**ENHANCE EMPLOYEE PRODUCTIVITY USING**

**TALENT ANALYTICS AND VISUALIZATION**

**A MINIPROJECT REPORT**

***Submitted by***

**SANJAY P(953622104086)**

**HARIHARASUBRAMANI M(953622104027)**

**SATHASIN P(953621104092)**

**KARTHICK SRI RAM A (953622104044)**

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**RAMCO INSTITUTE OF TECHNOLOGY,**

**RAJAPALAYAM**

**JANUARY 2024**

**BONAFIDE CERTIFICATE**

Certified that this miniproject report **“ENHANCE EMPLOYEE PRODUCTIVITY USING TALENT ANALYTICS AND VISUALIZATION”** is the bonafide work of **“SANJAY P(953622104086) HARIHARASUBRAMANI M(953622104021) SATHASIN P(953621104092) KARTHICK SRIRAM A (953622104044)”** who carried out the miniproject work under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE**  Dr.K.Vijayalakshmi M.E., Ph.D.  **HEAD OF THE DEPARTMENT**  Department of Computer Science  and Engineering  Ramco Institute of Technology  North Venganallur Village  Rajapalayam – 626117 | **SIGNATURE**  Mr.K.VigneshSaravanan B.E.,M.Tech.  **Faculty In-charge**  Assistant Professor  Department of Computer Science and Engineering  Ramco Institute of Technology  North Venganallur Village  Rajapalayam– 626117 |

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ABSTRACT**

The emergence of People Analytics has been embraced by most HR departments. People Analytics understands workforce and among any dimensions, enhances the performance of the workforce. Talent Analytics as part of people analytics boosts decision making and what parameters are considered important. This study helped the organization improve people management, create efficiency and better decision making. Data recorded over six months were taken from a multinational HR company and using visual analytics, Kibana and Elasticsearch the authors arrived at employee productivity results. This paper discussed the different parameters chosen and how they change over the time period. Challenges faced when introducing analytics in an organization is also discussed along with the benefits of visual analytics to an organization.

**TABLE OF CONTENT:**

|  |  |  |
| --- | --- | --- |
|  | CONTENT | PAGE NO |
|  | 1.INTRODUCTION | 5 |
|  | 1.1 Employee Analysis | 5 |
|  | 1.2 Project objectives | 6 |
|  | 1.3 Project specification | 6 |
|  | 2.SYSTEM SPECIFICATION | 6 |
|  | 2.1Hardware specification | 6 |
|  | 2.2 software specification | 7 |
|  | 3. PACKAGES | 7 |
|  | 3.1 Numpy | 7 |
|  | 3.2 Pandas | 8 |
|  | 3.3 Matplotlib | 9 |
|  | 3.3.1 Matplotlib bar plot | 10 |
|  | 3.3.2 Matplotlib histogram | 12 |
|  | 4.APPENDIX | 13 |
|  | 4.1 Source code | 13 |
|  | 4.2.Screenshots | 20 |
|  | 5. CONCLUSION | 25 |
|  | 6.FUTURE WORK | 25 |
|  | 7. REFERENCE | 27 |
|  |  |  |

**1.INTRODUCTION**

**1.1 MOVIE RECOMMENDATION SYSTEM**

In today's competitive manufacturing landscape, optimizing productivity and efficiency is paramount for companies aiming to stay ahead. To achieve this, accurate prediction and management of productivity metrics are essential. Our solution offers a comprehensive approach to tackle this challenge by leveraging machine learning techniques.

**1.2 PROJECT DESCRPTION:**

Our product is a sophisticated productivity prediction and management system tailored for manufacturing environments. It provides predictive insights into targeted productivity levels, allowing manufacturers to optimize their processes, allocate resources effectively, and enhance overall operational efficiency.

