STUDENT DETAILS

MANNEM YASWANTH SAI

YASWANTHSAI_MANNEM@SRMAP.EDU.IN

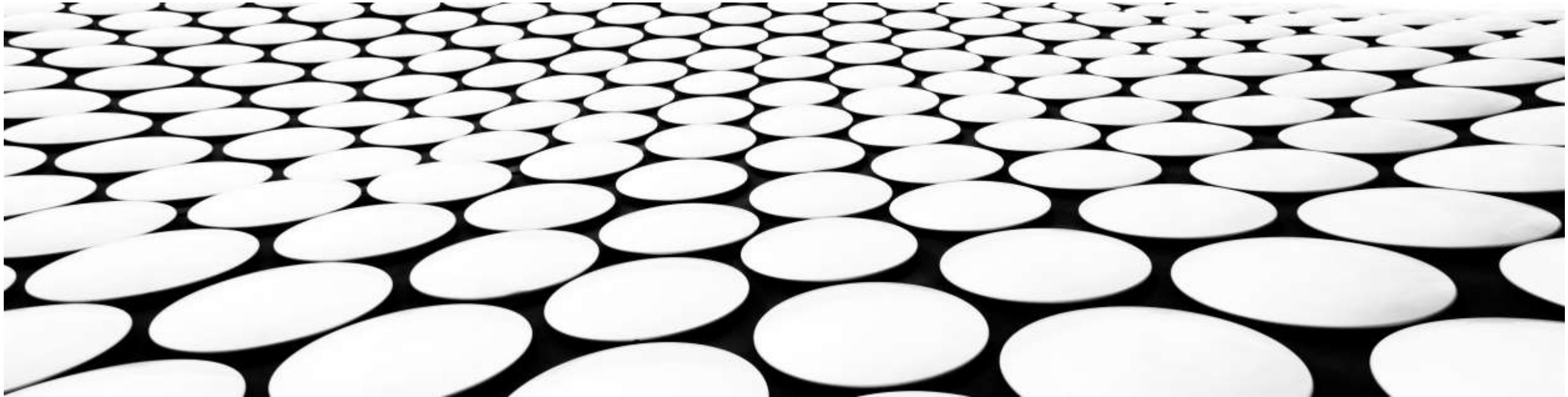
SRM

ANDHRA PRADESH,AMARAVATHI

START DATE: 05/06/2024

END DATE: 25/07/2024

EMPLOYEE BURN RATE ANALYSIS USING LINEAR REGRESSION



AGENDA

- Problem statement
- Project overview
- Who are the End users of the project?
- My solution and its value proposition
- Project Customization trails and its defects
- Modelling
- Result and links

PROBLEM STATEMENT

- **Definition and Importance of Burnout Analysis**: Burnout is a state of emotional, physical, and mental exhaustion caused by excessive and prolonged stress. Analyzing burnout is crucial as it helps organizations identify and address the root causes of burnout, promoting healthier work environments and enhancing employee well-being.
- **Increasing Incidence of Burnout**: Employee burnout is becoming increasingly common in various industries, leading to reduced productivity, higher turnover rates, and significant mental and physical health issues among employees.
- **Lack of Early Detection**: Organizations often fail to identify the early signs of burnout, resulting in prolonged stress and decreased overall employee well-being.
- **Limited Predictive Tools**: Existing tools and methods for detecting and managing burnout are often reactive rather than proactive, lacking the ability to predict burnout before it severely impacts employees.
- **Impact on Organizational Performance**: Unaddressed employee burnout can lead to significant financial and reputational costs for organizations, making it imperative to develop effective solutions to mitigate this issue.

PROJECT OVERVIEW

- **Objective:** The project aims to develop a predictive model using linear regression to analyze employee burnout based on various factors such as demographics, work setup, and mental fatigue.
- **Use of the Project:** By pinpointing the primary factors contributing to burnout, the project aims to assist organizations in implementing proactive measures to manage and reduce employee burnout. This predictive capability enables targeted interventions, fostering a healthier and more productive workplace environment.

WHO ARE THE END USERS OF THIS PROJECT?

