FINAL REPORT

GARMENT WORKER PRODUCTIVITY PREDICTION

TEAM:

SHREYAS REDDY GUVVALA

NIKHIL PULAGAM

MULE VIGNESH

1. Data Collection and Preprocessing Phase

3.2Data Collection Plan and Raw Data Sources Identified

There are many popular open sources for collecting the data. E.g., kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset. Download the dataset from the above link. Let us understand and analyze the dataset properly using visualization techniques.

Note: There are several approaches for understanding the data. But we have applied some of it here. You can also employ a variety of techniques.

Activity 1.1: Importing the libraries

Import the libraries required for this machine learning project, as shown in the image below.

```
# Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, mean_squared_log_error
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.ensemble import StackingRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import AdaBoostRegressor
```

Activity 1.2: Read the Dataset

Our dataset format could be in .csv, excel files,.txt, .json, and so on. With the help of pandas, we can read the dataset.

Since our dataset is a csv file, we use read_csv() which is pandas function to read the dataset. As a parameter we have to give the directory of the csv file.
df.head() will display first 5 rows of the dataset.

```
df = pd.read_csv('garments_worker_productivity.csv')
```

```
df.head()
```

Python

quarter	department	day	team	targeted_productivity	smv	wip	over_time	incentive	idle_time	idle_men	no_of_style_change	no_of_workers	actual_productivity
Quarter1	sweing	Thursday	8	0.80	26.16	1108.0	7080	98	0.0	0	0	59.0	0.940725
Quarter1	finishing	Thursday	1	0.75	3.94	NaN	960	0	0.0	0	0	8.0	0.886500
Quarter1	sweing	Thursday	11	0.80	11.41	968.0	3660	50	0.0	0	0	30.5	0.800570
Quarter1	sweing	Thursday	12	0.80	11.41	968.0	3660	50	0.0	0	0	30.5	0.800570
Quarter1	sweing	Thursday	6	0.80	25.90	1170.0	1920	50	0.0	0	0	56.0	0.800382

a. Data Exploration and Preprocessing

Data preparation, also known as data preprocessing, is the process of cleaning, transforming, and organizing raw data before it can be used in a data analysis or machine learning model.

The activity include following steps:

- removing missing values
- handling outliers

- encoding categorical variables
- normalizing data

    Note: These are general steps to take in pre-processing before feeding data to machine learning for training. The pre-processing steps differ depending on the dataset. Depending on the condition of your dataset, you may or may not have to go through all these steps.

    Activity 2.1: Handling Missing Values

    Let's first figure out what kind of data is in our columns by using df.info(). We may deduce from this that our columns include data of the types "object", "float64" and “int64”.

```
df.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
```

```
df.isna().sum()
```

```
quarter                  0
department               0
day                      0
team_number              0
time_allocated           0
unfinished_items       506
over_time                0
incentive                0
idle_time                0
idle_men                 0
style_change             0
no_of_workers            0
actual_productivity      0
dtype: int64
```

This line of code is used to count the number of missing or null values in a pandas DataFrame. It returns a list of the total number of missing values in each column of the DataFrame.

```
df['unfinished_items'] = df['unfinished_items'].fillna(df['unfinished_items'].mean())
```

This line of code fills the missing values in the “unfinished_items” column with the column mean.

Activity 2.2: Handling Independent Columns

```
df.drop(['date', 'targeted_productivity'], axis=1, inplace=True)
df.head()
```

	quarter	department	day	team	smv	wip	over_time	incentive	idle_time	idle_men	no_of_style_change	no_of_workers	actual_productivity
0	Quarter1	sweing	Thursday	8	26.16	1108.0	7080	98	0.0	0	0	59.0	0.940725
1	Quarter1	finishing	Thursday	1	3.94	NaN	960	0	0.0	0	0	8.0	0.886500
2	Quarter1	sweing	Thursday	11	11.41	968.0	3660	50	0.0	0	0	30.5	0.800570
3	Quarter1	sweing	Thursday	12	11.41	968.0	3660	50	0.0	0	0	30.5	0.800570
4	Quarter1	sweing	Thursday	6	25.90	1170.0	1920	50	0.0	0	0	56.0	0.800382