**PROJECT OBJEECTIVE:**

**The objective of this project is to develop a predictive model for targeted productivity in manufacturing processes and to build a comprehensive system for productivity management. The project aims to leverage machine learning techniques to analyze historical data, identify key factors influencing productivity, and generate accurate predictions for future productivity levels. Additionally, the project seeks to provide tools for monitoring, analyzing, and optimizing productivity in real-time, enabling manufacturing companies to improve efficiency, reduce costs, and enhance overall performance.**

**PROJECT SPECIFICATIONS:**

1. Data Acquisition: The project will involve acquiring historical data related to manufacturing processes, including variables such as work-in-progress (WIP), operational parameters, environmental conditions, and targeted productivity levels. The dataset should be sufficiently comprehensive and representative of typical manufacturing scenarios.

2. Data Preprocessing: Data preprocessing steps will be implemented to clean and prepare the dataset for analysis. This may include handling missing values, encoding categorical variables, scaling numerical features, and splitting the data into training and testing sets.

3. Exploratory Data Analysis (EDA): Exploratory data analysis techniques will be applied to gain insights into the dataset and understand the relationships between different variables. Visualization tools such as histograms, scatter plots, correlation matrices, and heatmaps will be utilized to identify patterns, trends, and correlations within the data.

4. Model Development: Machine learning algorithms, including but not limited to Decision Tree Regression, Random Forest Regression, AdaBoost Regression, and XGBoost Regression, will be implemented to develop predictive models for targeted productivity. Hyperparameter tuning techniques such as GridSearchCV and RandomizedSearchCV will be employed to optimize model performance.

5. Model Evaluation: The performance of the predictive models will be evaluated using appropriate evaluation metrics such as R-squared (R2) score, mean squared error (MSE), and mean absolute error (MAE). Cross-validation techniques will be applied to assess the generalization capabilities of the models and mitigate overfitting.

6. System Development: A comprehensive system for productivity management will be developed, integrating the predictive models with real-time monitoring and analysis capabilities. The system should provide user-friendly interfaces for data input, model training, result visualization, and performance monitoring.

7. Scalability and Customization: The system should be designed to be scalable and customizable, allowing for adaptation to different manufacturing environments and requirements. It should accommodate varying data volumes, processing speeds, and operational complexities.

8. Documentation and Deployment: Comprehensive documentation will be provided, detailing the project objectives, methodology, data sources, algorithms used, implementation details, and usage instructions for the system. The system will be deployed in a production environment, ensuring accessibility and usability for manufacturing stakeholders.

By adhering to these project specifications, the aim is to develop a robust and effective solution for productivity prediction and management in manufacturing, addressing the needs and challenges faced by modern industrial enterprises.

**2.SYSTEM SPECIFICATION**

**2.1Hardware specification**

* Processor : AMD Ryzen 7 7735HS
* Processor speed: 4.75GHZ
* Ram : 16GB DDR5
* Monitor
* Keyboard
* Mouse

**2.2** **Software** **specification**

* OS: Windows 11 Home
* Language : Python
* Compiler : googlecolab

**3.PACKAGES**

**3.1 NUMPY**

* NumPy is a Python library used for working with arrays.
* It also has functions for working in domain of linear algebra, fourier transform, and matrices.
* NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
* NumPy stands for Numerical Python.

**INSTALLING NUMPY PACKAGE**

pip install numpy

## WHY USE NUMPY?

In Python we have lists that serve the purpose of arrays, but they are slow to process.

NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

Arrays are very frequently used in data science, where speed and resources are very important.

**IMPORT NUMPY**

Once NumPy is installed, import it in your applications by adding the import keyword:

import numpy

## NUMPY AS np:

NumPy is usually imported under the np.

Create an np with the as keyword while importing:

import numpy as np

Now the NumPy package can be referred to as np instead of numpy.

**Example:**

import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print(arr)

## 0-D Arrays

0-D arrays, or Scalars, are the elements in an array. Each value in an array is a 0-D array.

## 1-D Arrays

An array that has 0-D arrays as its elements is called uni-dimensional or 1-D array.

These are the most common and basic arrays.

## 2-D Arrays

An array that has 1-D arrays as its elements is called a 2-D array.