- Human Resources (HR) Professionals: HR professionals are primary end users of the burnout analysis model. They can utilize the insights to proactively address employee burnout, improve work conditions, enhance employee satisfaction, and reduce turnover rates. By identifying high-risk individuals or groups, HR can implement targeted interventions to maintain a healthier, more productive workforce.
- Management and Executive Teams: Management and executive teams can use the burnout predictions to inform strategic decision-making. Understanding burnout trends and risk factors allows them to allocate resources effectively, design better policies, and foster a supportive work culture. This helps in improving overall organizational performance and maintaining a positive company reputation

YOUR SOLUTION AND ITS VALUE PROPOSITION

- My solution involves creating a predictive model using linear regression to analyze and forecast employee burnout rates. The model utilizes a dataset sourced from Kaggle, which includes various employee-related factors such as Employee ID, Date of Joining, Gender, Company Type, WFH Setup Available, Designation, Resource Allocation, Mental Fatigue Score, and Burn Rate. The approach includes the following

SOLUTION

- **Data Preprocessing**: Cleaning and preparing the dataset by handling missing values, encoding categorical variables, and normalizing numerical features to ensure accurate and reliable predictions.
- **Model Development**: Utilize a linear regression model to predict the Burn Rate of employees by analyzing the relationships between the input features (such as Mental Fatigue Score, Resource Allocation) and the target variable (Burn Rate).
- **Evaluation and Validation**: Assess the model's performance using evaluation metrics such as R-squared and Mean Squared Error (MSE), and validate its accuracy with a test dataset to ensure reliability.
- **Implementation and Insights**: Implement the model to predict the burnout rate of employees, focusing on key predictive factors identified through the analysis.

ITS VALUE PROPOSITION

- **Enhanced Employee Well-being**: The predictive model enables organizations to proactively manage employee burnout by identifying at-risk employees early. This allows HR and management teams to implement targeted interventions, improving overall employee well-being and satisfaction, and reducing turnover rates.
- **Strategic Resource Allocation**: By providing data-driven insights into the factors contributing to burnout, the model helps organizations make informed decisions about resource allocation, work policies, and employee support programs. This strategic approach enhances productivity, fosters a healthier work environment, and supports the organization's long-term success.

HOW DID YOU CUSTOMIZE THE PROJECT AND MAKE IT YOUR OWN

- I Tried not to drop the rows and instead fill those values using various imputation methods as we will loose 14% of the data if we drop the rows so I tried filling the data with mean , mode, median in different ways. I also tried using knn imputer(k nearest neighbours)
- But I didn’t get the accuracy that is attained by dropping the all the rows with null values.

USING MEAN TO FILL DATA

These are the error values and r squared Score after filling the null values with mean

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
columns_to_impute = ['Resource Allocation', 'Mental Fatigue Score']
imputer = SimpleImputer(strategy='mean')
data_cleaned[columns_to_impute] = imputer.fit_transform(data_cleaned[columns_to_impute])
print(data_cleaned.isnull().sum())
```

```
Linear Regression Model Performance Metrics:

Mean Squared Error: 0.005077869795585186
Root Mean Squared Error: 0.0712591734135696
Mean Absolute Error: 0.053658146503600034
R-squared Score: 0.8707522444395512
```

USING MODE TO FILL DATA

These are the error values and r squared Score after filling the null values with mode

```
Linear Regression Model Performance Metrics:

Mean Squared Error: 0.0050863580833320067
Root Mean Squared Error: 0.071318707807419 42
Mean Absolute Error: 0.053629559476704 46
R-squared Score: 0.8705361908220997
```

```
#this is with filling null values with mode
```

USING MEDIAN TO FILL DATA

These are the error values and r squared Score after filling the null values with median

```
Linear Regression Model Performance Metrics:

Mean Squared Error: 0.005079354488165694
Root Mean Squared Error: 0.0712695902062422
Mean Absolute Error: 0.05360532168320186
R-squared Score: 0.8707144543442056
```

```
#this is with filling null values with median
```

USING KNN TO FILL DATA

These are the error values and r squared Score after filling the null values with KNN imputer

```
Linear Regression Model Performance Metrics:

Mean Squared Error: 0.0041239082922526 16
Root Mean Squared Error: 0.06421766339764019
Mean Absolute Error: 0.050014386285775 36
R-squared Score: 0.8950335647884904
```

```
#this is with filling null values with knn imputer
```

USING KNN TO FILL DATA

These are the error values and r squared Score after filling the null values with KNN imputer after dropping rows with two or more null values (i.e 187 rows) ,this accuracy did come near to that of dropping all rows but not better .

