# Project 16 --- Absenteeism at work.

#### What is absenteeism?

Employee absenteeism is a frequent lack of attendance at work without valid cause. Generally, absenteeism is unplanned absences. It hampers regular working activity. Some common reasons for absenteeism are: workplace environment, family care, low morale, **illness, work stress & performance stress.**

**Here with the given dataset we will try to predict employee absenteeism.**

## Purpose:

This model will help the organization to predict the employee absenteeism ,so that the organization evaluates the cause of targeted high absenteeism employees. And comes out with targeted solution for each individual to eventually reduce the absenteeism time of employees . Resulting in better working of the organization and performance.

# Dataset taken from:

# <https://archive.ics.uci.edu/ml/datasets/Absenteeism+at+work>

Absenteeism\_at\_work.csv

Attribute Information:

1. Individual identification (ID)  
2. Reason for absence (ICD).  
 Absences attested by the International Code of Diseases (ICD) stratified into 21 categories (I to XXI) as follows:  
  
 I Certain infectious and parasitic diseases  
 II Neoplasms  
 III Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism  
 IV Endocrine, nutritional and metabolic diseases  
 V Mental and behavioural disorders  
 VI Diseases of the nervous system  
 VII Diseases of the eye and adnexa  
 VIII Diseases of the ear and mastoid process  
 IX Diseases of the circulatory system  
 X Diseases of the respiratory system  
 XI Diseases of the digestive system  
 XII Diseases of the skin and subcutaneous tissue  
 XIII Diseases of the musculoskeletal system and connective tissue  
 XIV Diseases of the genitourinary system  
 XV Pregnancy, childbirth and the puerperium  
 XVI Certain conditions originating in the perinatal period  
 XVII Congenital malformations, deformations and chromosomal abnormalities  
 XVIII Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified  
 XIX Injury, poisoning and certain other consequences of external causes  
 XX External causes of morbidity and mortality  
 XXI Factors influencing health status and contact with health services.  
  
 And 7 categories without (CID) patient follow-up (22), medical consultation (23), blood donation (24), laboratory examination (25), unjustified absence (26), physiotherapy (27), dental consultation (28).  
3. Month of absence  
4. Day of the week (Monday (2), Tuesday (3), Wednesday (4), Thursday (5), Friday (6))  
5. Seasons (summer (1), autumn (2), winter (3), spring (4))  
6. Transportation expense  
7. Distance from Residence to Work (kilometers)  
8. Service time  
9. Age  
10. Work load Average/day  
11. Hit target  
12. Disciplinary failure (yes=1; no=0)  
13. Education (high school (1), graduate (2), postgraduate (3), master and doctor (4))  
14. Son (number of children)  
15. Social drinker (yes=1; no=0)  
16. Social smoker (yes=1; no=0)  
17. Pet (number of pet)  
18. Weight  
19. Height  
20. Body mass index  
21. Absenteeism time in hours (target) --- How many hours an employee has been absent.

This project is done on Jupyter notebook i.e. Python based.

At first we import some basic libraries like pandas to load analysis and manipulation of data, numpy to perform a number of mathematical operations on arrays, seaborn and matplot for EDA visualisations.

*#Import libraries*

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**import** **seaborn** **as** **sns**

**import** **matplotlib.pyplot** **as** **plt**

%**matplotlib** inline

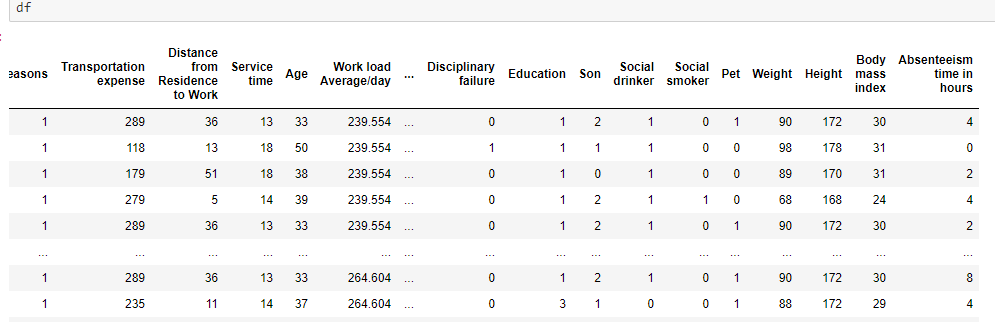
sns.set\_style('whitegrid')

#### **Load the data:**

*#load dataset*

df = pd.read\_csv('Absenteeism\_at\_work.csv' , sep = ';')

Import the ‘**Absenteeism\_data.csv’** with the help of pandas.