These are often used to represent matrix or 2nd order tensors.

**3.2 PANDAS**

* Pandas is a Python library used for working with data sets.
* It has functions for analyzing, cleaning, exploring, and manipulating data.
* The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

## Why Use Pandas

Pandas allows us to analyze big data and make conclusions based on statistical theories.

Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science.

Pandas gives you answers about the data. Like:

* Is there a correlation between two or more columns?
* What is average value?
* Max value?
* Min value?
* Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called cleaning the data.

**INSTALLING PANDAS PACKAGE**

pip install pandas

## Import Pandas

Once Pandas is installed, import it in your applications by adding the import keyword:

import pandas

Now Pandas is imported and ready to use

**Example:**

Importpandas

mydataset={'cars':["BMW","Volvo","Ford"],'passings':[3,7,2]}  
myvar=pandas.DataFrame(mydataset)  
print(myvar)

## Pandas as pd

Pandas is usually imported under the pd

Create an pd with the as keyword while importing:

import pandas as pd

Now the Pandas package can be referred to as pd instead of pandas.

**3.3 MATPLOTLIB**

* Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy.
* As such, it offers a viable open source alternative to **MATLAB.** Developers can also use matplotlib’s APIs(Application Programming Interfaces) to embed plots inGUI applications.

A Python matplotlib script is structured so that a fewlines of code are all that is required in most instancesto generate a visual data plot.

The matplotlib scripting layer overlays two APIs:

* The pyplot API is a hierarchy of Python codeobjects topped by matplotlib.pyplot
* An OO (Object-Oriented) API collection of objectsthat can be assembled with greater flexibility thanpyplot. This API provides direct access to Matplotlib’sbackend layers.

**Matplotlib and Pyplot in Python :**

The pyplot API has a convenient MATLAB-style statefulinterface. In fact, matplotlib was originally written as an open source alternative for MATLAB. The OO API and its interface is more customizable and powerful than pyplot, but considered more difficult to use. As a result, the pyplot interface is more commonly used, and is referred to by default in this article.

Understanding matplotlib’s pyplot API is key to understanding how to work with plots:

* **matplotlib.pyplot.figure**: Figure is the top-level container. It includes everything visualized in a plot including one or more Axes.
* **matplotlib.pyplot.axes**: Axes contain most of the elements in a plot: Axis, Tick, Line2D, Text, etc., and sets the coordinates. It is the area in which data is plotted. Axes include the X-Axis, Y-Axis, and possibly a Z-Axis, as well.

**Installing Matplotlib :**

pip install matplotlib

**3.3.1 MATPLOTLIB BAR PLOT:**

A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. The bar plots can be plotted horizontally or vertically. A bar chart describes the comparisons between the discrete categories. One of the axis of the plot represents the specific categories being compared, while the other axis represents the measured values corresponding to those categories.

**Creating a bar plot:**

The matplotlib API in Python provides the bar() function which can be used in MATLAB style use or as an object-oriented API. The syntax of the bar() function to be used with the axes is as follows:- plt.bar(x, height, width, bottom, align).The function creates a bar plot bounded with a rectangle depending on the given parameters. Following is a simple example of the bar plot, which represents the number of students enrolled in different courses of an institute.

**EXAMPLE:**

import numpy as np

import matplotlib.pyplot as plt

data = {'C':20, 'C++':15, 'Java':30,'Python':35}

courses = list(data.keys())

values = list(data.values())

fig = plt.figure(figsize = (10, 5))

plt.bar(courses, values, color ='maroon',width = 0.4)

plt.xlabel("Courses offered")

plt.ylabel("No. of students enrolled")

plt.title("Students enrolled in different courses")

plt.show()