```
Linear Regression Model Performance Metrics:

Mean Squared Error: 0.0038162195903115636
Root Mean Squared Error: 0.06177555819506258
Mean Absolute Error: 0.049100294686696966
R-squared Score: 0.9041008402130281
```

```
#this is with filling null values with knn imputer after droping the rows with two or more null values
```

MODELLING

- Importing libraries
- Handling missing values
- Analysing categorical data
- Encoding categorical data
- Splitting dataset and scaling
- Fitting into linear regression model

IMPORTING LIBRARIES

Importing libraries(i.e. Numpy, pandas, matplotlib, skikit) and loading the data set

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

```
data = pd.read_excel(r"C:\Users\mjaya\Downloads\employee_burnout_analysis-AI.xlsx")
```

```
data
```

	Employee ID	Date of Joining	Gender	Company Type	WFH Setup Available	Designation	Resource Allocation	Mental Fatigue Score	Burn Rate
0	fffe32003000360033003200	2008-09-30	Female	Service	No	2	3.0	3.8	0.16
1	fffe3700360033003500	2008-11-30	Male	Service	Yes	1	2.0	5.0	0.36
2	fffe31003300320037003900	2008-03-10	Female	Product	Yes	2	NaN	5.8	0.49
3	fffe32003400380032003900	2008-11-03	Male	Service	Yes	1	1.0	2.6	0.20
4	fffe31003900340031003600	2008-07-24	Female	Service	No	3	7.0	6.9	0.52

HANDLING MISSING VALUES

We have dropped all the rows which consists null values to handle missing data and after that we have found correlation of the columns with numeric data type to see their percentage effect on Burn rate.

```
data_cleaned.isnull().sum()
```

```
Employee ID                0
Date of Joining            0
Gender                     0
Company Type               0
WFH Setup Available        0
Designation                0
Resource Allocation     1278
Mental Fatigue Score    1945
Burn Rate                  0
dtype: int64
```

```
data_cleaned=data_cleaned.dropna()
```

```
data_cleaned.shape #we haveremoved 3036 rows from 21626 rows
```

```
(18590, 9)
```

```
data_cleaned.corr(numeric_only=True)['Burn Rate'][:-1]
```

```
Designation             0.736412
Resource Allocation     0.855005
Mental Fatigue Score    0.944389
Name: Burn Rate, dtype: float64
```

HANDLING MISSING VALUES

We have also tried filling the data with KNN imputer as well but that didn't give the accurate results as much as we get by dropping all the rows.

```
from sklearn.impute import KNNImputer
# Step 1: Drop rows where both 'Resource Allocation' and 'Mental Fatigue Score' are null
data_cleaned = data_cleaned[~(data_cleaned['Resource Allocation'].isnull() & data_cleaned['Mental Fatigue Score'].isnull())]

# Step 2: Apply KNN imputer to the remaining rows with missing values in 'Resource Allocation' or 'Mental Fatigue Score'
columns_to_impute = ['Resource Allocation', 'Mental Fatigue Score']

# Initialize the KNN imputer
knn_imputer = KNNImputer(n_neighbors=5)

# Perform KNN imputation and assign the transformed values back to the DataFrame
data_cleaned[columns_to_impute] = knn_imputer.fit_transform(data_cleaned[columns_to_impute])

