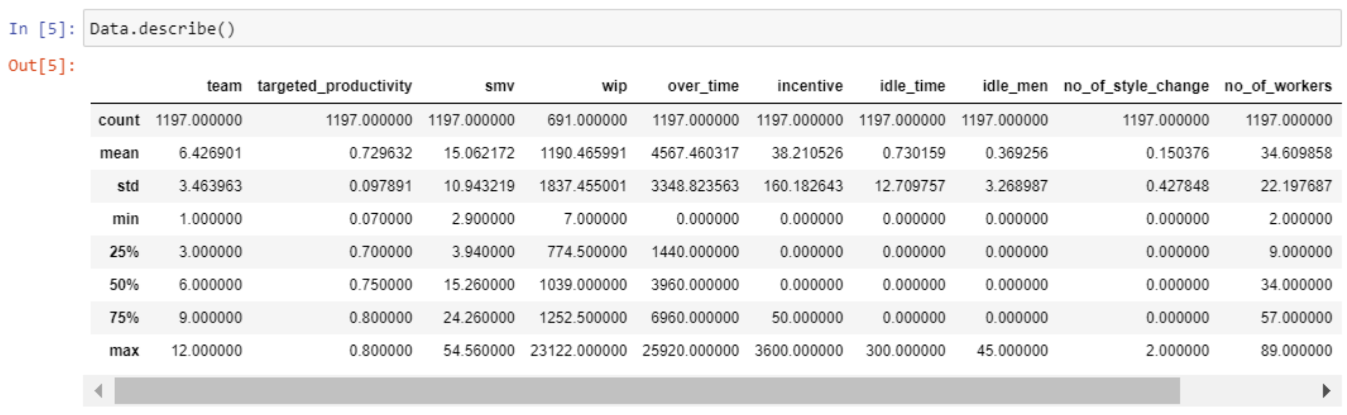
In [3]:
```
Data.info()
```

```
<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
```

In [4]:
```
Data.shape
```

Out[4]:
```
(1197, 15)
```

- From the given data set, we found that there are total 15 attributes with 1197 instances/entries.
- We also found out that there is about 691 missing values for the attribute, “wip”. So We need to fix this.



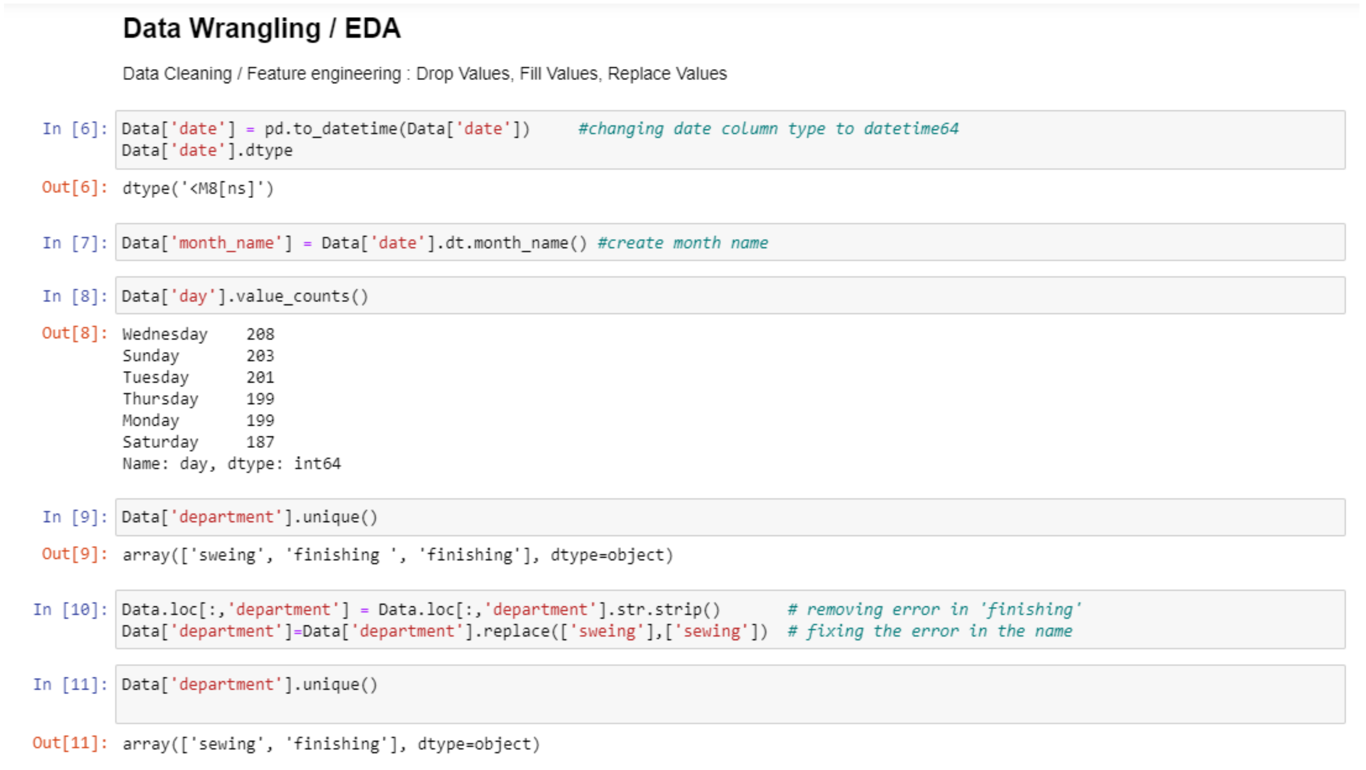
```
In [5]: Data.describe()
Out[5]:
```

| | team | targeted_productivity | smv | wip | over_time | incentive | idle_time | idle_men | no_of_style_change | no_of_workers |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 1197.000000 | 1197.000000 | 1197.000000 | 691.000000 | 1197.000000 | 1197.000000 | 1197.000000 | 1197.000000 | 1197.000000 | 1197.000000 |
| mean | 6.426901 | 0.729632 | 15.062172 | 1190.465991 | 4567.460317 | 38.210526 | 0.730159 | 0.369256 | 0.150376 | 34.609858 |
| std | 3.463963 | 0.097891 | 10.943219 | 1837.455001 | 3348.823563 | 160.182643 | 12.709757 | 3.268987 | 0.427848 | 22.197687 |
| min | 1.000000 | 0.070000 | 2.900000 | 7.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 2.000000 |
| 25% | 3.000000 | 0.700000 | 3.940000 | 774.500000 | 1440.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 9.000000 |
| 50% | 6.000000 | 0.750000 | 15.260000 | 1039.000000 | 3960.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 34.000000 |
| 75% | 9.000000 | 0.800000 | 24.260000 | 1252.500000 | 6960.000000 | 50.000000 | 0.000000 | 0.000000 | 0.000000 | 57.000000 |
| max | 12.000000 | 0.800000 | 54.560000 | 23122.000000 | 25920.000000 | 3600.000000 | 300.000000 | 45.000000 | 2.000000 | 89.000000 |

- The describe() method is used for calculating some statistical data like **percentile, mean** and **std** of the numerical values of the Series or DataFrame. It analyzes both numeric and object series and also the DataFrame column sets of mixed data types.

  This method helps us to understand, what sort of data we have and how the data is distributed and gives initial insights.