After loading the dataset we get a glimpse of data. We come to know that our target attribute is having discrete numerical data i.e. Absenteeism time in hours. We decide to apply regression algorithms on our dataset to get the predicted target value.

Studying the dataset:

*#basic insights*

df.info()

This gives the basic insights of the dataset like the shape of dataset, number of rows and columns, attribute names , datatype of attributes. Count of non null values in each columns, count of each datatype present in dataset, and finaly the memory size consumed by the dataset.

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 740 entries, 0 to 739

Data columns (total 21 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 ID 740 non-null int64

1 Reason for absence 740 non-null int64

2 Month of absence 740 non-null int64

3 Day of the week 740 non-null int64

4 Seasons 740 non-null int64

5 Transportation expense 740 non-null int64

6 Distance from Residence to Work 740 non-null int64

7 Service time 740 non-null int64

8 Age 740 non-null int64

9 Work load Average/day 740 non-null float64

10 Hit target 740 non-null int64

11 Disciplinary failure 740 non-null int64

12 Education 740 non-null int64

13 Son 740 non-null int64

14 Social drinker 740 non-null int64

15 Social smoker 740 non-null int64

16 Pet 740 non-null int64

17 Weight 740 non-null int64

18 Height 740 non-null int64

19 Body mass index 740 non-null int64

20 Absenteeism time in hours 740 non-null int64

dtypes: float64(1), int64(20)

memory usage: 121.5 KB

Now we check for null values in dataset.

*#Check null values*

df.isnull().sum()

We found no null values.

Let’s check for the statistical insights of the dataset.

*#statistical insights*

df.describe()

This gives count, min, max, mean, standard deviation & 25th 50th 75th values of each column of the dataset.

Now let’s check for unique values in each column.

*# Number of Unique values present in each variable*

df.nunique()

Till here we came to know a lot about our dataset like frequency of unique values, presence of null values, mean, standard deviation of each columns shape of dataset and data types of each columns.

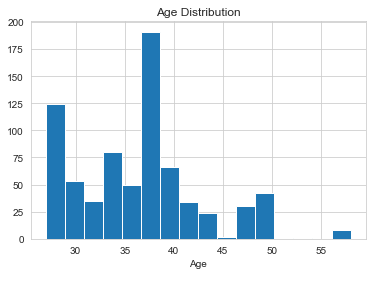
Now let’s do the EDA part i.e. inter-relating the columns to find pattern and meaning form the dataset with graphical visualization.

plt.hist(data=df, x='Age', bins='auto', label='Age')

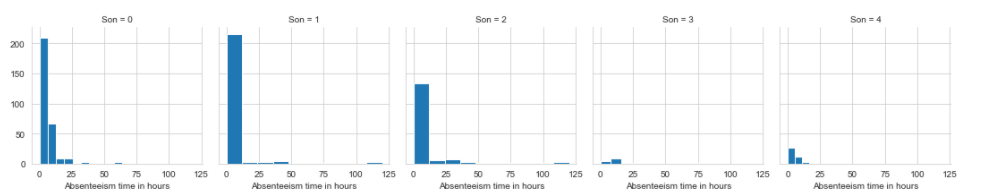
plt.xlabel('Age')

plt.title("Age Distribution")

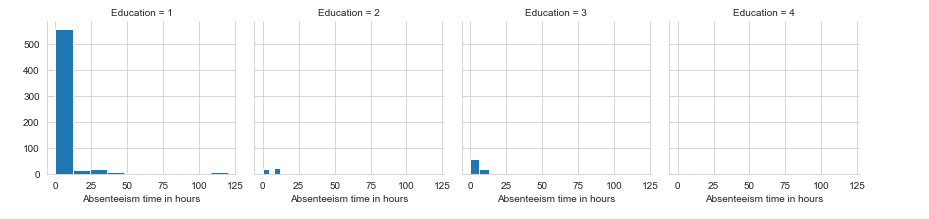
We see that the majority employee between age 35 to 40 have absenteeism recorded.



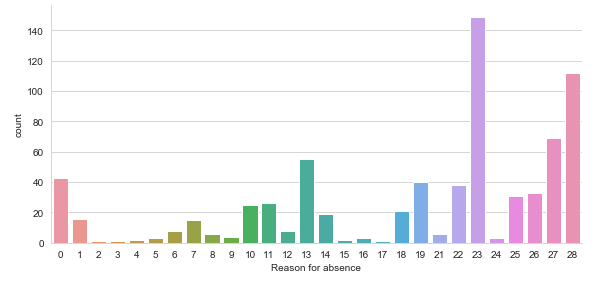
When we compare sons attribute with our target attribute we see with increase in child absenteeism increases.



When we compare education attribute with our target attribute we see with increase in education absenteeism decreases.

