
Date: 31 July 2025

SkillWallet ID: SWUID20240141492
Project Title: Employee Performance Prediction using Machine Learning

---

Maximum Marks: 6 Marks

Data Exploration and Preprocessing Report

Dataset variables were statistically analyzed to identify patterns, correlations, and outliers. Python was employed for preprocessing tasks such as normalization, encoding, and feature engineering. Data cleaning ensured the removal of inconsistencies, missing values, and outliers, providing a strong foundation for model development.

Section Description

Data Overview:

- **Dimension:** 1,197 rows × 15 columns
- **Descriptive Statistics:**
    - Mean targeted productivity: **0.76**
    - Average overtime: **1.45 hours**
    - Average incentive: **110.35 (monetary units)**
    - Average idle time: **14.2 minutes**

Univariate Analysis:

- Plotted histograms for continuous features: **targeted_productivity, smv, over_time, incentive**.
- Observed skewness in **over_time** and **idle_time** distributions.

Bivariate Analysis:

- Scatter plots between **targeted_productivity** and **performance_score** indicated a strong positive correlation.
- Boxplots revealed that excessive **over_time** reduces actual performance efficiency.

Multivariate Analysis:

- Correlation heatmap showed high correlation between **idle_time** and **idle_men**.
- Productivity is influenced by a combination of **department**, **quarter**, and **incentive**.

-

Images

1.Loading Data

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

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving garments_worker_productivity.csv to garments_worker_productivity.csv
```

```
import pandas as pd

df = pd.read_csv("garments_worker_productivity.csv")
df.head()
```

	date	quarter	department	day	team	targeted_productivity	smv	wip	over_time	incentive	idle_time	idle_men	no_of_style_change	no_of_workers	actual_productivity
0	1/1/2015	Quarter1	sweing	Thursday	8	0.80	26.16	1108.0	7080	98	0.0	0	0	59.0	0.940725
1	1/1/2015	Quarter1	finishing	Thursday	1	0.75	3.94	NaN	960	0	0.0	0	0	8.0	0.886500
2	1/1/2015	Quarter1	sweing	Thursday	11	0.80	11.41	968.0	3660	50	0.0	0	0	30.5	0.800570
3	1/1/2015	Quarter1	sweing	Thursday	12	0.80	11.41	968.0	3660	50	0.0	0	0	30.5	0.800570
4	1/1/2015	Quarter1	sweing	Thursday	6	0.80	25.90	1170.0	1920	50	0.0	0	0	56.0	0.800382

2.Handling Missing Data:

4	1/1/2015	Quarter1	sweing	Thursday	6	0.80	25.90	1170.0	1920	50	0.0	0	0	56.0	0.800382

```
from sklearn.preprocessing import LabelEncoder
import numpy as np

df_clean = df.dropna()

label_encoders = {}
for col in ['quarter', 'department', 'day']:
    le = LabelEncoder()
    df_clean[col] = le.fit_transform(df_clean[col])
    label_encoders[col] = le

df_clean.head()
```

```
/tmp/ipython-input-3-644424723.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_clean[col] = le.fit_transform(df_clean[col])
/tmp/ipython-input-3-644424723.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_clean[col] = le.fit_transform(df_clean[col])
/tmp/ipython-input-3-644424723.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_clean[col] = le.fit_transform(df_clean[col])
```

	date	quarter	department	day	team	targeted_productivity	smv	wip	over_time	incentive	idle_time	idle_men	no_of_style_change	no_of_workers	actual_productivity
0	1/1/2015	0	0	3	8	0.8	26.16	1108.0	7080	98	0.0	0	0	59.0	0.940725
2	1/1/2015	0	0	3	11	0.8	11.41	968.0	3660	50	0.0	0	0	30.5	0.800570
3	1/1/2015	0	0	3	12	0.8	11.41	968.0	3660	50	0.0	0	0	30.5	0.800570
4	1/1/2015	0	0	3	6	0.8	25.90	1170.0	1920	50	0.0	0	0	56.0	0.800382
5	1/1/2015	0	0	3	7	0.8	25.90	984.0	6720	38	0.0	0	0	56.0	0.800125

Data Transformation:

```
[ ] scaler = StandardScaler()

[ ] scaler.fit(X_train)

    StandardScaler()

[ ] X_train_standardized = scaler.transform(X_train)

    print(X_train_standardized)

    [[ 1.40381088  1.79283426  1.37960065 ...  1.044121    0.52295995
       0.64990763]
     [ 1.16565505 -0.14461158  1.07121375 ...  0.5940779   0.44153782
      -0.85281516]
     [-0.0307278  -0.77271123 -0.09822185 ... -0.64047556 -0.31161687
      -0.69292805]
     ...
     [ 1.06478904  0.20084323  0.89267396 ...  0.01694621  3.06583565
      -1.29952679]
     [ 1.51308238  2.3170559   1.67987211 ...  1.14728703 -0.16599653
       0.82816016]
     [-0.73678981 -1.02636686 -0.74380549 ... -0.31826862 -0.40713129
      -0.38233653]]

[ ] X_test_standardized = scaler.transform(X_test)

[ ] print(X_train_standardized.std())

    1.0

[ ] print(X_test_standardized.std())

    0.8654541077212674
```