## Step 2 Data Cleaning / Feature Engineering : Drop , fill , replace etc.

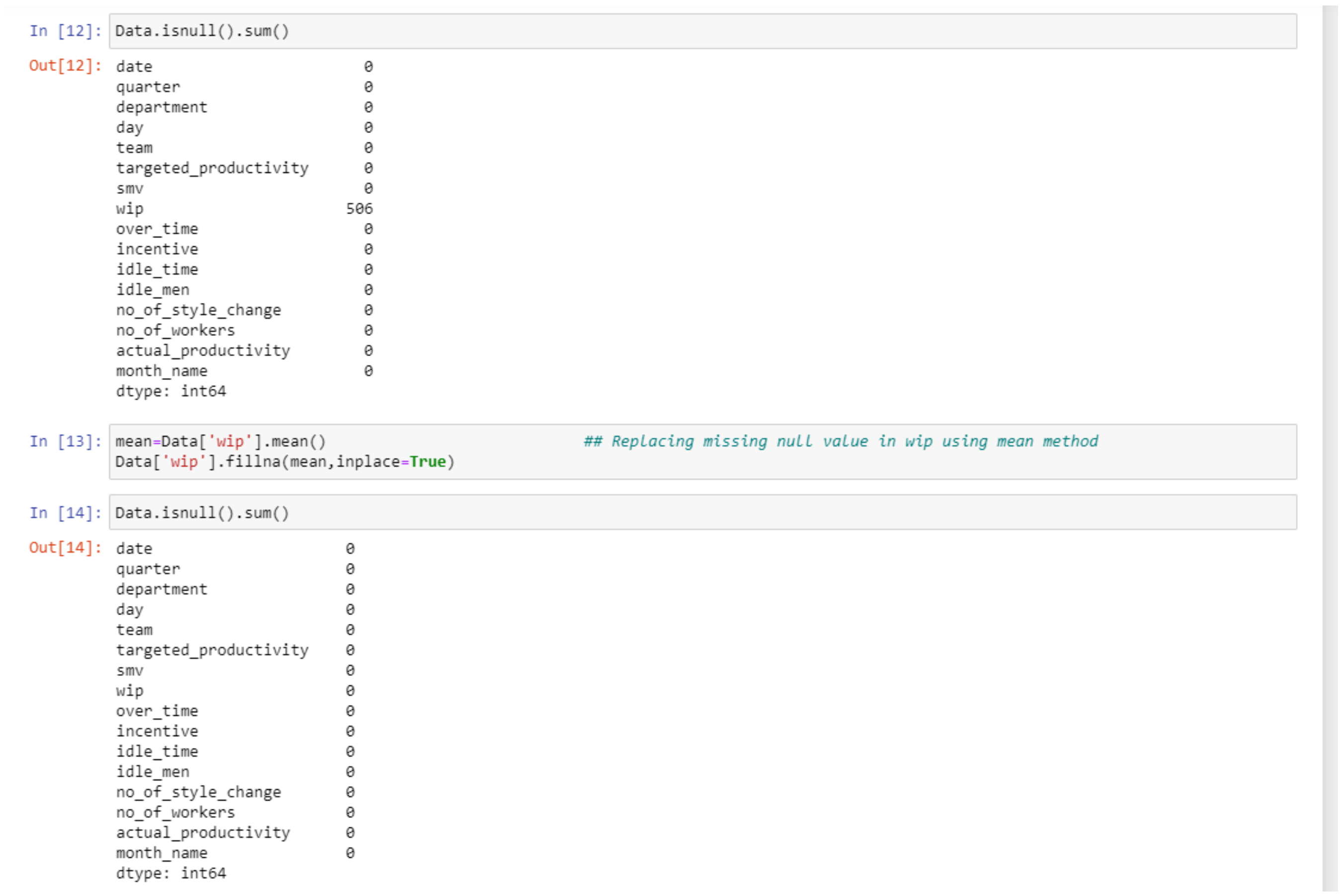


### Data Wrangling / EDA

Data Cleaning / Feature engineering : Drop Values, Fill Values, Replace Values

```
In [6]: Data['date'] = pd.to_datetime(Data['date'])     #changing date column type to datetime64
        Data['date'].dtype

Out[6]: dtype('<M8[ns]')

In [7]: Data['month_name'] = Data['date'].dt.month_name() #create month name

In [8]: Data['day'].value_counts()

Out[8]: Wednesday    208
        Sunday       203
        Tuesday      201
        Thursday     199
        Monday       199
        Saturday     187
        Name: day, dtype: int64

In [9]: Data['department'].unique()

Out[9]: array(['sweing', 'finishing ', 'finishing'], dtype=object)

In [10]: Data.loc[:,'department'] = Data.loc[:,'department'].str.strip()      # removing error in 'finishing'
         Data['department']=Data['department'].replace(['sweing'],['sewing'])  # fixing the error in the name

In [11]: Data['department'].unique()

Out[11]: array(['sewing', 'finishing'], dtype=object)
```

- For the day data – we don't find any entry for Friday, suggests us that the Friday could be a holiday.
- For the department data we found that because of spaces and spelling mistakes it showing wrong information such as 'Sweing' in place of 'Sewing' and also it was considering 3 department because of extra space i.e 'finishing ' & 'finishing' so we fixed this and made 2 unique entries for Department Data.



```
In [12]: Data.isnull().sum()
```

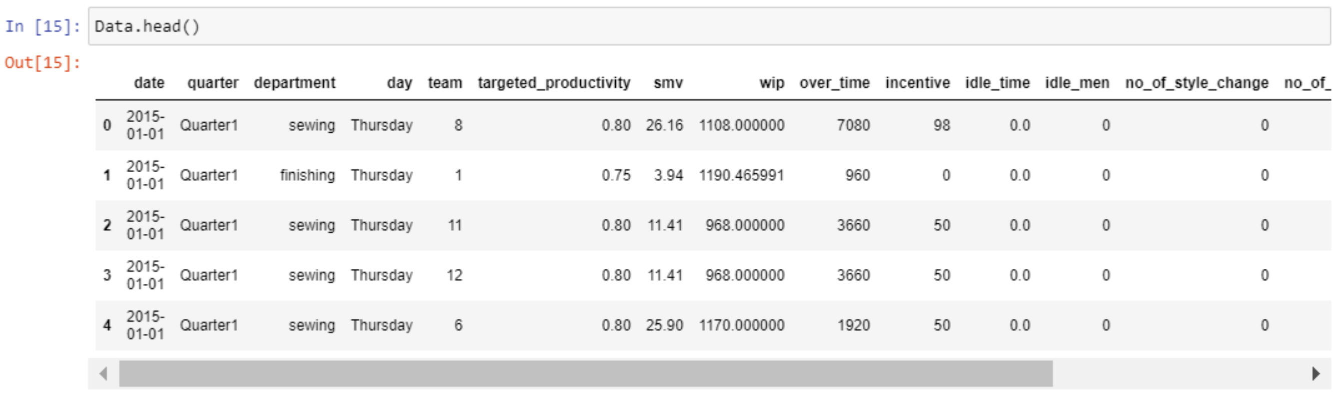
```
Out[12]: date                       0
         quarter                    0
         department                 0
         day                        0
         team                       0
         targeted_productivity      0
         smv                        0
         wip                      506
         over_time                  0
         incentive                  0
         idle_time                  0
         idle_men                   0
         no_of_style_change         0
         no_of_workers              0
         actual_productivity        0
         month_name                 0
         dtype: int64
```

```
In [13]: mean=Data['wip'].mean()                          ## Replacing missing null value in wip using mean method
         Data['wip'].fillna(mean,inplace=True)
```

```
In [14]: Data.isnull().sum()
```

```
Out[14]: date                     0
         quarter                  0
         department               0
         day                      0
         team                     0
         targeted_productivity    0
         smv                      0
         wip                      0
         over_time                0
         incentive                0
         idle_time                0
         idle_men                 0
         no_of_style_change       0
         no_of_workers            0
         actual_productivity      0
         month_name               0
         dtype: int64
```

- ➢ To fix the null values for 'wip' dataset, we use mean method to replace all the missing values in the dataset.



```
In [15]: Data.head()
```

Out[15]:

| | date | quarter | department | day | team | targeted_productivity | smv | wip | over_time | incentive | idle_time | idle_men | no_of_style_change | no_of_ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2015-01-01 | Quarter1 | sewing | Thursday | 8 | 0.80 | 26.16 | 1108.000000 | 7080 | 98 | 0.0 | 0 | 0 | |
| **1** | 2015-01-01 | Quarter1 | finishing | Thursday | 1 | 0.75 | 3.94 | 1190.465991 | 960 | 0 | 0.0 | 0 | 0 | |
| **2** | 2015-01-01 | Quarter1 | sewing | Thursday | 11 | 0.80 | 11.41 | 968.000000 | 3660 | 50 | 0.0 | 0 | 0 | |
| **3** | 2015-01-01 | Quarter1 | sewing | Thursday | 12 | 0.80 | 11.41 | 968.000000 | 3660 | 50 | 0.0 | 0 | 0 | |
| **4** | 2015-01-01 | Quarter1 | sewing | Thursday | 6 | 0.80 | 25.90 | 1170.000000 | 1920 | 50 | 0.0 | 0 | 0 | |

## II. Visualization using python Libraries

Data visualization is a crucial step in any exploratory data analysis or report. python has many libraries that facilitate creating a well-constructed visualization. We will use three libraries, **Matplotlib, Seaborn and Plotly**.

### 1. Matplotlib

Matplotlib is the standard graphing library in python. It is functionally integrated with pandas and numpy for easy and efficient plotting. Furthermore, Matplotlib gives the user full control over fonts, graph styling and axes properties, though this control comes at the potential cost of lengthy blocks of code. Matplotlib is especially good for performing exploratory analysis because of the integration with pandas, allowing for quick transformations from dataframe to graph. Matplotlib is particularly good for creating basic plots like scatter plots, bargraphs and lineplots, but looks a little rough when creating more complex plots like polar scatterplots.

### 2. Seaborn

Seaborn is a library built on top of the pyplot module in Matplotlib. It provides a high-level interface to create a more intuitive feel. This entails using a simpler syntax and more intuitive parameter settings.
Additionally, Seaborn includes a more aesthetically pleasing collection of colors, themes and styles. This produces a smoother and more professional looking plot than those created from the pyplot module. This library is especially useful when creating more complex plots where more refined graphics.

### 3. Plotly :

Unlike Matplotlib and Seaborn, Plotly is used to make interactive charts. While the plots look very similar to those produced by Seaborn in terms of graphics, they have the added utility of displaying information when a user hovers their mouse over the chart. This effect is accomplished by utilizing JavaScript behind the scenes and is a particularly useful feature when looking at busy or complex charts as you are immediately able to select the information that you are interested in. The drawback to using charts in Plotly, is that the code can get a bit complex and quite long depending on the method being used.