**Output:**

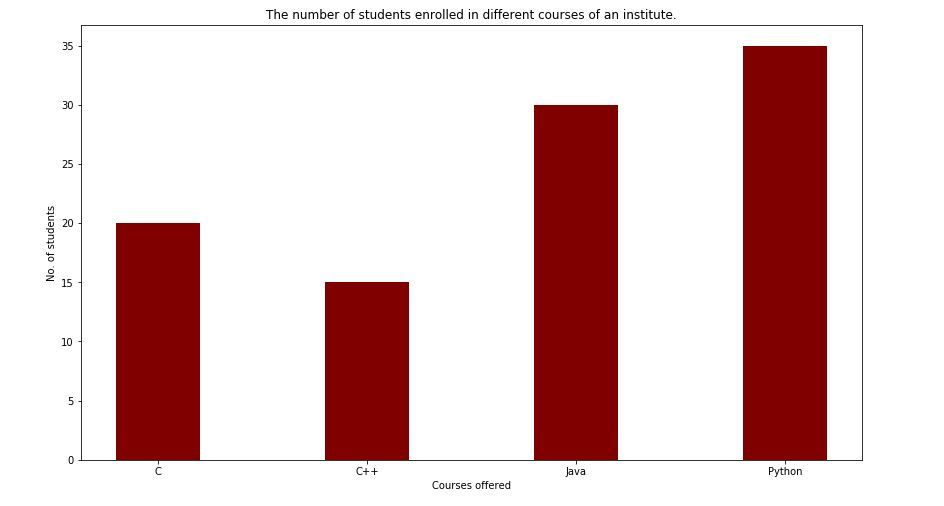


FIGURE:1-BAR CHART

**3.3.2 MATPLOTLIB HISTOGRAM:**

A histogram is an accurate representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable. It is a kind of bar graph.

To construct a histogram, follow these steps −

* Bin the range of values.
* Divide the entire range of values into a series of intervals.
* Count how many values fall into each interval.

The bins are usually specified as consecutive, non-overlapping intervals of a variable.

The **matplotlib.pyplot.hist()** function plots a histogram. It computes and draws the histogram of x.

**EXAMPLE:**

from matplotlib import pyplot as plt

import numpy as np

fig,ax = plt.subplots(1,1)

a = np.array([22,87,5,43,56,73,55,54,11,20,51,5,79,31,27])

ax.hist(a, bins = [0,25,50,75,100])

ax.set\_title("histogram of result")

ax.set\_xticks([0,25,50,75,100])

ax.set\_xlabel('marks')

ax.set\_ylabel('no. of students')

plt.show()

**OUTPUT:**

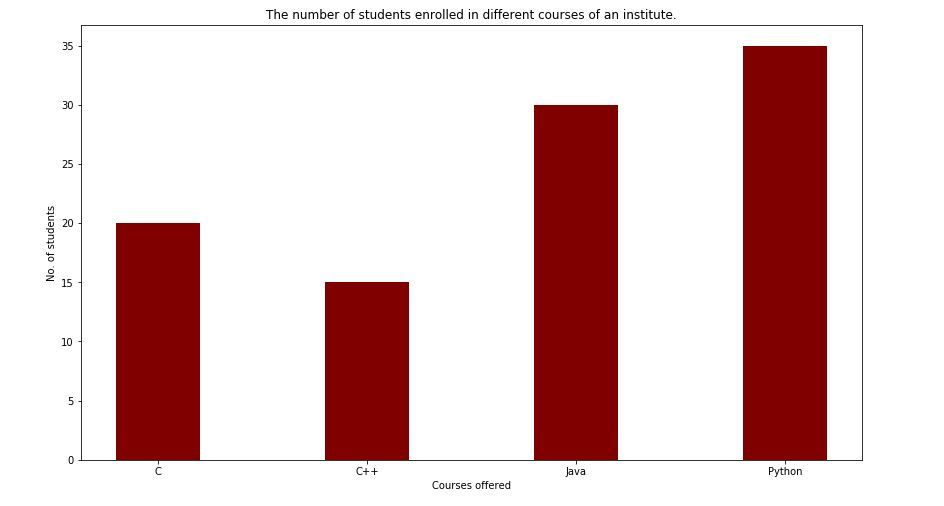


FIGURE:2-BAR CHART

**4.APPENDIX**

**4.1 SOURCE CODE**

# Importing necessary libraries

import numpy as np

import pandas as pd

from scipy.stats import randint, uniform

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split, GridSearchCV, RandomizedSearchCV