# Print the number of missing values in each column to confirm imputation
print(data_cleaned.isnull().sum())
```

ANALYSING CATEGORICAL DATA

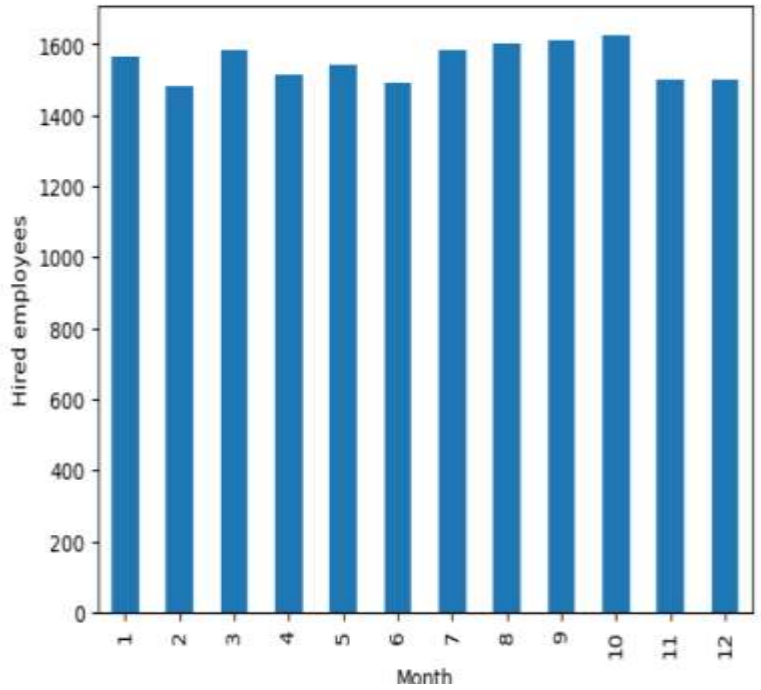
We have dropped the Employee ID column since it shows no effect on the burn rate, and to check if the columns Date of Joining we convert it into date time format and calculate from how many days the employee is working for.

```
data_cleaned=data_cleaned.drop('Employee ID',axis=1)
```

```
print(f"Min date {data_cleaned['Date of Joining'].min()}")
print(f"Max date {data_cleaned['Date of Joining'].max()}")
# Convert Date of Joining to datetime format
data_cleaned["Date of Joining"] = pd.to_datetime(data_cleaned["Date of Joining"])
# Group by month and count the number of employees hired each month
data_cleaned["Date of Joining"].groupby(data_cleaned['Date of Joining'].dt.month).count().plot(kind="bar", xlabel='Month', ylabel="Hired employees")
```

```
Min date 2008-01-01 00:00:00
Max date 2008-12-31 00:00:00
```

```
<Axes: xlabel='Month', ylabel='Hired employees'>
```



ANALYSING CATEGORICAL DATA

After converting it in to days we check the correlation of it with burn rate as it is in numerical format aswell ,after checking we got very less correlation with burn rate so we dropped the column Date of Joining.

```
data_2008 = pd.to_datetime(["2008-01-01"]*len(data_cleaned))
# Specify time unit as nanoseconds when converting to datetime64
data_cleaned["Days"] = data_cleaned['Date of Joining'].astype("datetime64[ns]").sub(data_2008).dt.days
data_cleaned.Days
```

```
0        273
1        334
3        307
4        205
5        330
        ...
22743    349
22744    147
22746     18
22748      9
22749      5
Name: Days, Length: 18590, dtype: int64
```

```
numeric_data = data_cleaned.select_dtypes(include=['number'])
correlation = numeric_data.corr()['Burn Rate']
print(correlation)
```