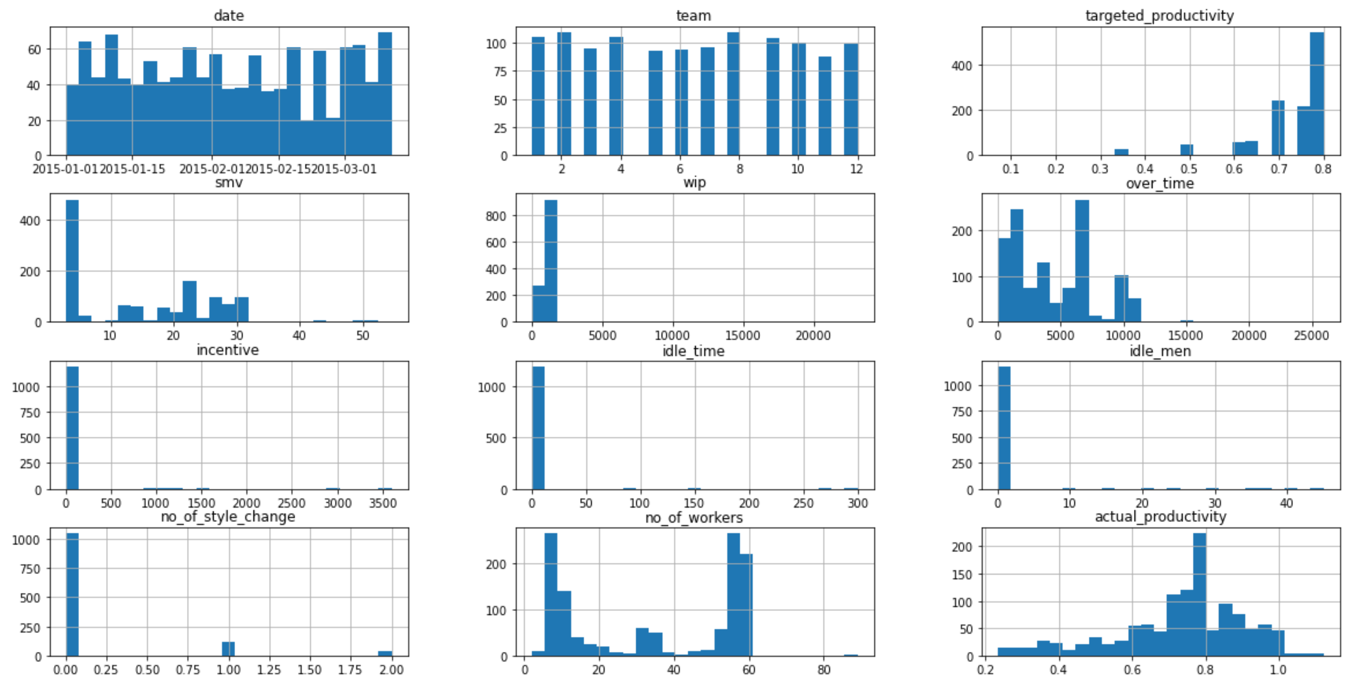
# Visualization from the Data-set Provided using python Libraries

## Visualization for Productivity of garment workers

### Import Plotly library

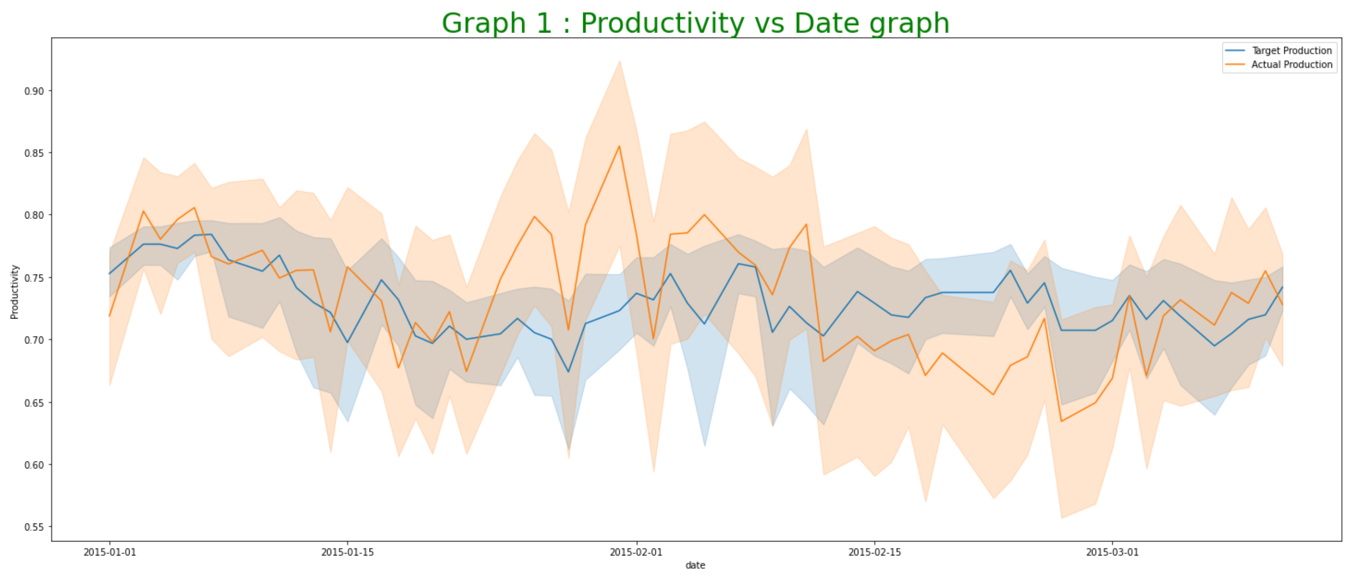
```
In [16]: import plotly
         import plotly_express as px
         import plotly.figure_factory as ff
         import plotly.graph_objects as go
         import plotly.io as pio

         from plotly.subplots import make_subplots
         from plotly.offline import plot, iplot,init_notebook_mode
```

```
In [17]: Data.hist(bins=25, figsize=(20,10))  #using hist() method we can get histograms of all numerical data in the dataset.
                                              #Helps in quick understanding of the distribution of certain numerical variables within it

Out[17]: array([[<AxesSubplot:title={'center':'date'}>,
                 <AxesSubplot:title={'center':'team'}>,
                 <AxesSubplot:title={'center':'targeted_productivity'}>],
                [<AxesSubplot:title={'center':'smv'}>,
                 <AxesSubplot:title={'center':'wip'}>,
                 <AxesSubplot:title={'center':'over_time'}>],
                [<AxesSubplot:title={'center':'incentive'}>,
                 <AxesSubplot:title={'center':'idle_time'}>,
                 <AxesSubplot:title={'center':'idle_men'}>],
                [<AxesSubplot:title={'center':'no_of_style_change'}>,
                 <AxesSubplot:title={'center':'no_of_workers'}>,
                 <AxesSubplot:title={'center':'actual_productivity'}>]],
               dtype=object)
```

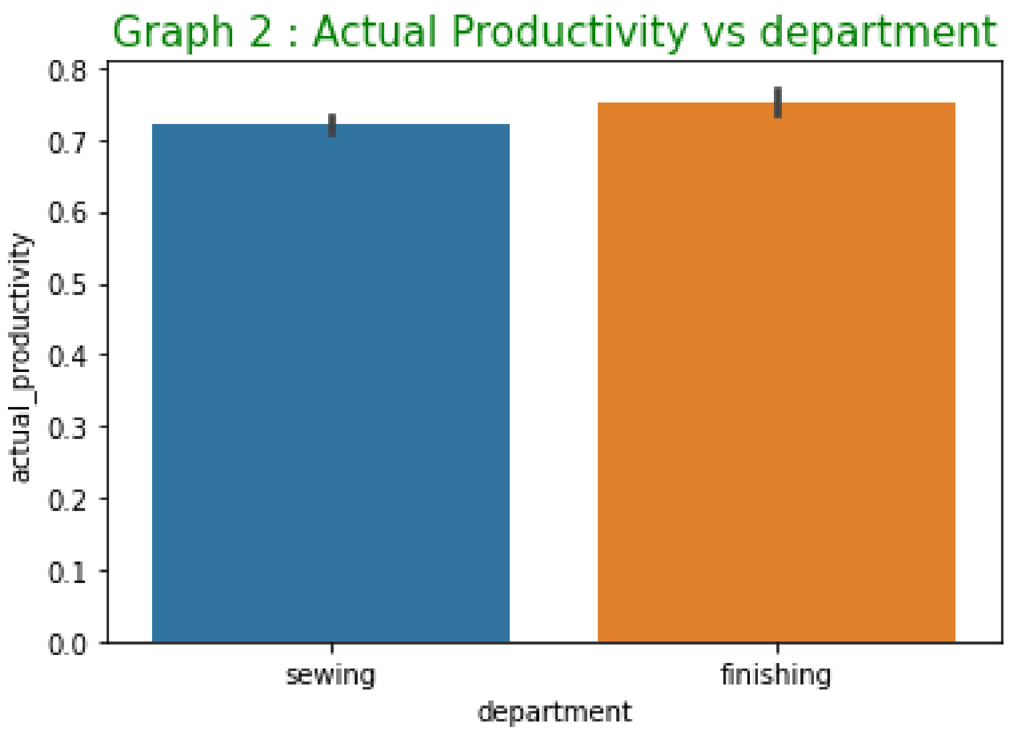


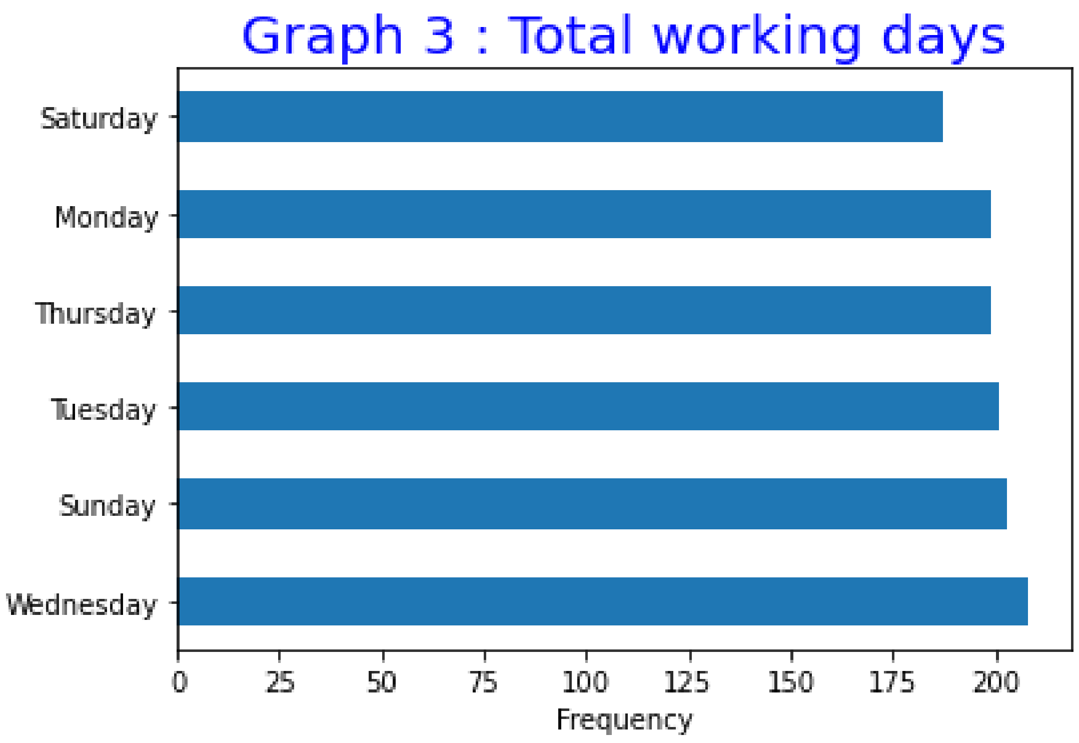
- ➢ A histogram is basically used to represent data provided in a form of some groups. It is accurate method for the graphical representation of numerical data distribution. It is a type of bar plot where X-axis represents the bin ranges while Y-axis gives information about frequency.
- ➢ To get the initial insights about the numerical data in the dataset histogram is very helpful.

