Data Collection and Preprocessing Phase

Date	4 July 2024
Team ID	SWTID1720090524
Project Title	Garment Worker Productivity Prediction
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis. Dataset variables will be statistically analyzed to identify patterns and outliers, with Python employed for preprocessing tasks like normalization and feature engineering. Data cleaning will address missing values and outliers, ensuring quality for subsequent analysis and modeling, and forming a strong foundation for insights and predictions.

Section: Data Overview

Description:

Dimension: 1197 rows x 15 columns

```
[ ]  # checking the shape of the dataset
     df.shape

     (1197, 15)
```

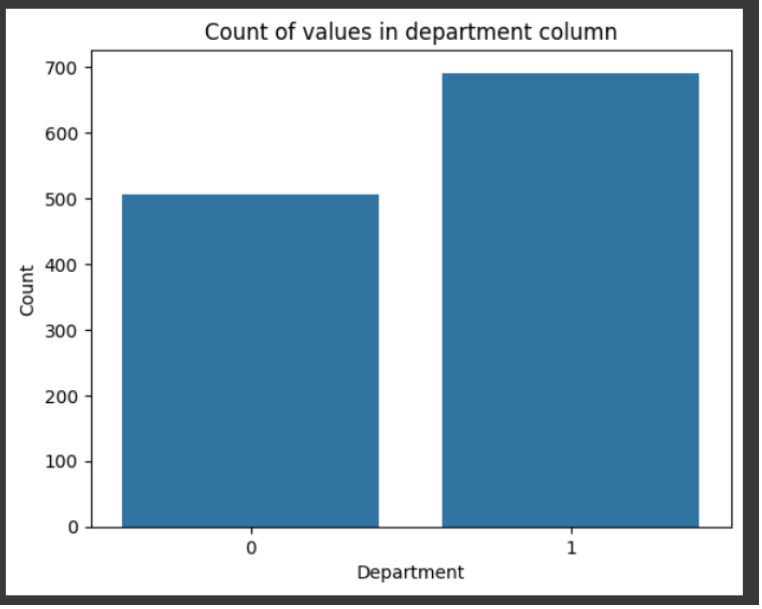
Descriptive Statistics:

```
[ ]  #statistical summaries of various numerical variables
     df.describe(include='all')
```

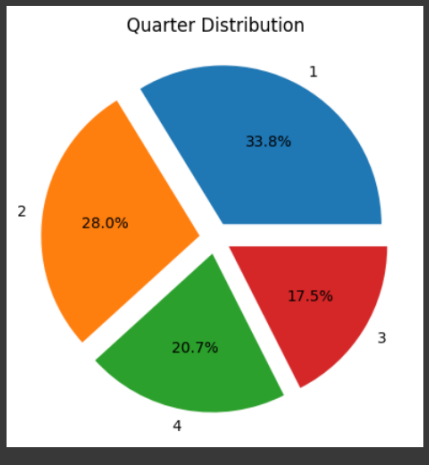
	quarter	department	day	team_number	std_minute_value
count	1197.000000	1197.000000	1197.000000	1197.000000	1197.000000
mean	2.252297	0.577277	2.534670	6.426901	14.508772
std	1.130974	0.494199	1.714538	3.463963	11.067638
min	1.000000	0.000000	0.000000	1.000000	2.000000
25%	1.000000	0.000000	1.000000	3.000000	3.000000
50%	2.000000	1.000000	3.000000	6.000000	15.000000
75%	3.000000	1.000000	4.000000	9.000000	24.000000
max	4.000000	1.000000	5.000000	12.000000	54.000000