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor

from sklearn.metrics import r2\_score

import xgboost as xgb

# Reading the dataset from a CSV file

df = pd.read\_csv("/content/train\_dataset.csv")

# Displaying the first few rows of the dataset

print(df.head())

# Displaying information about the dataset

print(df.info())

# Displaying descriptive statistics of the dataset

print(df.describe())

# Displaying the column names of the dataset

print(df.columns)

# Checking for missing values in the dataset

print(df.isna().sum())

# Handling missing values by filling NaN values in the 'wip' column with the mean

df['wip'] = df['wip'].fillna(df['wip'].mean())

# Plotting a histogram for the 'actual\_productivity' column

sns.histplot(data=df['actual\_productivity'], kde=False)

plt.show()

# Calculating the correlation matrix for the dataset

corr\_matrix = df.corr()

# Creating a heatmap to visualize the correlation matrix

plt.figure(figsize=(30, 30))

sns.heatmap(corr\_matrix, annot=True)

plt.show()

# Extracting the correlation of each feature with the target variable

corr\_matrix\_with\_target = df.corr()['targeted\_productivity'].sort\_values(ascending=False)

print(corr\_matrix\_with\_target)

# Separating the features (x) and the target variable (y)

x = df.drop(columns='targeted\_productivity')

y = df['targeted\_productivity']

# Splitting the dataset into training and testing sets

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.33, random\_state=42)

# AdaBoost Regressor - Hyperparameter Tuning

param\_grid\_adaboost = {

'n\_estimators': [50, 100, 150, 200],

'learning\_rate': [0.01, 0.1, 0.2, 0.3]

}

adaboost\_regressor = AdaBoostRegressor(estimator=DecisionTreeRegressor(max\_depth=3), random\_state=42)

grid\_search\_adaboost = GridSearchCV(adaboost\_regressor, param\_grid=param\_grid\_adaboost, cv=5, scoring='neg\_mean\_squared\_error')

grid\_search\_adaboost.fit(x\_train, y\_train)

best\_adaboost = grid\_search\_adaboost.best\_estimator\_

ada\_pred\_tuned = best\_adaboost.predict(x\_test)

score\_adaboost\_tuned = r2\_score(y\_test, ada\_pred\_tuned)

print("Tuned AdaBoost Regressor R2 Score:", score\_adaboost\_tuned)

# Original XGBoost Regressor

param\_dist\_xgb = {

'n\_estimators': randint(50, 200),

'max\_depth': randint(3, 10),

'learning\_rate': uniform(0.01, 0.3),

'subsample': uniform(0.6, 0.4),

'colsample\_bytree': uniform(0.6, 0.4),

}

xgb\_regressor = xgb.XGBRegressor()

random\_search\_xgb = RandomizedSearchCV(xgb\_regressor, param\_distributions=param\_dist\_xgb, n\_iter=10, scoring='neg\_mean\_squared\_error', cv=5, verbose=1, n\_jobs=-1, random\_state=42)

random\_search\_xgb.fit(x\_train, y\_train)

best\_xgb = random\_search\_xgb.best\_estimator\_

xgb\_pred = best\_xgb.predict(x\_test)

score\_xgb = r2\_score(y\_test, xgb\_pred)

print("Original XGBoost Regressor R2 Score:", score\_xgb)