## Following Graphs, we plotted using Python Libraries.

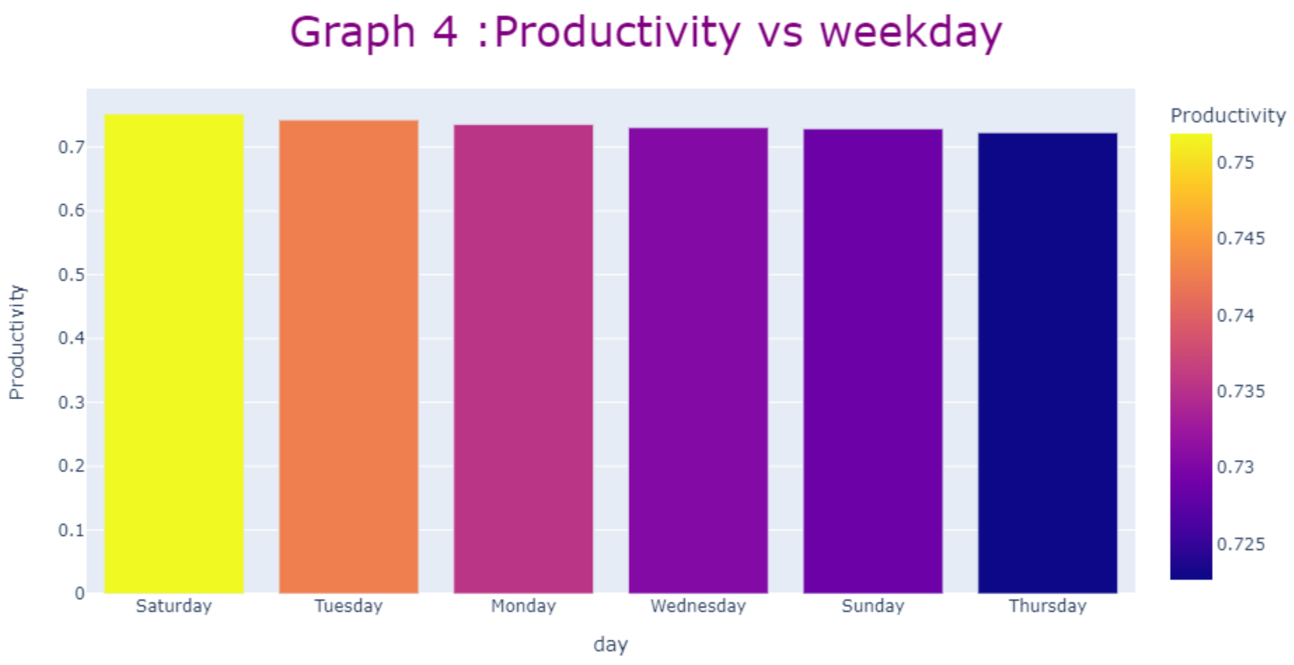


- In month of February Actual productivity found to be high as compared to target productivity suggesting overperformance, but in month of march we observed that actual productivity was found lower as compared to set target productivity.

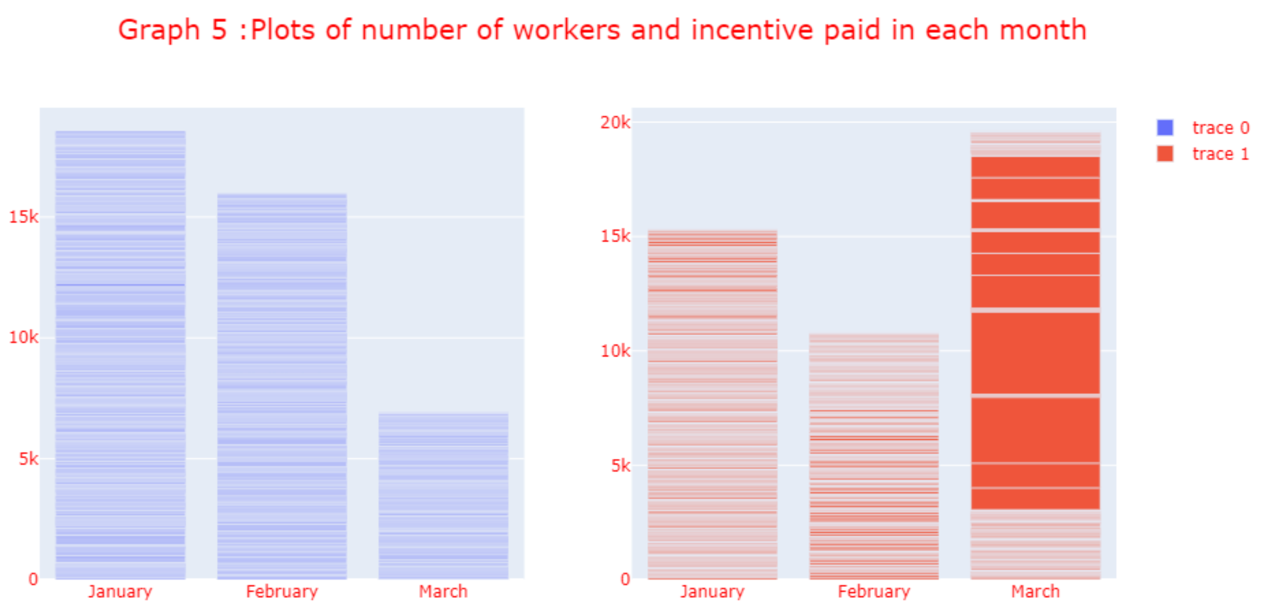




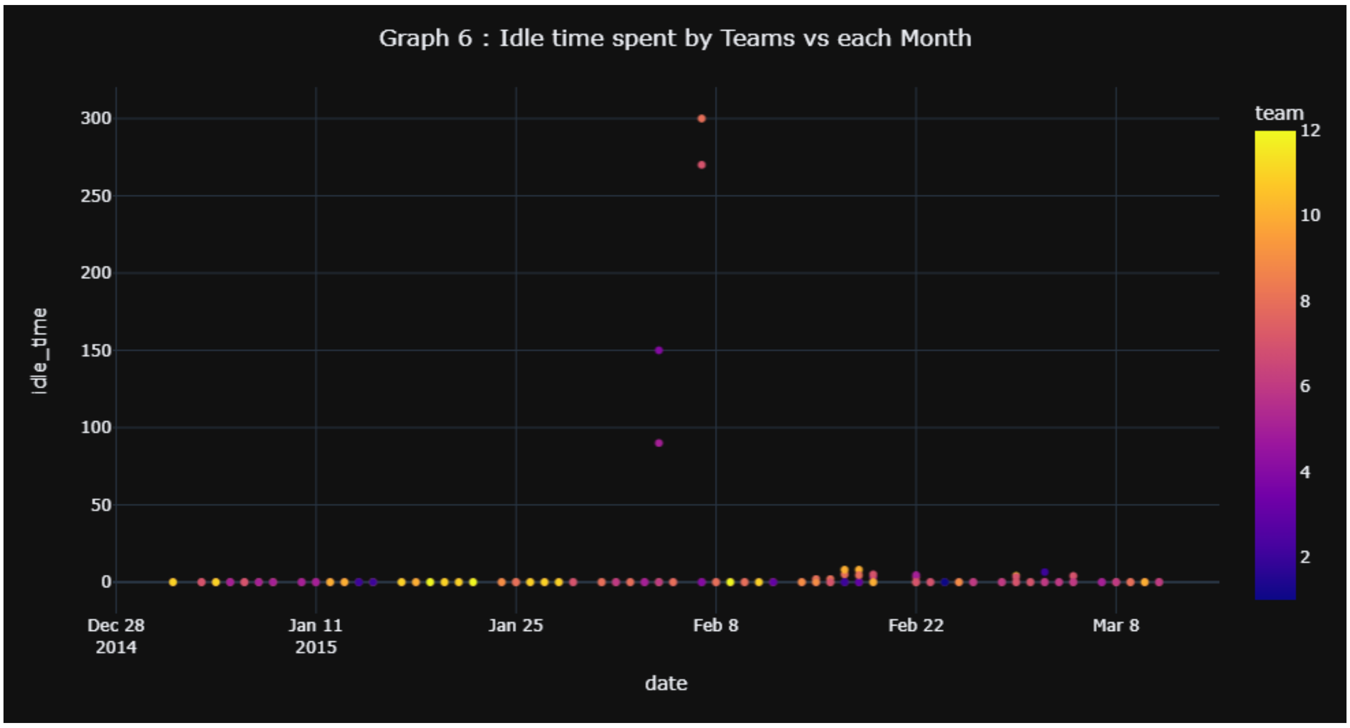
- No data for plant activity on Friday means, plant activities are stopped on Friday and they have considered it as holiday.