```
Designation             0.736412
Resource Allocation     0.855005
Mental Fatigue Score    0.944389
Burn Rate               1.000000
Days                    0.000309
Name: Burn Rate, dtype: float64
```

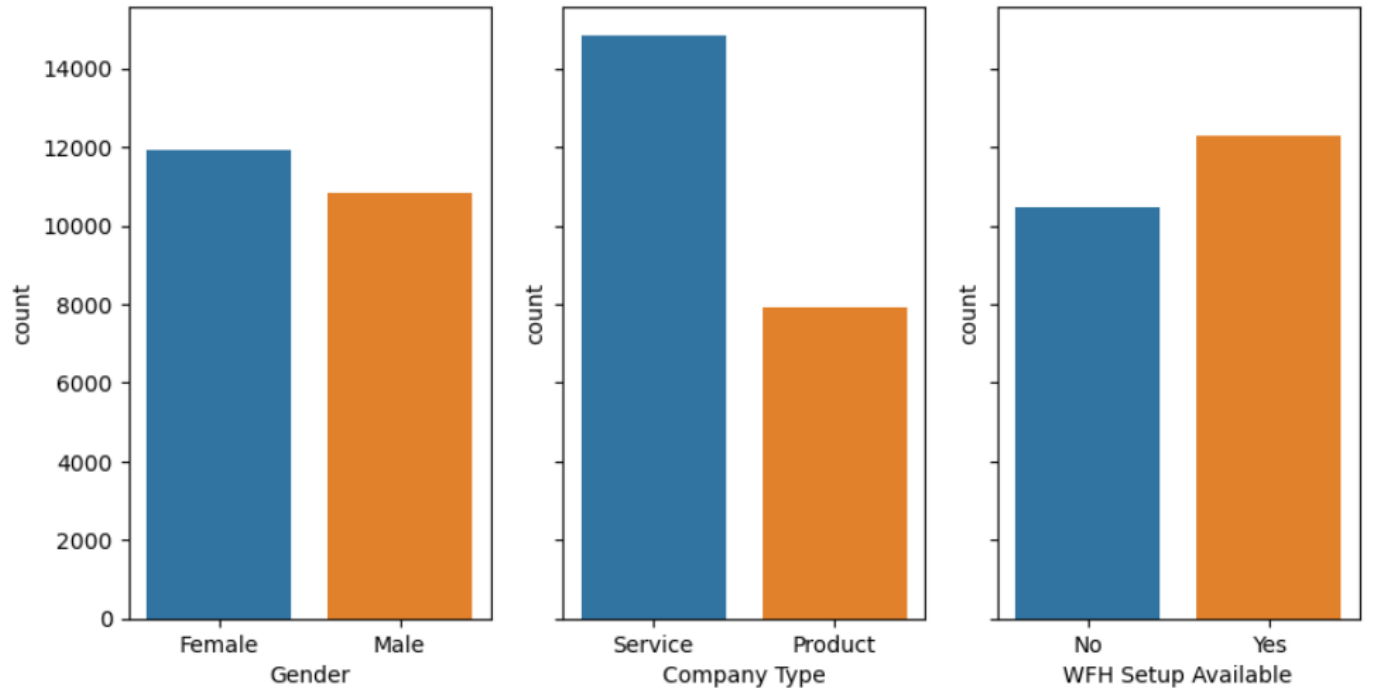
```
#Days are not strongly correlated to burnrate so we can drop it
data_cleaned = data_cleaned.drop(['Date of Joining','Days'], axis = 1)
```

ANALYSING CATEGORICAL DATA

Analysing remaining categorical columns ,and proceeding further. Leaving the conclusion that the data in each column is normally distributed except in the company column.

```
#Analysing Catagorical variables
```

```
cat_columns = data_cleaned.select_dtypes(object).columns
fig, ax = plt.subplots(nrows=1, ncols=len(cat_columns), sharey=True, figsize=(10, 5))
for i, c in enumerate(cat_columns):
    sns.countplot(x=c, data=data, ax=ax[i])
plt.show()
```



ENCODING CATEGORICAL DATA

We are using one hot encoding to encode the categorical data . In this process we Encode the categorical values with dummy numerical values .

```
if all(col in data_cleaned.columns for col in ['Company Type', 'WFH Setup Available', 'Gender']):
    data_cleaned = pd.get_dummies(data_cleaned, columns=['Company Type', 'WFH Setup Available','Gender'], drop_first=True)
    data_cleaned.head()
    encoded_columns = data_cleaned.columns
else:
    print("Error: One or more of the specified columns are not present in the DataFrame.")
    print(data_cleaned.columns)
```

SPLITTING DATASET AND SCALING

Here we split the data set into training and testing data set into 7:3 ratio, to fit into linear regression model.

And to scale the data we used StandardScaler.

```
y=data_cleaned['Burn Rate']
X=data_cleaned.drop('Burn Rate',axis=1)
```

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, shuffle=True, random_state=1)

# Scale X
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = pd.DataFrame(scaler.transform(X_train), index=X_train.index, columns=X_train.columns)
X_test = pd.DataFrame(scaler.transform(X_test), index=X_test.index, columns=X_test.columns )
```

FITTING INTO LINEAR REGRESSION MODEL

Here we fit the training data set into linear regression model.

We now have 5 independent variables(i.e. Designation , Resource Allocation, MFS, Company Type , WFH setup) and one dependent variable that is to be predicted(i.e. burn rate)

```
linear_regression_model = LinearRegression()
linear_regression_model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

RESULTS

Here is the result that how accurate our model is able to predict the dependent variable using testing set (By comparing the y_test and y_predicted).