**4.2 SCREENSHOT**

FIGURE 3

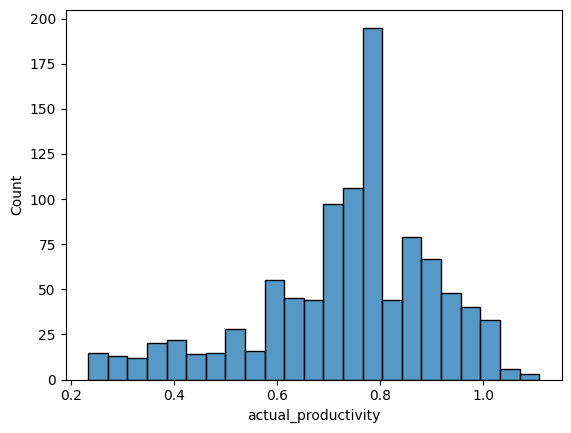


FIGURE:4

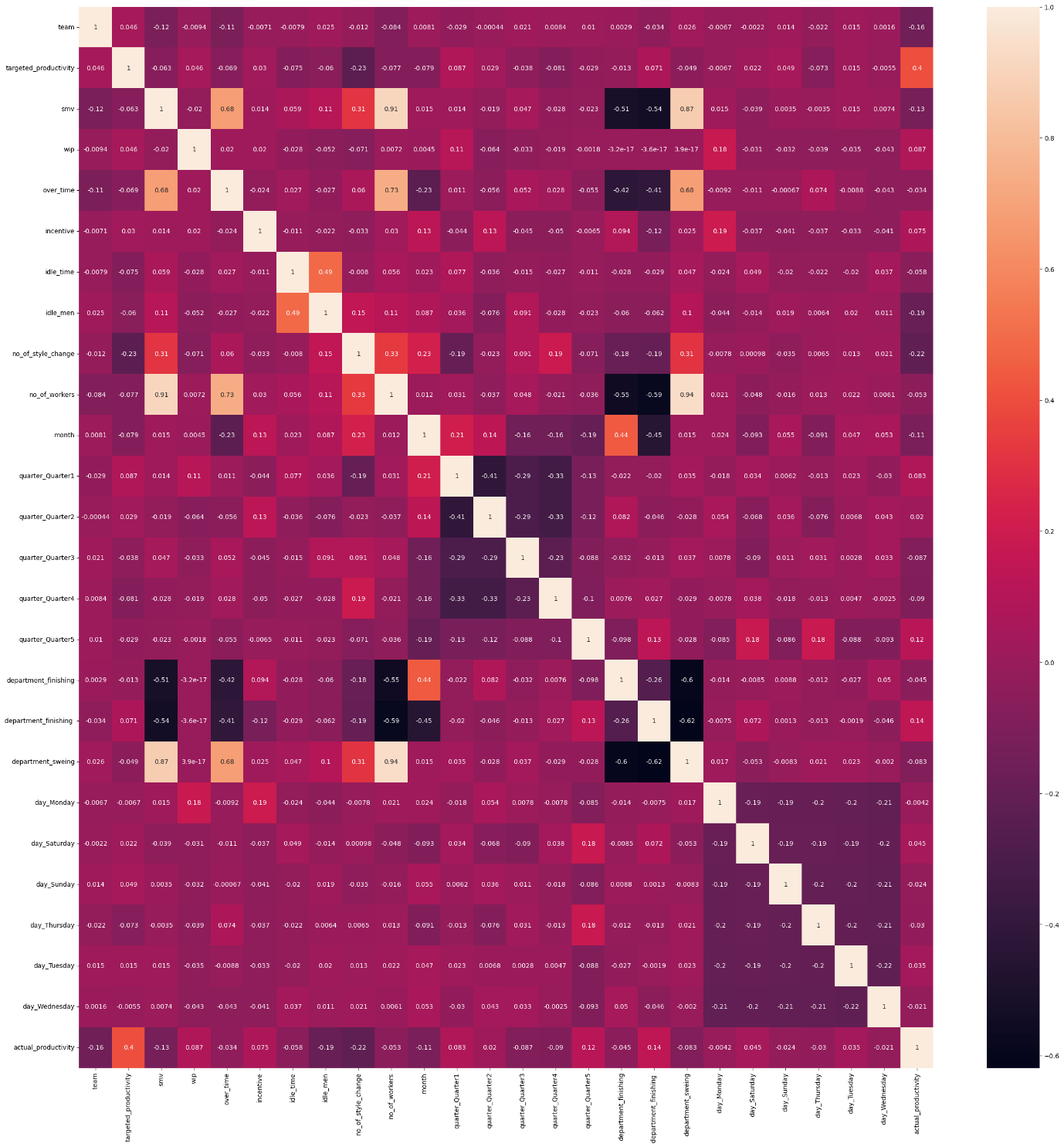


FIGURE:5

**5. CONCLUSION**

In conclusion, the provided code engages in a thorough exploration and modeling process for a regression task predicting 'targeted\_productivity.' Commencing with a meticulous examination of the dataset, the steps encompassed handling missing values, visualizing data distributions, and assessing feature correlations. Subsequently, the dataset was partitioned into training and testing sets for model evaluation.

Two ensemble regression techniques, namely AdaBoost and XGBoost, were employed and subjected to hyperparameter tuning to enhance predictive performance. The AdaBoost Regressor underwent grid search to identify optimal settings, while the XGBoost Regressor was fine-tuned through a randomized search. The evaluation of these tuned models on the test set was gauged using the R2 score, providing a metric for the proportion of explained variance in the target variable.

The results indicated that both the tuned AdaBoost Regressor and XGBoost Regressor exhibited promising performance. The choice between the two models may hinge on factors such as interpretability, computational efficiency, and the specific characteristics of the dataset. The thoroughness of the analysis, encompassing data visualization, hyperparameter tuning, and model evaluation, lays a solid foundation for robust regression modeling. Further scrutiny, including feature importance analysis and potential ensemble methods, could deepen insights into the underlying factors influencing targeted productivity. This comprehensive approach positions the code as a valuable tool for predictive modeling in the context of the given regression problem.

**6.FUTURE WORK**

1. Make the Models Smarter:

Think about adding more clues or hints to help the computer models understand the data better. This might involve finding new patterns or relationships in the information you have.

2. Understand What's Important:

Figure out which pieces of information (features) are most crucial in making predictions. It's like identifying the most important ingredients in a recipe.