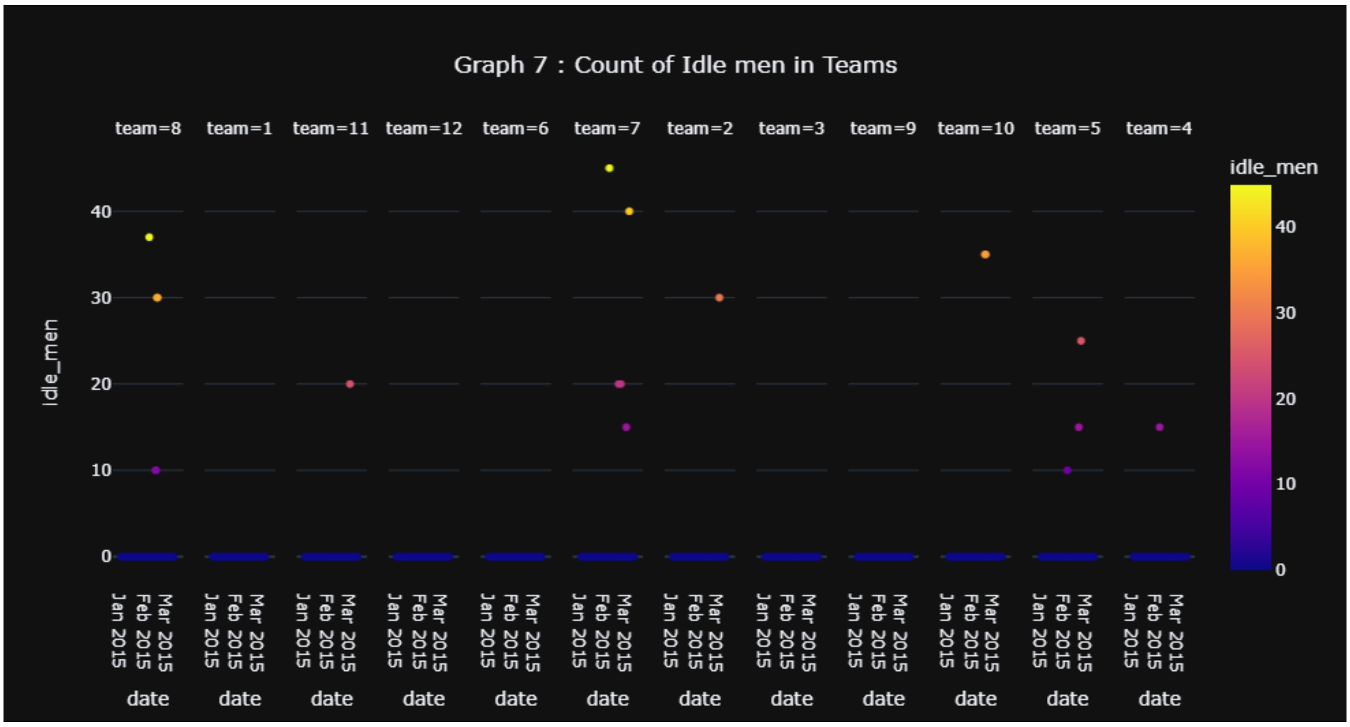
- Above graph suggest that; Saturday is the most productive weekday with an average of 0.75, it maybe because Friday is holiday which makes sense, as generally after a break our productivity increases



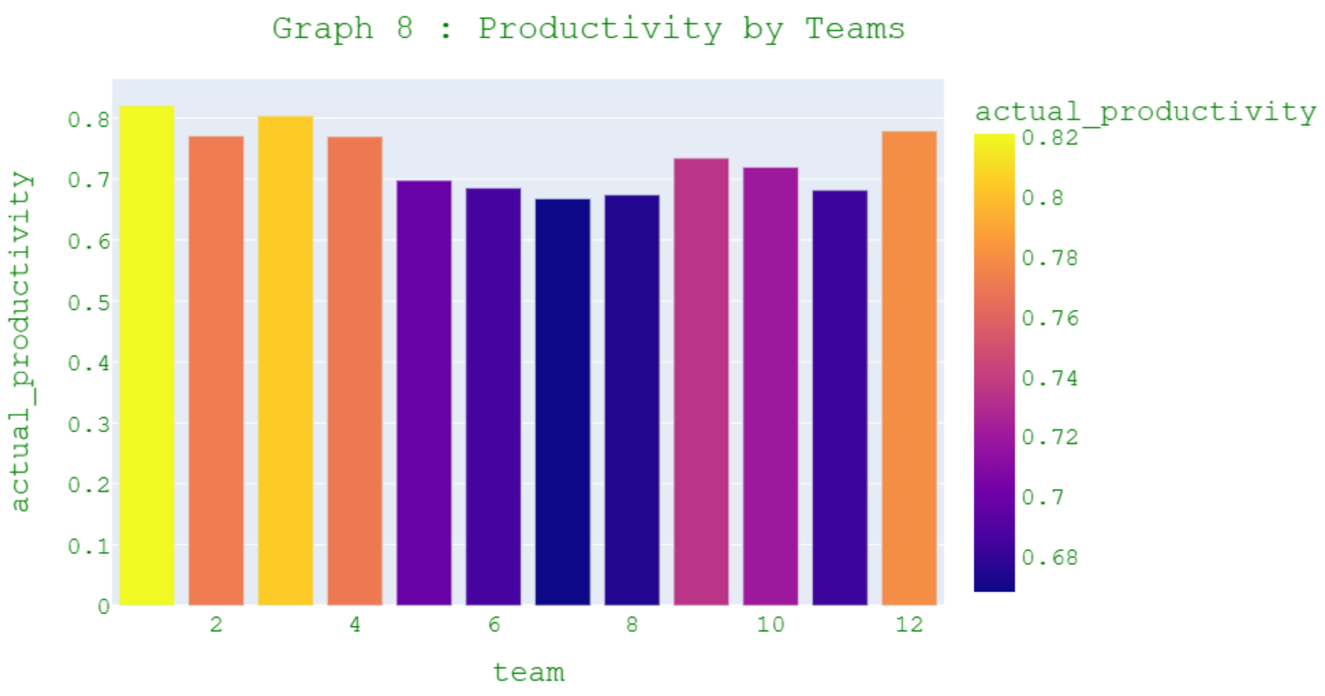
➢ The above graph using plotly library suggest that no of employees/worker worked in march are less but still high incentives was paid, meaning workers are working overtime.



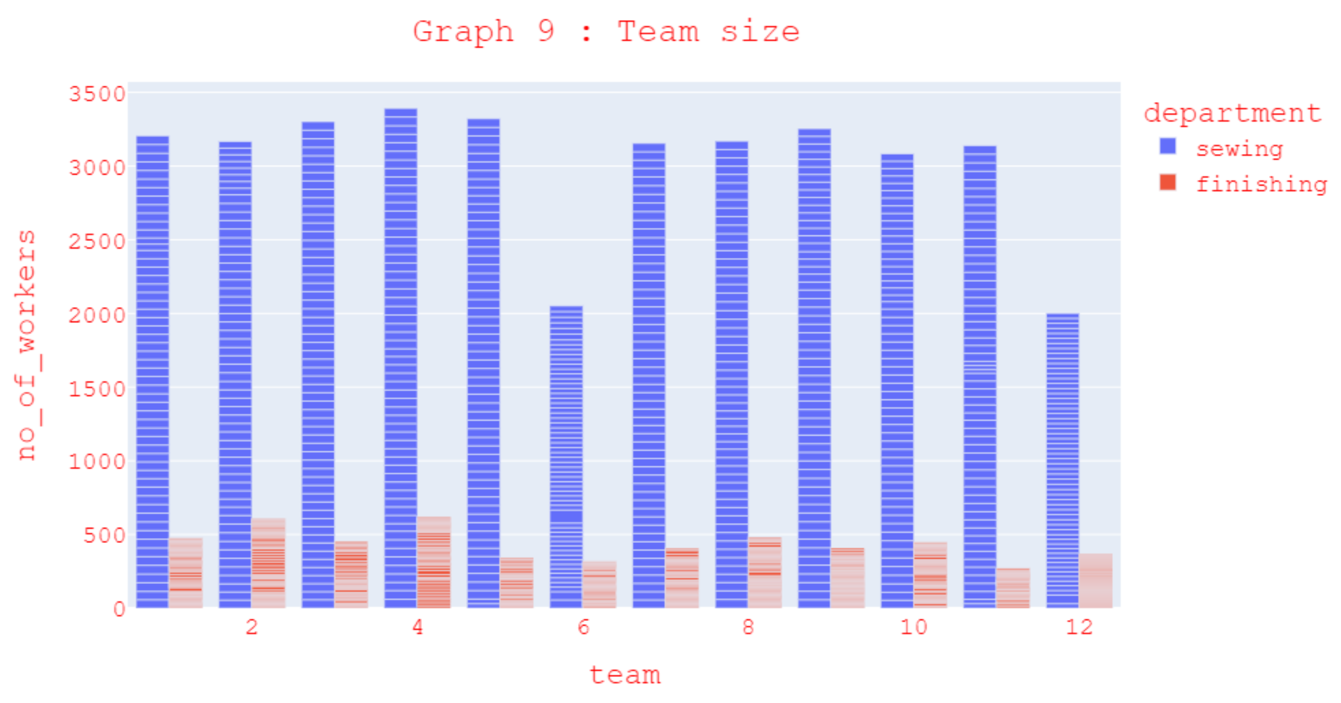
➢ Above scatter plot Suggest that Team 8 and Team 7 spent most Idle Time.