+ Code + Markdown

This line of code drops the columns date and targeted_productivity from the dataframe.

```
df = df.rename(columns={
    'team' : 'team_number',
    'smv' : 'time_allocated',
    'wip' : 'unfinished_items',
    'no_of_style_change' : 'style_change'
})
df
```

The rename() method is being used to change the names of some columns. The method takes a dictionary as its argument, where the keys are the original column names and the values are the new names.

```
df['quarter'].unique()
```

```
array(['Quarter1', 'Quarter2', 'Quarter3', 'Quarter4', 'Quarter5'],
      dtype=object)
```

```
df['quarter'] = df['quarter'].str.replace('Quarter5','Quarter1')
```

The first line of code selects a specific column named "quarter" from a table of data. The unique()method is then used to list all the different values in that column.
The second line of code changes some of the values in the "quarter" column. Specifically, it replaces any instances of the text "Quarter5" with the text "Quarter1" using the str.replace()method.

```
# Extract the numerical part from the 'quarter' column
df['quarter'] = df['quarter'].str.extract(r'(\d+)')
df['quarter']
```

The first line of code is using the str.extract() method on the "quarter"

column to extract the numerical part of each value. It uses a regular expression pattern r'(\d+)' to match any sequence of one or more digits in each value.
The second line of code is assigning the modified "quarter" column back to the same "quarter" column, effectively replacing the original column with the modified one.

```
df['department'] = df['department'].str.replace('sweing','sewing')
df['department'] = df['department'].str.replace('finishing ','finishing')
```

The first line of code is using the str.replace() method on the "department" column to replace any instances of the misspelled word "sweing" with the correct spelling "sewing".
The second line of code is also using the str.replace() method on the "department" column to replace any instances of the word "finishing " (with a space at the end) with the word "finishing" (without a space at the end). This will remove the extra space and standardize the spelling of the word.

```
df['team_number'] = df['team_number'].astype(int)
df['over_time'] = df['over_time'].astype(int)
df['incentive'] = df['incentive'].astype(int)
df['idle_time'] = df['idle_time'].astype(int)
df['idle_men'] = df['idle_men'].astype(int)
df['style_change'] = df['style_change'].astype(int)
```

```
df['time_allocated'] = df['time_allocated'].astype(int)
```

```
df['unfinished_items'] = df['unfinished_items'].astype(int)
```

```
df['idle_time'] = df['idle_time'].astype(int)
```

```
df['no_of_workers'] = df['no_of_workers'].astype(int)
```

Each line of code is using the astype() method to convert the data type

of a specific column to an integer type. The column name is specified on the left-hand side of the equation, and the new integer type is specified as an argument to the astype() method.
Activity 2.3: Handling Categorical Values

```
lc = LabelEncoder()
```

```
print('Before encoding: ', df['department'].unique())
df['department'] = lc.fit_transform(df['department'])
print('After encoding: ',df['department'].unique())
```

This code is encoding the values in a column named "department" by using a LabelEncoder() object to convert the original values into numerical encoded values. The original and encoded values are printed before and after the encoding is performed.

```
Before encoding:  ['sewing' 'finishing']
After encoding:  [1 0]
```

```
print('Before encoding: ', df['day'].unique())
df['day'] = lc.fit_transform(df['day'])
print('After encoding: ',df['day'].unique())
```

```
Before encoding:  ['Thursday' 'Saturday' 'Sunday' 'Monday' 'Tuesday' 'Wednesday']
After encoding:  [3 1 2 0 4 5]
```

This code is encoding the values in a column named "day" by using a LabelEncoder() object to convert the original values into numerical encoded values. The original and encoded values are printed before and

after the encoding is performed.