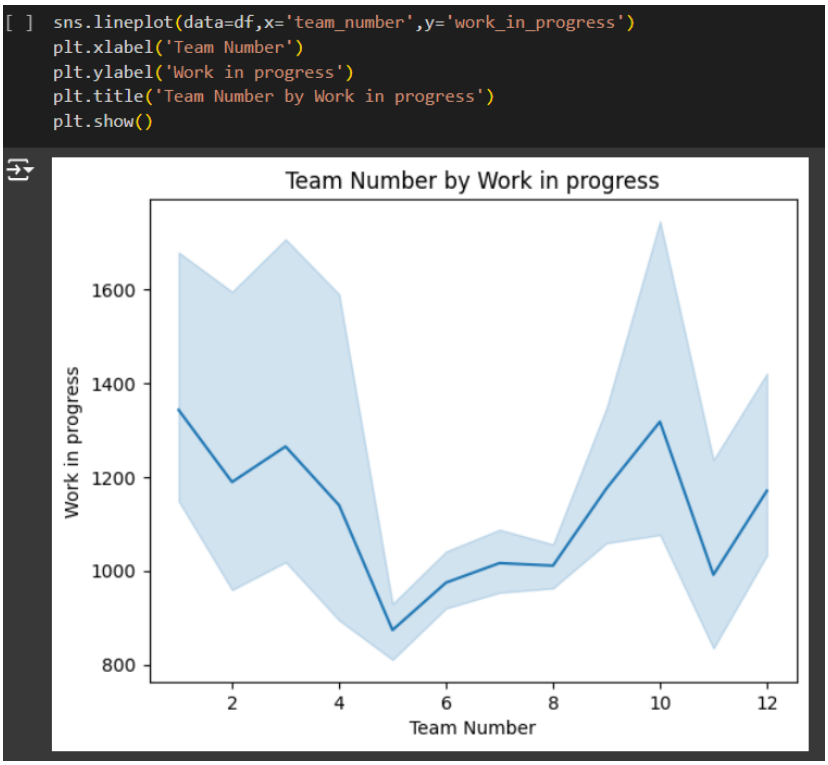
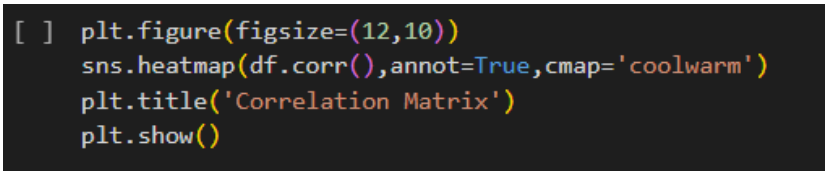
Univariate Analysis

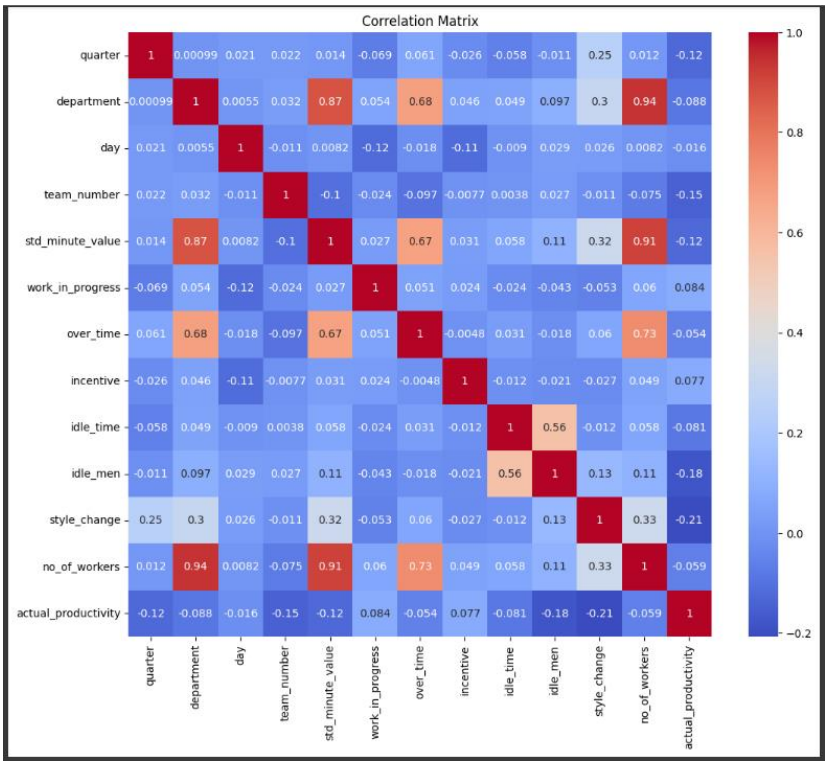
```
sns.countplot(data = df,x='department')
plt.xlabel('Department')
plt.ylabel('Count')
plt.title('Count of values in department column')
plt.show()
```




```
quarter_count=df['quarter'].value_counts()
plt.figure()
plt.pie(quarter_count,labels=quarter_count.index,autopct='%1.1f%%',explode=[0.1,0.1,0.1,0.1])
plt.title('Quarter Distribution')
plt.show()
```



Bivariate Analysis	
Multivariate Analysis	

<table>
<tr>
<td></td>
<td> 
</td>
</tr>
<tr>
<td>Outliers and Anomalies</td>
<td>Outliers:

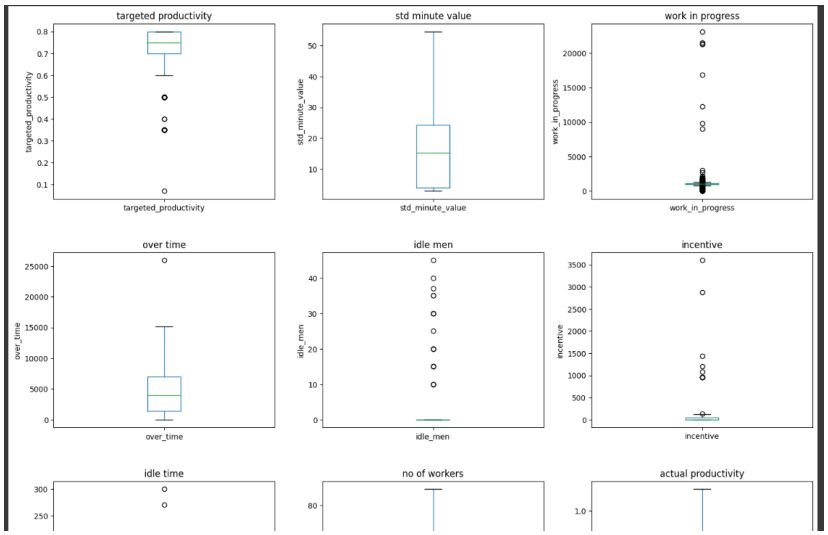

```
# Checking for outliers in continuous data

fig, axes = plt.subplots(nrows=3, ncols=3,figsize=(15,8))


#plotting boxplots
for i, ax in enumerate(axes.ravel()):
  df.boxplot(contin[i], grid=False, ax=ax)
  ax.set_title(f"{title(contin[i])}")
  ax.set_ylabel(contin[i])


fig.tight_layout()
plt.subplots_adjust(top=1.5)
plt.show()
```

</td>
</tr>
</table>



Anomalies:

```
#checking for anomalies

#None of the values in the continuous columns should be negative
for col in contin:
  print(len(df[df[col] < 0]))
#number of workers in each team should be a whole number
df['no_of_workers'].unique()
```

```
0
0
0
0
0
0
0
0
0
0
array([59. ,  8. , 30.5, 56. , 57.5, 55. , 54. , 18. , 60. , 12. , 20. ,
       17. , 56.5, 54.5, 29.5, 31.5, 31. , 55.5, 58. , 10. , 16. , 32. ,
       58.5, 15. ,  5. , 57. , 53. , 51.5,  2. ,  9. ,  7. , 19. , 28. ,
       34. , 89. , 14. , 25. , 52. ,  4. , 21. , 35. , 51. , 33. , 11. ,
       33.5, 22. , 26. , 27. , 59.5, 50. , 44. , 49. , 47. , 48. , 42. ,
       24. , 45. , 46. , 39. , 38. ,  6. ])
```

```
#to deal with the anomalies above, the figures will be truncated

df['no_of_workers'] = df['no_of_workers'].apply(lambda x: int(x))
#checking that the figures were truncated

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

```
array([59,  8, 30, 56, 57, 55, 54, 18, 60, 12, 20, 17, 29, 31, 58, 10, 16,
       32, 15,  5, 53, 51,  2,  9,  7, 19, 28, 34, 89, 14, 25, 52,  4, 21,
       35, 33, 11, 22, 26, 27, 50, 44, 49, 47, 48, 42, 24, 45, 46, 39, 38,
        6])
```

```
#checking for anomalies

for col in cat:
  print(col)
  print(df[col].unique())
  print('\n')
```

```
quarter
['Quarter1' 'Quarter2' 'Quarter3' 'Quarter4' 'Quarter5']


department
['sweing' 'finishing ' 'finishing']


day
['Thursday' 'Saturday' 'Sunday' 'Monday' 'Tuesday' 'Wednesday']


team
[ 8  1 11 12  6  7  2  3  9 10  5  4]


no_of_style_change
[0 1 2]
```

Data Preprocessing Code Screenshots

Loading Data

```
#reading csv file

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

```
#previewing top rows
df.head(4)
```

	date	quarter	department	day	team	targeted_productivity	smv	wip	over_time	incentive	idle_time
0	1/1/2015	Quarter1	sweing	Thursday	8	0.80	26.16	1108.0	7080	98	0.0
1	1/1/2015	Quarter1	finishing	Thursday	1	0.75	3.94	NaN	960	0	0.0
2	1/1/2015	Quarter1	sweing	Thursday	11	0.80	11.41	968.0	3660	50	0.0
3	1/1/2015	Quarter1	sweing	Thursday	12	0.80	11.41	968.0	3660	50	0.0