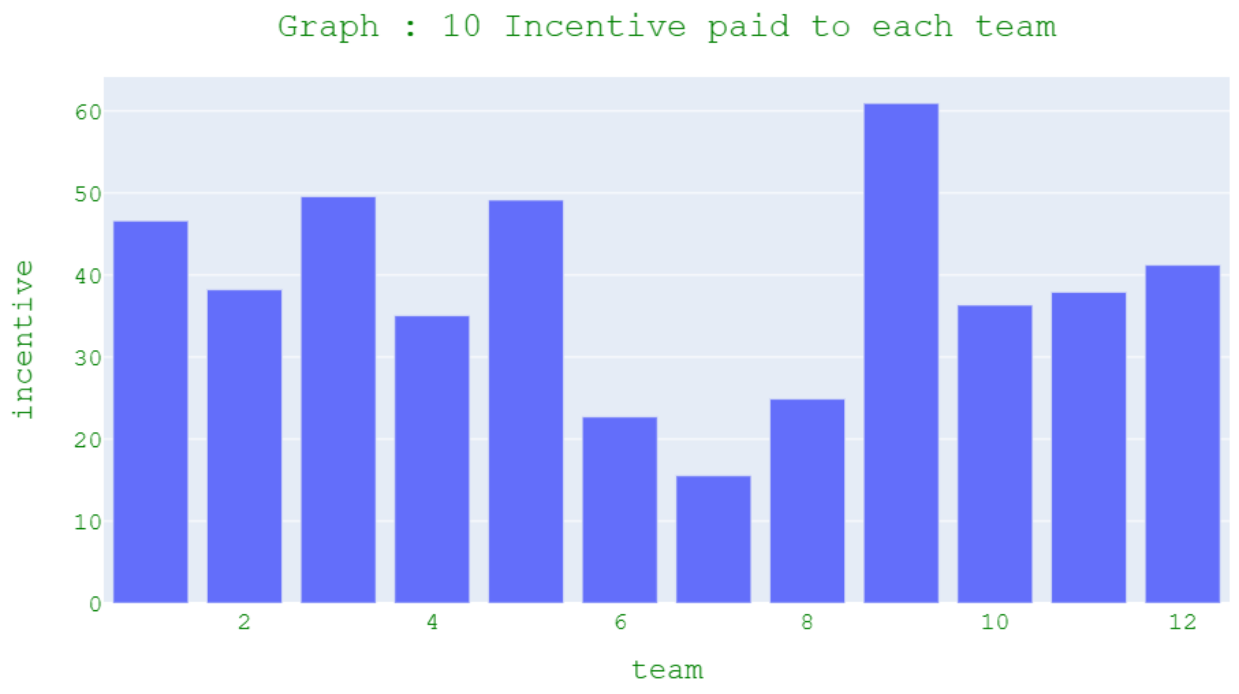
➢ No of idle men for team 8 and team 7 are found to be 45 and 37 respectively in the month of February.



➢ Team 1 is the most productive followed by Team 3

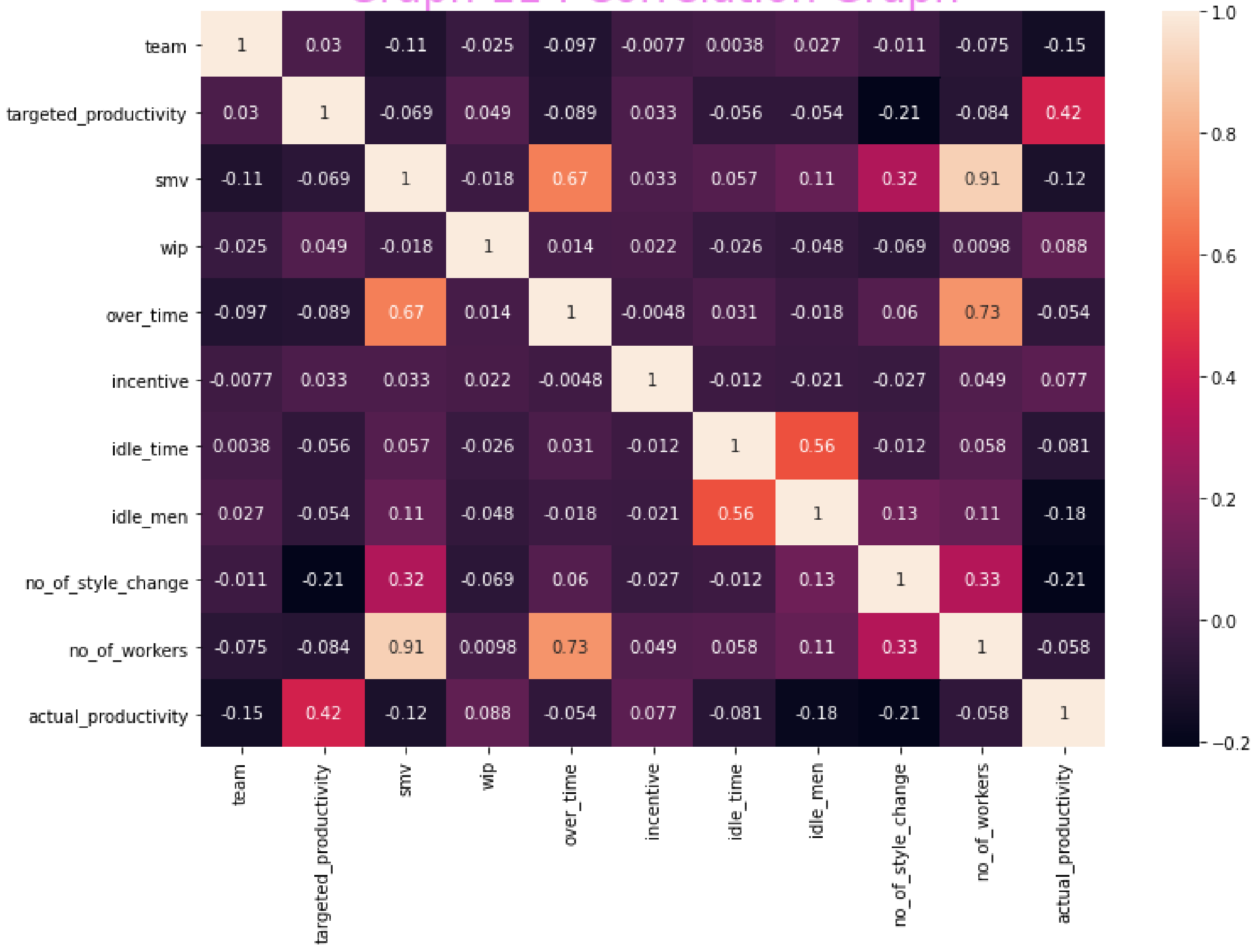


➢ No. of workers required in each team for sewing department is comparatively high



➢ Above graph suggest that Team 9 gets the highest incentive on an average.

Graph 11 : Correlation Graph



➢ Above Correlation Graph shows that their is very High Collinearity i.e 0.91 between no of workers and **SMV** (Standard Minute Value, it is the allocated time for a task)

- Highest Positive Correlations (Dsecending order).

1. No_of workers and smv (0.91)
2. No_of workers and over_time (0.73)
3. Over_time and smv (0.67)
4. Idle_men and Idle_time (0.56)
5. No_of workers and no_of_style_change(0.33)
6. No_of_style_changehas and smv (0.32)

# III. Machine Learning Model

## ◆ Linear Regression using Python.

### ➢ Regression

→ Regression analysis is one of the most important fields in statistics and machine learning. There are many regression methods available. Linear regression is one of them.

→ Regression searches for relationships among variables.

→ Generally, in regression analysis, we usually consider some phenomenon of interest and have a number of observations. Each observation has two or more features. Following the assumption that (at least) one of the features depends on the others, you try to establish a relation among them.

- The dependent features are called the **dependent variables**, **outputs**, or **responses**.
- The independent features are called the **independent variables**, **inputs**, or **predictors**.

→ Regression problems usually have one continuous and unbounded dependent variable. The inputs, however, can be continuous, discrete, or even categorical data such as gender, nationality, brand, and so on.

→ It is a common practice to denote the outputs with $y$ and inputs with $x$. If there are two or more independent variables, they can be represented as the vector $\mathbf{x} = (x_1, \ldots, x_r)$, where $r$ is the number of inputs.

### ➢ Linear Regression

Linear regression is probably one of the most important and widely used regression techniques. It's among the simplest regression methods. One of its main advantages is the ease of interpreting results.

**Uses** : It is a very powerful technique and can be used to understand the factors that influence profitability. It can be used to forecast sales in the coming months by analyzing the sales data for previous months. It can also be used to gain various insights about customer behaviour.

#### ➢ Problem Formulation

→ When implementing linear regression of some dependent variable $y$ on the set of independent variables $\mathbf{x} = (x_1, \ldots, x_r)$, where $r$ is the number of predictors, you assume a linear relationship between $y$ and $\mathbf{x}$: $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_r x_r + \varepsilon$. This equation is the **regression equation**. $\beta_0, \beta_1, \ldots, \beta_r$ are the **regression coefficients**, and $\varepsilon$ is the **random error**.

→ Linear regression calculates the **estimators** of the regression coefficients or simply the **predicted weights**, denoted with $b_0, b_1, \ldots, b_r$. They define the **estimated regression function** $f(\mathbf{x}) = b_0 + b_1 x_1 + \cdots + b_r x_r$. This function should capture the dependencies between the inputs and output sufficiently well.

→ The **estimated** or **predicted response**, $f(\mathbf{x}_i)$, for each observation $i = 1, \ldots, n$, should be as close as possible to the corresponding **actual response** $y_i$. The differences $y_i - f(\mathbf{x}_i)$ for all observations $i = 1, \ldots, n$, are called the **residuals**. Regression is about determining the **best predicted weights**, that is the weights corresponding to the smallest residuals.