3.3 Data Exploration and Preprocessing

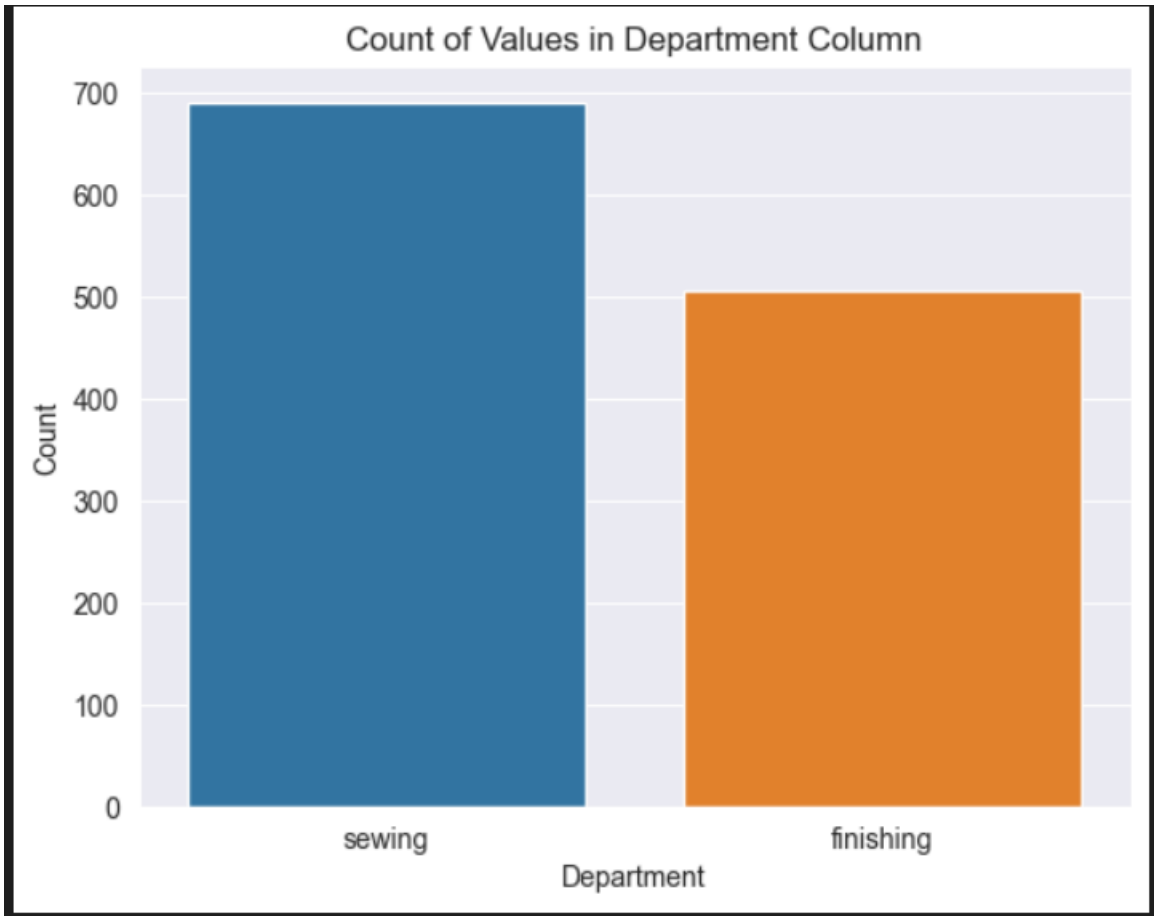
Visual analysis is the process of examining and understanding data via the use of visual representations such as charts, plots, and graphs. It is a method for quickly identifying patterns, trends, and outliers in data, which can aid in gaining insights and making sound decisions.

Activity 2.1: Univariate analysis

Univariate analysis is a statistical method used to analyse a single variable in a dataset. This analysis focuses on understanding the distribution, central tendency, and dispersion of a single variable.