```
#Linear Regressing Model Performance Metrics

print("Linear Regression Model Performance Metrics:\n")
# Make predictions on the test set
y_pred = linear_regression_model.predict(X_test)

# Calculate mean squared error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)

# Calculate root mean squared error
rmse = mean_squared_error(y_test, y_pred, squared=False)
print("Root Mean Squared Error:", rmse)

# Calculate mean absolute error
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error:", mae)

# Calculate R-squared score
r2 = r2_score(y_test, y_pred)
print("R-squared Score:",r2)
```

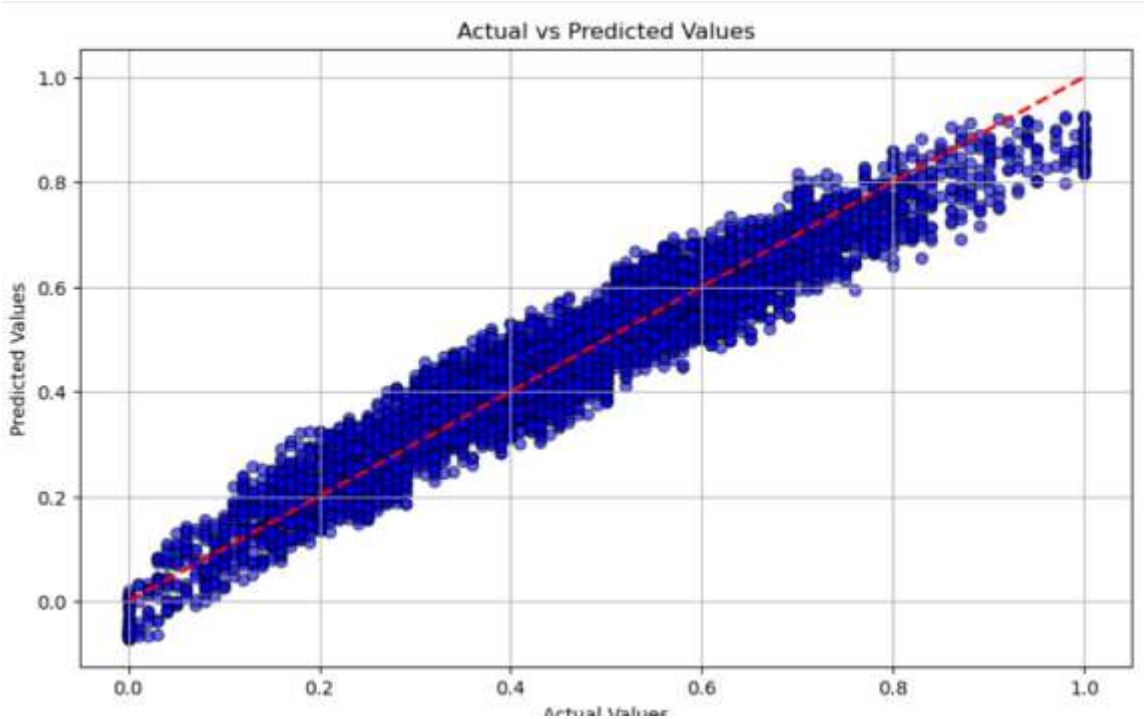
```
Linear Regression Model Performance Metrics:

Mean Squared Error: 0.003156977911361073
Root Mean Squared Error: 0.056186990588223115
Mean Absolute Error: 0.045950320326447726
R-squared Score: 0.918822674247248
```