→ To get the best weights, you usually **minimize the sum of squared residuals** (SSR) for all observations $i = 1, \ldots, n$: $\text{SSR} = \Sigma_i (y_i - f(\mathbf{x}_i))^2$. This approach is called the **method of ordinary least squares**.

## ◆ Data Pre-Processing

### ➢ One-hot Encoding.

One Hot Encoding is a process in the data processing that is applied to categorical data, to convert it into **a binary vector representation** for use in machine learning algorithms.

### ➢ Why and When shall we use One-Hot Encoding.

One of the major problems with machine learning is that a lot of algorithms cannot work directly with categorical data. Categorical data [1] are variables that can take on one of a limited number of possible values.

Some examples are:

- The sex of a person: female or male.
- The airline travel class: First Class, Business Class, and Economy Class.
- The computer vendor: Lenovo, HP, Dell, Apple, Acer, Asus, and Others.

Therefore, we need a way **to convert categorical data into a numerical form** and our machine learning algorithm can take in that as input.
The most widely used encoding techniques are:

1. **Integer Encoding** : encodes the values as integer.

→ Some categories may have a natural relationship (known as a natural ordering) to each other.

For e.g. - The **airline travel class** variable does have a natural ordering of values. This type of categorical variable is known as an **ordinal variable**. **Ordinal variables** should be treated differently in machine learning because the ordinality generally comes with some significance.

In the case of **airline travel class**, the passengers in the higher classes tend to be wealthier and older. The integer values have a natural ordered relationship between each other and machine learning algorithms may be able to learn this relationship.

Example :

| | **Travel Class** |
|---|---|
| **Passenger 1** | 1 |
| **Passenger 2** | 2 |
| **Passenger 3** | 3 |
| **Passenger 4** | 1 |

2. **One-Hot Encoding** : encodes the values as a binary vector array.

→ For categories variables where no such ordinal relationship exists, **Integer Encoding** could result in poor performance or unexpected results.

→ **One-Hot Encoding** is a way to combat this. **One-Hot Encoding** simply creates one column for every possible value and put a 1 in the appropriate column.

| | Travel Class 1 | Travel Class 2 | Travel Class 3 |
|---|---|---|---|
| **Passenger 1** | 1 | 0 | 0 |
| **Passenger 2** | 0 | 1 | 0 |
| **Passenger 3** | 0 | 0 | 1 |
| **Passenger 4** | 1 | 0 | 0 |

→ With **One-Hot Encoding**, the binary vector arrays representation allows a machine learning algorithm to leverage the information contained in a category value **without the confusion caused by ordinality**.

3. **Dummy Variable Encoding** : same as One-Hot Encoding, but one less column.

→ There is some problem in **One-Hot encoding**.

→ For instance, in the above example, if we know that a passenger's flight ticket is not *First Class* and not *Economy Class*, then it must be *Business Class*.
So, we only need to use two of these three dummy-coded variables as a predictor. More generally, the number of dummy-coded variables needed is one less than the number of possible values, which is K-1. In statistics, this is called a **dummy encoding variable**, or **dummy variable**.

| | Travel Class 2 | Travel Class 3 |
|---|---|---|
| **Passenger 1** | 0 | 0 |
| **Passenger 2** | 1 | 0 |
| **Passenger 3** | 0 | 1 |
| **Passenger 4** | 0 | 0 |

→ **Dummy encoding variable** is a standard advice in statistics to avoid the **dummy variable trap**, However, in the world of machine learning, **One-Hot encoding** is more recommended because **dummy variable trap** is not really a problem when applying regularization.

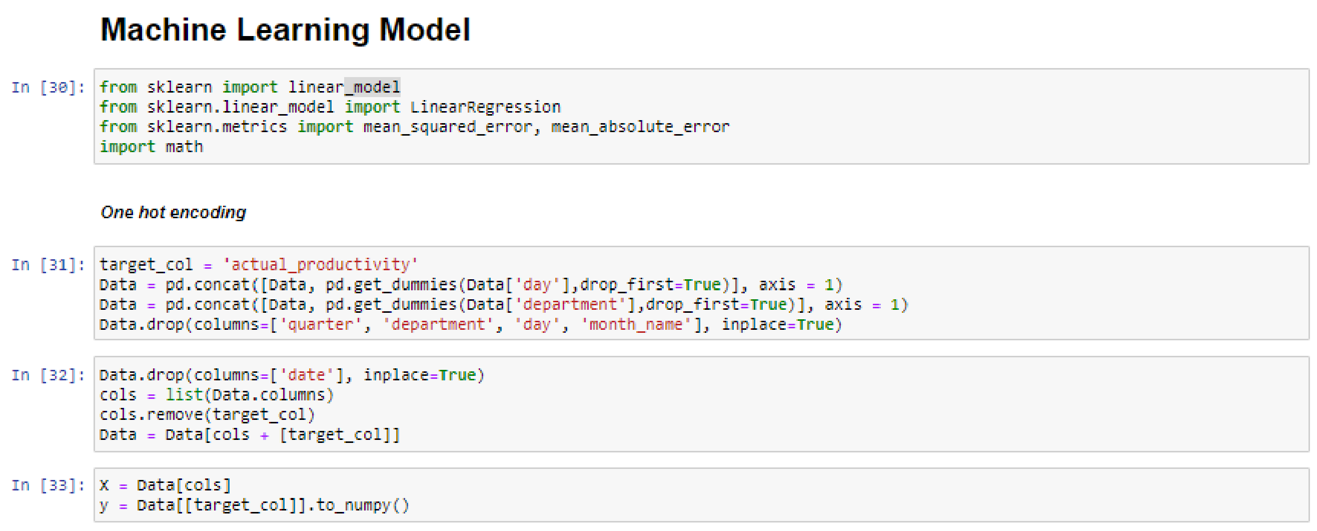
◆ **Creating a One-Hot encoding variable.**

**Syntax :** `pd.get_dummies(df.category)`

◆ **Creating a dummy encoding variable**

**Syntax :** `pd.get_dummies(df.Category, drop_first=True)`

## CODE : For deploying ML Model



### Machine Learning Model

```
In [30]: from sklearn import linear_model
         from sklearn.linear_model import LinearRegression
         from sklearn.metrics import mean_squared_error, mean_absolute_error
         import math
```

*One hot encoding*

```
In [31]: target_col = 'actual_productivity'
         Data = pd.concat([Data, pd.get_dummies(Data['day'],drop_first=True)], axis = 1)
         Data = pd.concat([Data, pd.get_dummies(Data['department'],drop_first=True)], axis = 1)
         Data.drop(columns=['quarter', 'department', 'day', 'month_name'], inplace=True)
```

```
In [32]: Data.drop(columns=['date'], inplace=True)
         cols = list(Data.columns)
         cols.remove(target_col)
         Data = Data[cols + [target_col]]
```

```
In [33]: X = Data[cols]
         y = Data[[target_col]].to_numpy()
```

- **Feature Selection and Checking for their Importance.**

When building a machine learning model in real-life, it's almost rare that all the variables in the dataset are useful to build a model. Adding redundant variables reduces the generalization capability of the model and may also reduce the overall accuracy of a classifier. Furthermore adding more and more variables to a model increases the overall complexity of the model.

**GOAL** : The goal of feature selection in machine learning is to find the best set of features that allows one to build useful models of studied phenomena.
The techniques for feature selection in machine learning can be broadly classified into the following categories:

- **Supervised Techniques**: These techniques can be used for labeled data, and are used to identify the relevant features for increasing the efficiency of supervised models like classification and regression.
- **Unsupervised Techniques**: These techniques can be used for unlabeled data. From a taxonomic point of view, these techniques are classified as under:
  A. Filter methods
  B. Wrapper methods
  C. Embedded methods
  D. Hybrid methods

**Feature Importance**
Feature importance refers to a class of techniques for assigning scores to input features to a predictive model that indicates the relative importance of each feature when making a prediction.
Feature importance scores can be calculated for problems that involve predicting a numerical value, called regression, and those problems that involve predicting a class label, called classification.
The scores are useful and can be used in a range of situations in a predictive modelling problem, such as:

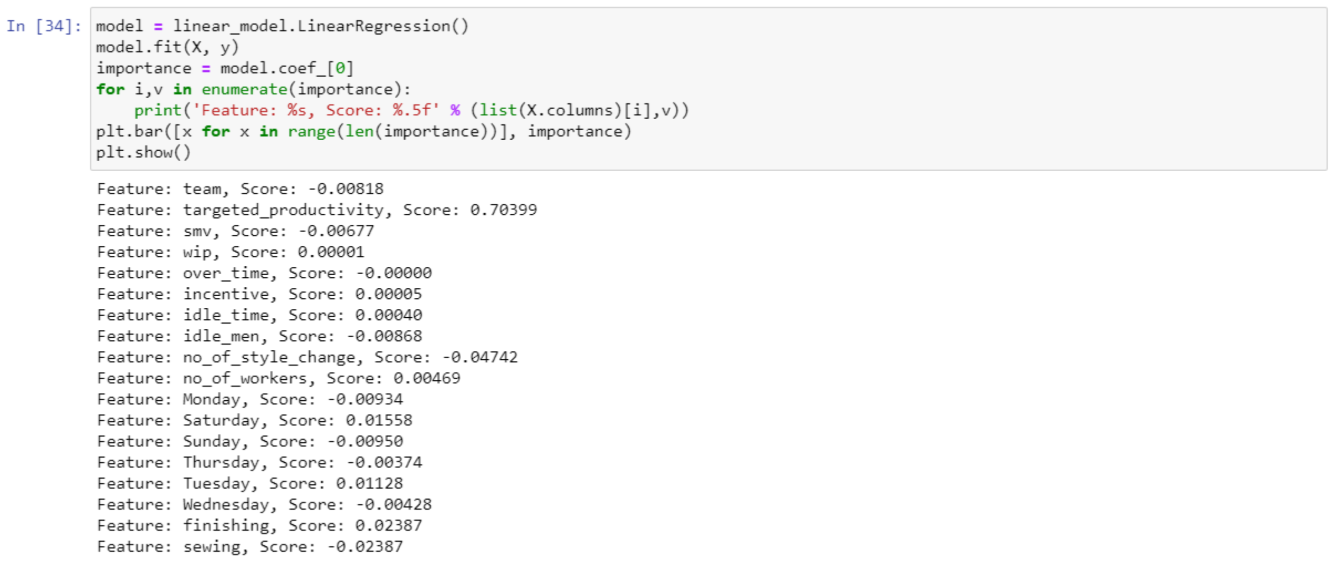
- Better understanding the data.
- Better understanding a model.
- Reducing the number of input features.