+ Code + Text

Handling Missing Data

```
#checking for missing values
df.isnull().sum()
```

```
date                        0
quarter                     0
department                  0
day                         0
team                        0
targeted_productivity       0
std_minute_value            0
work_in_progress          506
over_time                   0
incentive                   0
idle_time                   0
idle_men                    0
no_of_style_change          0
no_of_workers               0
actual_productivity         0
dtype: int64
```

```
#checking skewness
df.work_in_progress.skew()
```

```
9.741786273952965
```

```
#filling in missing values with median as opposed to mean since the data is skewed

df['work_in_progress'].fillna(df['work_in_progress'].median(),inplace=True)
```

```
#checking that there are no longer missing values
df.isnull().sum()
```

```
date                      0
quarter                   0
department                0
day                       0
team                      0
targeted_productivity     0
std_minute_value          0
work_in_progress          0
over_time                 0
incentive                 0
idle_time                 0
idle_men                  0
no_of_style_change        0
no_of_workers             0
actual_productivity       0
dtype: int64
```

Data Transformation

```
#datatype conversion

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['std_minute_value'] = df['std_minute_value'].astype(int)
df['style_change'] = df['style_change'].astype(int)
df['work_in_progress'] = df['work_in_progress'].astype(int)
df['idle_time'] = df['idle_time'].astype(int)
df['no_of_workers'] = df['no_of_workers'].astype(int)
df.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   int64
 1   department           1197 non-null   object
 2   day                  1197 non-null   object
 3   team_number          1197 non-null   int64
 4   std_minute_value     1197 non-null   int64
 5   work_in_progress     1197 non-null   int64
 6   over_time            1197 non-null   int64
 7   incentive            1197 non-null   int64
 8   idle_time            1197 non-null   int64
 9   idle_men             1197 non-null   int64
 10  style_change         1197 non-null   int64
 11  no_of_workers        1197 non-null   int64
 12  actual_productivity  1197 non-null   float64
dtypes: float64(1), int64(10), object(2)
memory usage: 121.7+ KB
```

```
[ ] lc = LabelEncoder()

[ ] #encoding the values in a column named "department"
    print('Before Encoding', df['department'].unique())
    df['department']= lc.fit_transform(df['department'])
    print('After Encoding', df['department'].unique())

    Before Encoding ['sewing' 'finishing']
    After Encoding [1 0]

[ ] #encoding the values in a column named "department"
    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]
```

<table>
<tr><td></td><td>

Applying Standard Scaler

```
[ ]  sc = StandardScaler()
     x_train = sc.fit_transform(x_train)
     x_test = sc.transform(x_test)
```

</td></tr>
<tr><td>Feature Engineering</td><td>

```
#renaming ambiguous columns
df = df.rename(columns={'wip': 'work_in_progress', 'smv':'std_minute_value'})

#separating variables for ease of analysis to those that can
#be considered continuous and categorical
cat = ["quarter", "department", "day", "team", "no_of_style_change"]
contin = ["targeted_productivity", "std_minute_value", "work_in_progress", "over_time",
```

```
[ ]  #function to remove _ from titles

     def title(x):
       return x.replace('_', ' ')
```

```
#replacing quarter5 (given to jan days above 28) with quarter1
df['quarter'] = df.quarter.str.replace('Quarter5', 'Quarter1')

#checking that there are now 4 quarters only
df.quarter.unique()
```

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

```
[ ]  # Removing the word Quarter from the quarter column and leave the numbers
     df['quarter'] = df['quarter'].str.replace('Quarter','')

     # Changing the datatype to numeric
     df['quarter'] = df['quarter'].astype(int)

     #confirming changes
     print(df.quarter.unique())
     print(df.quarter.dtype)

     #Correcting the spelling of sewing in the department column
     df['department'] = df['department'].str.replace('sweing','sewing')

     # Removing the spacing from the word finishing in the department column
     df['department'] = df['department'].str.replace('finishing ','finishing')

     #confirming changes
     df['department'].unique()
```

```
[1 2 3 4]
int64
array(['sewing', 'finishing'], dtype=object)
```

</td></tr>
<tr><td>Save Processed Data</td><td>-----</td></tr>
</table>