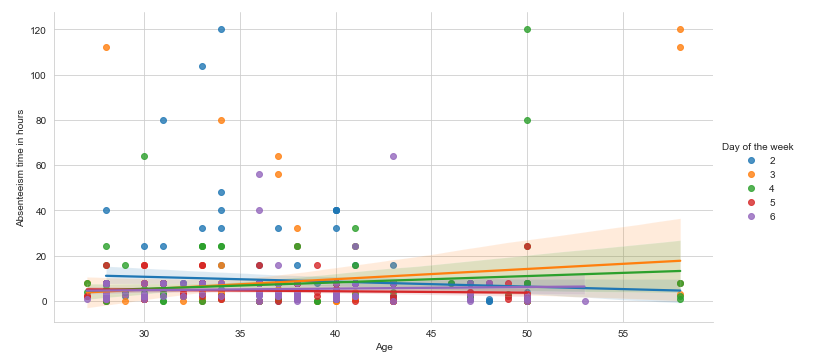


**We see** medical consultation, physiotherapy & dental consultation as the top three reasons for absenteeism from below graph.

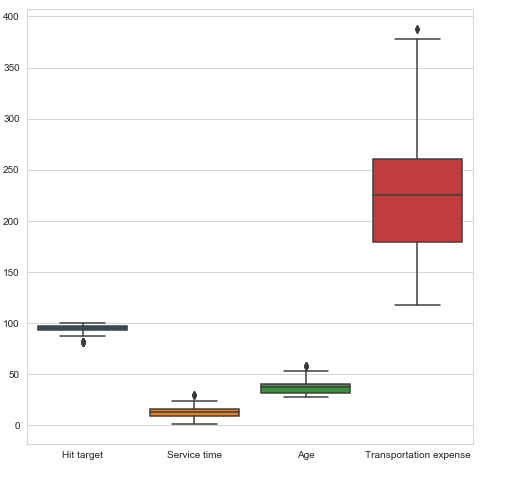
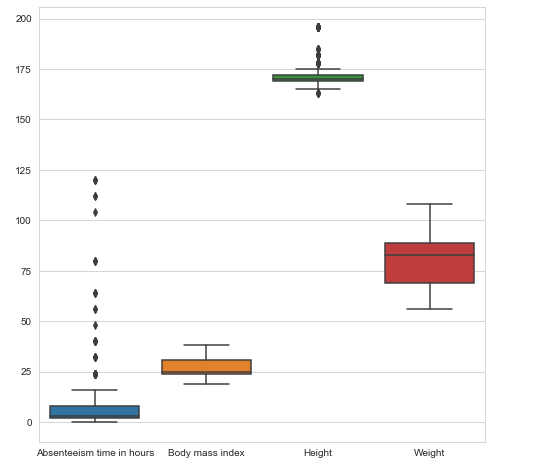


We see from below graph that most of the time, absence is present in 6th day of the week, so,

there are some correlation with not only age , but with day of the week as well.



From below plot we see there are outliers present in our dataset.

Let’s fix the outliers using KNN imputers and replacing all the outliers with maximum or minumum values.

*#Outlier Analysis*

**from** **sklearn.impute** **import** KNNImputer

**for** i **in** ['Distance from Residence to Work', 'Service time', 'Age', 'Work load Average/day ', 'Transportation expense',

'Hit target', 'Weight', 'Height', 'Body mass index', 'Absenteeism time in hours']:

*# Getting 75 and 25 percentile of variable "i"*

q75, q25 = np.percentile(df[i], [75,25])

*# Calculating Interquartile range*

iqr = q75 - q25

*# Calculating upper extream and lower extream*

minimum = q25 - (iqr\*1.5)

maximum = q75 + (iqr\*1.5)