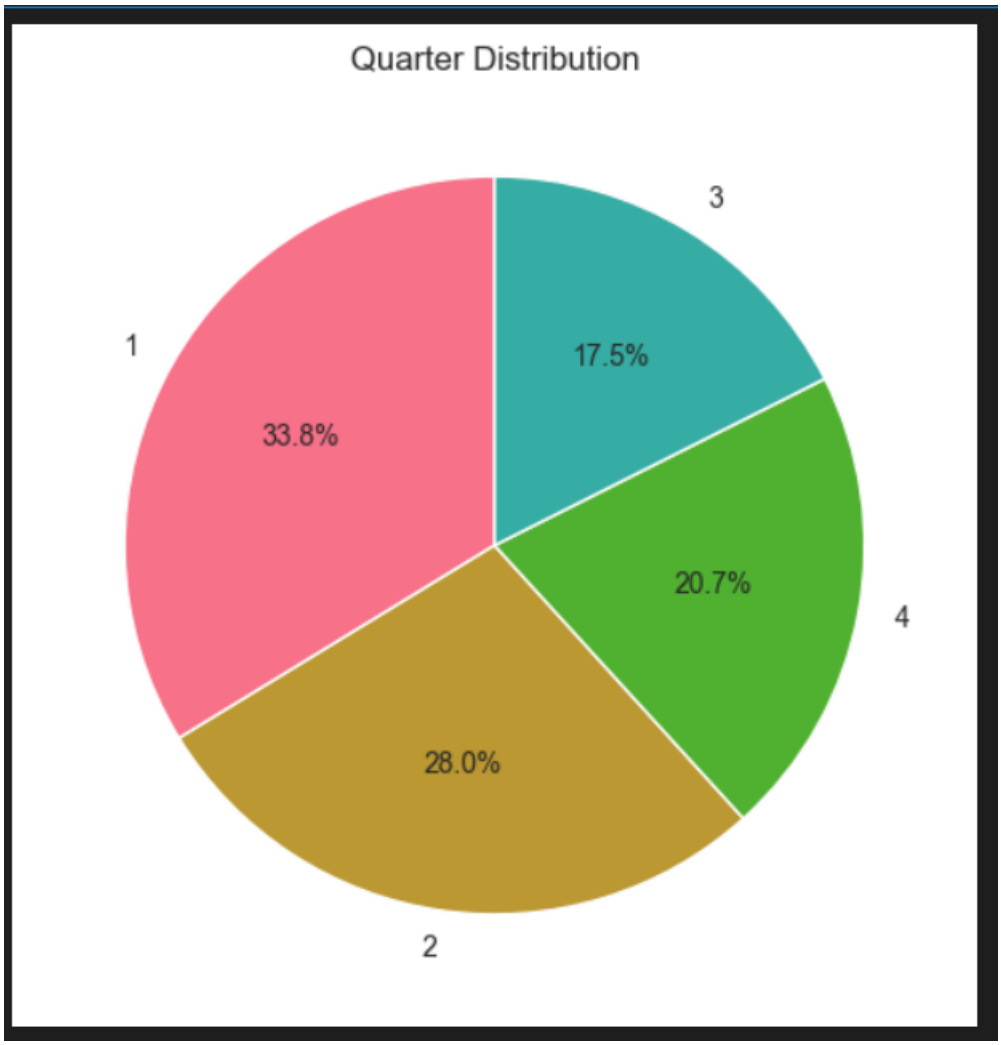
```
sns.countplot(data=df, x='department')
plt.xlabel('Department')
plt.ylabel('Count')
plt.title('Count of Values in Department Column')
plt.show()
```

This code creates a bar chart using the Seaborn library to show the number of occurrences of each unique value in the "department" column of a Pandas DataFrame. The x-axis represents the unique values of the "department" column, and the y-axis represents the number of times each unique value appears in the column. The plt.xlabel(), plt.ylabel(), and plt.title() functions are used to add labels and a title to the plot. Finally, plt.show() is used to display the plot.




```
quarter_counts = df['quarter'].value_counts()
plt.figure(figsize=(8, 6))
sns.set_palette("husl")  # Set custom color palette
plt.pie(quarter_counts, labels=quarter_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Quarter Distribution')
plt.show()
```

This code generates a pie chart using the Seaborn library to show the distribution of different quarters in a dataset. The value_counts() function counts the number of occurrences of each quarter in the dataset, and the resulting counts are used to create the pie chart. The pie chart shows the proportion of the total number of entries in the dataset that corresponds to each quarter. The title of the pie chart is "Quarter Distribution".



Activity 2.2: Bivariate analysis

Bivariate analysis is a statistical method used to analyse the relationship between two variables in a dataset. This analysis focuses on examining how changes in one variable are related to changes in another variable.

```
sns.lineplot(data=df, x='team_number', y='unfinished_items')
plt.xlabel('Team Number')
plt.ylabel('Unfinished Items')
plt.title('Line Plot of Unfinished Items by Team Number')
plt.show()
```

This code generates a line plot using the Seaborn library to show the relationship between team number and the number of unfinished items.

The line plot shows how the number of unfinished items varies across different teams. The x-axis represents the team number, and the y-axis represents the number of unfinished items. The title of the line plot is "Line Plot of Unfinished Items by Team Number".