```
#this is obtained after droping all the rows with null values (i.e 3036 rows from 21626 rows)
```

RESULTS

The red dotted line is the line that represents that our predicted values are best fit



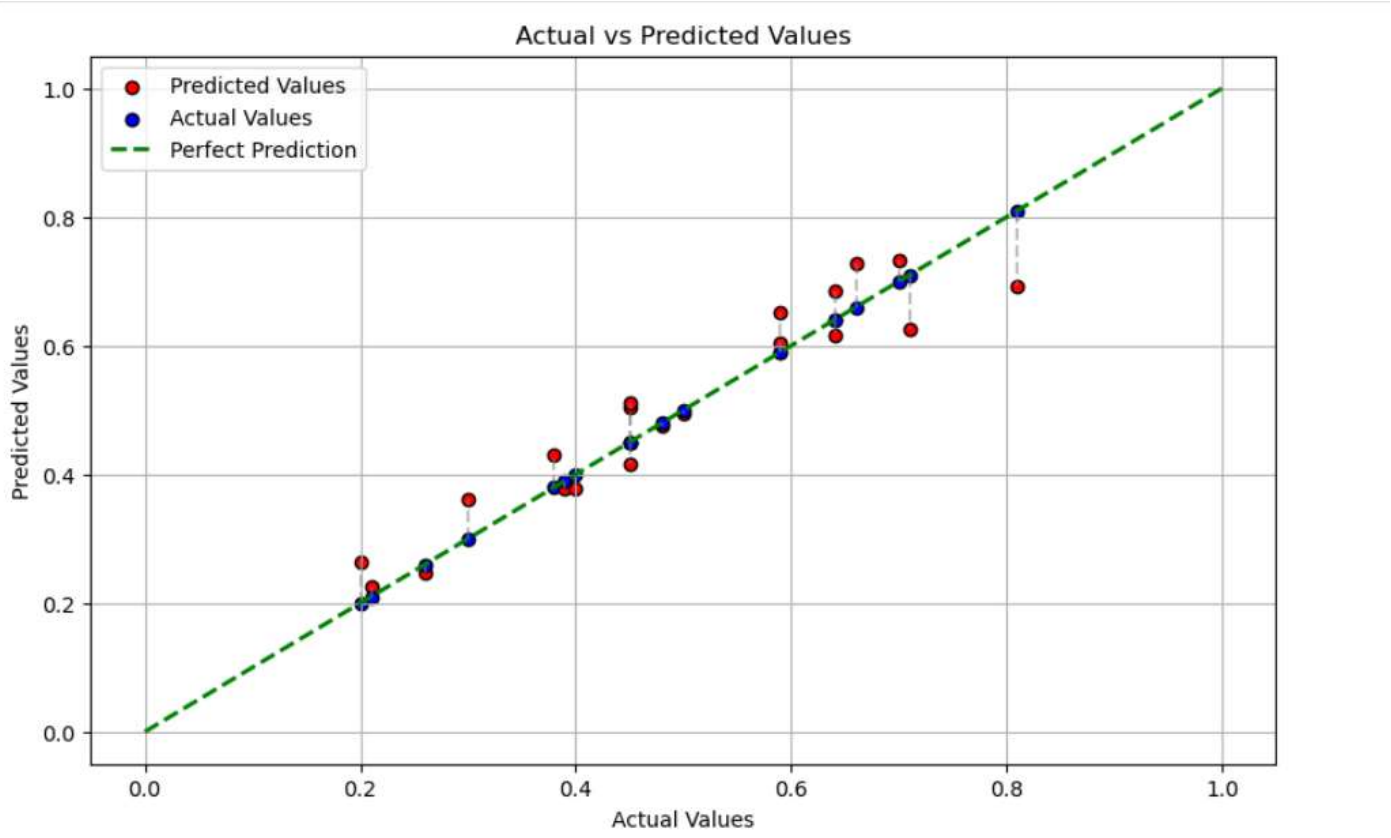
Visualizing the relationship between the predicted and actual values means creating a plot where you can compare the predictions made by your model to the actual values from your dataset. This is typically done using a scatter plot where:

- The **x-axis** represents the actual values from your dataset.
- The **y-axis** represents the predicted values from your model.

In this plot, each point represents an observation from your dataset. If your model were perfect, all points would lie on a 45-degree line (y = x), because the predicted values would exactly match the actual values. The best fit line in this context is the 45-degree line which indicates perfect prediction.

RESULTS

Here we took 20 random samples and plotted their actual and predicted values on the best fit line.



FINAL TAKEAWAYS

- We have built a machine leaning model
  - Using linear regression
- We can now find out the Burn rate by taking the inputs of the independent variables
- We have found out the factors that are majorly contributing to Burnout
- With all these outputs we can create a better environment for employees by preventing them from stress and burnout

THANKYOU

YASWANTH SAI MANNEM

AP22110011084

SRM UNIVERSITY

YASWANTHSAI_MANNEM@SRMAP.EDU.IN