```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

%matplotlib inline

dataset = pd.read_csv('11Absenteeism_at_work.csv')

dataset.head(10)

dataset.columns

#shape and type of dataset


print('Shape of dataset is:{}'.format(dataset.shape))

print('Type of features is:{}'.format(dataset.dtypes))

#mean of column 'Absenteeism time in hours'

dataset['Absenteeism time in hours'].mean()

#histogram view of Absenteeism time in hours

plt.hist(dataset["Absenteeism time in hours"])

sns.jointplot(y='Transportation expense',x='Month of absence',data=dataset,kind='scatter',color='green')

plt.figure(figsize=(14,7))

sns.lmplot(x='Age',y='Absenteeism time in hours',data=dataset,hue='Day of the week',height=4,aspect=3)

plt.figure(figsize=(12,6))

dataset[dataset['Son']==0]['Absenteeism time in hours'].plot.hist(bins=30)

g = sns.FacetGrid(data=dataset,col='Son')

g.map(plt.hist,'Absenteeism time in hours')

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

sns.displot(dataset['Reason for absence'])

#count of entries in column

dataset[dataset['Absenteeism time in hours']==0].count()

dataset.head(740)

#dropping unecessary columns
```

```
dataset.drop(['Work load Average/day '], axis=1, inplace=True)

dataset.head()

#dividing dataset inorder to test and train the model


X = dataset.iloc[:,:20]

Y = dataset.iloc[:,19:]


#Dividing data inorder to train and predict

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test= train_test_split(X, Y, test_size= 0.2,shuffle=False)

from sklearn.ensemble import RandomForestRegressor

#Passing parameters so that to train the model

regressor = RandomForestRegressor(n_estimators= 100, max_features=
'auto',max_depth=None,min_samples_leaf=1)

#predicting the Target Variable

regressor.fit(x_train, y_train)

y_pred = regressor.predict(x_test)

y_pred

dataset.tail(6)

from sklearn import metrics


print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))

print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))

print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test,y_pred)))

from sklearn.metrics import r2_score

r2_score(y_test,y_pred)
```