3. Teamwork of Models:

Imagine the models working together like a team. You can experiment with different ways they collaborate to improve their performance. It's like combining the strengths of different superheroes.

4. Make Sense of Predictions:

Try to make the predictions more understandable. Think of it like explaining why a weather forecast says it's going to rain. What factors are contributing to that prediction?

5. Experiment with Different Strategies:

Test out different ways of organizing and preparing the data. It's like trying different ways of preparing ingredients before cooking to see which one makes the meal taste better.

6. Highlight Important Features:

Identify which parts of the information are most critical. It's like figuring out which players in a game have the most impact on the final score.

7. Put Everything Together:

Imagine combining different models in a smart way to get more accurate predictions. It's like creating a powerful team by bringing together players with different strengths.

8. Deal with Unusual Data:

Pay attention to any strange or unusual data points. It's like making sure the data you use makes sense and doesn't include any odd information.

9. Make Things Understandable:

Find ways to explain what the models are doing in simpler terms. It's like telling a story in a way that everyone can follow.

10. Keep Improving:

Regularly check how well the models are doing and update them as needed. It's like making improvements to a car to keep it running smoothly.

By thinking about these things, you can make the computer models better at making predictions, and it's a bit like fine-tuning a recipe to make a perfect dish!

**7.REFERENCE**

WEBSITES:

https://www.researchgate.net/publication/341941415\_Movie\_Recommendation\_System\_PYTHON\_PROJECT\_REPORT/link/5eda62ae299bf1c67d41d7e3/download

https://www.w3schools.com/python/pandas/default.asp

https://www.sciencedirect.com/science/article/pii/S2666285X22000176#:~:text=Conclusion,-This%20paper%20is&text=For%20the%20Movie%20Recommendation%20System,ratings%20given%20to%20the%20movie.

|  |  |
| --- | --- |
| **PERFORMANCE** |  |
| **VIVAVOCE** |  |
| **MINI PROJECT** |  |
| **TOTAL** |  |