**Feature importance scores can provide insight into the dataset**. The relative scores can highlight which features may be most relevant to the target, and the converse, which features are the least relevant. This may be interpreted by a domain expert and could be used as the basis for gathering more or different data.

**Feature importance scores can provide insight into the model**. Most importance scores are calculated by a predictive model that has been fit on the dataset. Inspecting the importance score provides insight into that specific model and which features are the most important and least important to the model when making a prediction. This is a type of model interpretation that can be performed for those models that support it.

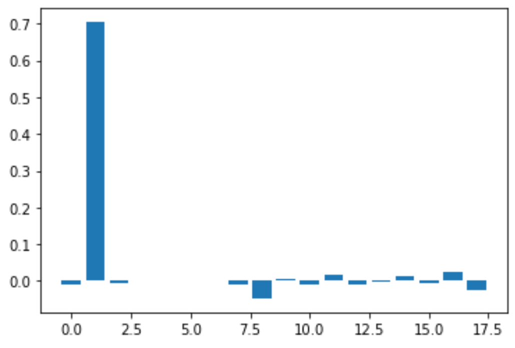
**Feature importance can be used to improve a predictive model**. This can be achieved by using the importance scores to select those features to delete (lowest scores) or those features to keep (highest scores). This is a type of feature selection and can simplify the problem that is being modelled, speed up the modelling process (deleting features is called dimensionality reduction), and in some cases, improve the performance of the model.

## Python Code for Our Model to check important features



```
In [34]: model = linear_model.LinearRegression()
         model.fit(X, y)
         importance = model.coef_[0]
         for i,v in enumerate(importance):
             print('Feature: %s, Score: %.5f' % (list(X.columns)[i],v))
         plt.bar([x for x in range(len(importance))], importance)
         plt.show()
```

```
Feature: team, Score: -0.00818
Feature: targeted_productivity, Score: 0.70399
Feature: smv, Score: -0.00677
Feature: wip, Score: 0.00001
Feature: over_time, Score: -0.00000
Feature: incentive, Score: 0.00005
Feature: idle_time, Score: 0.00040
Feature: idle_men, Score: -0.00868
Feature: no_of_style_change, Score: -0.04742
Feature: no_of_workers, Score: 0.00469
Feature: Monday, Score: -0.00934
Feature: Saturday, Score: 0.01558
Feature: Sunday, Score: -0.00950
Feature: Thursday, Score: -0.00374
Feature: Tuesday, Score: 0.01128
Feature: Wednesday, Score: -0.00428
Feature: finishing, Score: 0.02387
Feature: sewing, Score: -0.02387
```



**Conclusion** : From the above Output we find that the main or important feature are :

- Targeted Productivity
- Days : Saturday and Tuesday
- Department : Finishing

◆ **Data-Partitioning**

→ Data partitioning in data mining is the division of the whole data available into two or three non-overlapping sets: the training set , the validation set , and the test set .

→ The basic idea of data partitioning is to keep a subset of available data out of analysis, and to use it later for verification of the model.

→ Data partitioning is normally used in supervised learning techniques in data mining where a predictive model is chosen from a set of models, using their performance on the training set as the validation of choice. Some examples of such techniques are classification trees , regression trees , neural networks , nonlinear variants of the discriminant analysis .

◆ **Evaluation Matrices for Linear Regression**

1. **$R^2$ (R – Squared)**

This metric represents the part of the variance of the dependent variable explained by the independent variables of the model. It measures the strength of the relationship between your model and the dependent variable.

If the data points are very close to the regression line, then the model accounts for a good amount of variance, thus resulting in a high $R^2$ value.

$R^2 = 1 - \frac{RSS}{TSS}$ **---> represents the part of the variance of y, described by the independent variables**

→ If **$R^2$** is high (say 1), then the model represents the variance of the dependent variable.

→ If **$R^2$** is very low, then the model does not represent the variance of the dependent variable and regression is no better than taking the mean value, i.e. you are not using any information from the other variables.

→ *A Negative $R^2$ means you are doing worse than the mean value. It can have a negative value if the predictors do not explain the dependent variables at all such that RSS ~ TSS.*

→ Thus, $R^2$ evaluates the scattered data points about the regression line.

To summarize, the ratio of the **residual error** (RSS) against the **total error** (TSS) tells us how much of the total error remains in our regression model.
Subtracting that ratio from 1 gives how much error we removed using the regression analysis. That is the R-squared error.

2. **Adjusted $R^2$**

→ The main difference between **adjusted R-squared** and R-square is that **R-squared** describes the amount of variance of the dependent variable represented by every single independent variable, while **adjusted R-squared** measures variation explained by only the independent variables that actually affect the dependent variable.

$$R^2_{adjusted} = [\frac{(1-R^2)(n-1)}{n-k-1}]$$

→ In the equation above, n is the number of data points while k is the number of variables in your model, excluding the constant.

→ $R^2$ tends to increase with an increase in the number of independent variables. This could be misleading. Thus, the adjusted R-squared penalizes the model for adding furthermore independent variables (k in the equation) that do not fit the model.

3. **MSE (Mean Square error)**
   The most common metric for regression tasks is MSE. It has a convex shape. It is the average of the squared difference between the predicted and actual value. Since it is differentiable and has a convex shape, it is easier to optimize.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

   MSE penalizes large errors.

4. **MAE (Mean Absolute error)**
   This is simply the average of the absolute difference between the target value and the value predicted by the model. Not preferred in cases where outliers are prominent.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

   MAE does not penalize large errors.

5. **RMSE**

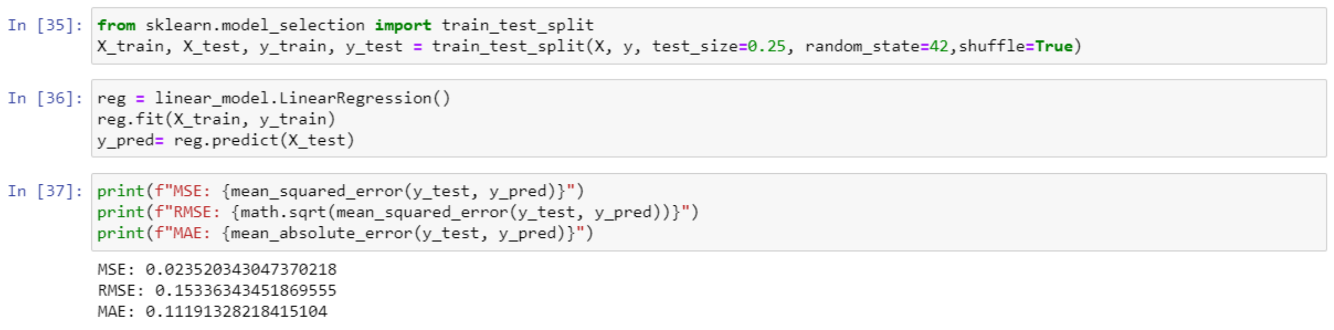
→ This is the square root of the average of the squared difference of the predicted and actual value.

→ R-squared error is better than RMSE. This is because R-squared is a relative measure while RMSE is an absolute measure of fit (highly dependent on the variables — not a normalized measure)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

→ RMSE penalizes large errors.

◆ **Python code : Implementation**



```
In [35]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42,shuffle=True)

In [36]: reg = linear_model.LinearRegression()
         reg.fit(X_train, y_train)
         y_pred= reg.predict(X_test)

In [37]: print(f"MSE: {mean_squared_error(y_test, y_pred)}")
         print(f"RMSE: {math.sqrt(mean_squared_error(y_test, y_pred))}")
         print(f"MAE: {mean_absolute_error(y_test, y_pred)}")

         MSE: 0.023520343047370218
         RMSE: 0.15336343451869555
         MAE: 0.11191328218415104
```

## CONCLUSION

1. Using different visualization libraries in Python – We have got different sort of insights about the data such as:

    - In month of February Actual productivity found to be high as compared to target productivity suggesting overperformance, but in month of march we observed that actual productivity was found lower as compared to set target productivity.
    - There was no data for plant activity on Friday means, plant activities are stopped on Friday and they have considered it as holiday.
    - Through productivity vs weekday Graph, we found that, Saturday is the most productive weekday with an average of 0.75, it maybe because Friday is holiday which makes sense, as generally after a break our productivity increases
    - We plotted the graph between no of workers and incentive paid in each month using plotly library suggest that no of employees/worker worked in march are less but still high incentives was paid, meaning workers are working overtime.
    - Scatter plot for Idle time spent by teams vs each month, suggest that Team 8 and Team 7 spent most Idle Time.
    - Using plotly library we also found that No of idle men for team 8 and team 7 are found to be 45 and 37 respectively in the month of February.
    - Graph 9 shows us the actual Productivity by different teams, and we found that Team 1 is the most productive followed by Team 3.
    - Also we found out that, No. of workers required in each team for sewing department is comparatively high.
    - By plotting graph between incentive paid to different teams, found that Team 9 gets the highest incentive on an average.

2. We applied linear regression on the given problem and obtained the MAE value 0.11191, which will help Garment manufacturer to set the Accurate production Targets with the present infrastructure.