*# Replacing all the outliers value to NA*

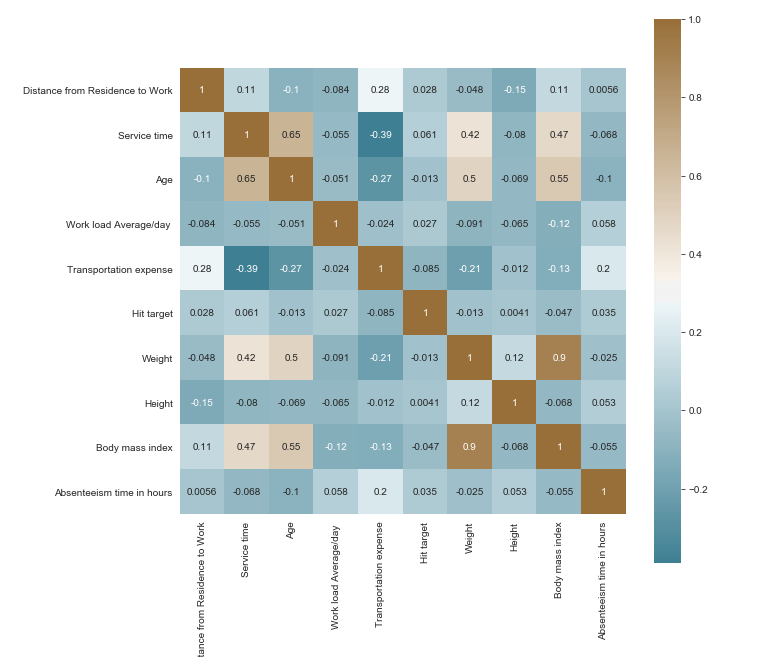
df.loc[df[i]< minimum,i] = np.nan

df.loc[df[i]> maximum,i] = np.nan

*# Impute missing values with KNN*

df = pd.DataFrame(KNNImputer(n\_neighbors=2).fit\_transform(df), columns = df.columns)

Let’s check for multicollinearity using corelation graph.



Now we do some feature scaling:

As the attributes are not normalized.

*Normalization of continuous variables*

**for** i **in** ['Distance from Residence to Work', 'Service time', 'Age',

'Work load Average/day ', 'Transportation expense',

'Hit target', 'Height', 'Body mass index',

'Absenteeism time in hours']:

**if** i == 'Absenteeism time in hours':

**continue**

df[i] = (df[i] - df[i].min())/(df[i].max()-df[i].min())

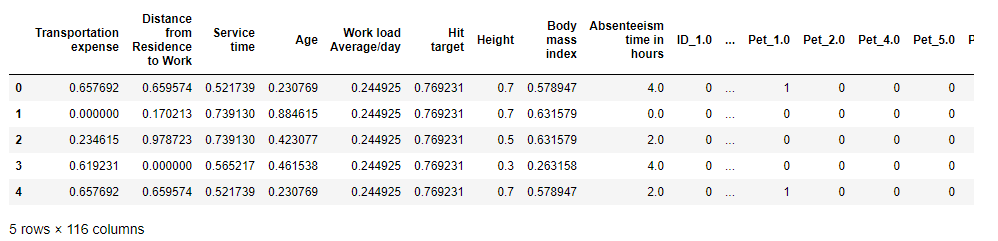
Now create dummy variables of factor variables

df = pd.get\_dummies(data = df, columns = ['ID','Reason for absence',

'Month of absence','Day of the week','Seasons','Disciplinary failure',

'Education', 'Social drinker','Social smoker', 'Pet', 'Son'])

So, now our data is ready to be used by regression models.



### **Split Data into train and test**

### Divide our data into train and test and build the model on train data set.

*#Splitting data into train and test data*

**from** **sklearn.model\_selection** **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split( df.iloc[:, df.columns != 'Absenteeism time in hours'], df.iloc[:, 8], test\_size = 0.20, random\_state = 1)

### **Apply Algorithm:**

As per our scenario we are going to use regression in our case.

* Let’s use Decision Tree

*# Importing libraries for Decision Tree*

**from** **sklearn.tree** **import** DecisionTreeRegressor

*#Build decsion tree using DecisionTreeRegressor*

dt\_model = DecisionTreeRegressor(random\_state = 1).fit(X\_train,y\_train)

*#Perdict for test cases*

dt\_predictions = dt\_model.predict(X\_test)

*#Create data frame for actual and predicted values*

df\_dt = pd.DataFrame({'actual': y\_test, 'pred': dt\_predictions})

print(df\_dt.head())

*#Define function to calculate RMSE*

**def** RMSE(y\_actual,y\_predicted):

rmse = np.sqrt(mean\_squared\_error(y\_actual,y\_predicted))

**return** rmse

*#Calculate RMSE and R-squared value*

print("Root Mean Squared Error: "+str(RMSE(y\_test, dt\_predictions)))

print("R^2 Score(coefficient of determination) = "+str(r2\_score(y\_test, dt\_predictions)))

We observe the result to be not satisfactory R2 square is in negative and RMSE is high upto 3.9

* Now let’s go for Random Forest

*# Importing libraries for Random Forest*

**from** **sklearn.ensemble** **import** RandomForestRegressor

*#Build random forest using RandomForestRegressor*

rf\_model = RandomForestRegressor(n\_estimators = 500, random\_state = 1).fit(X\_train,y\_train)

*#Perdict for test cases*

rf\_predictions = rf\_model.predict(X\_test)

*#Create data frame for actual and predicted values*

df\_rf = pd.DataFrame({'actual': y\_test, 'pred': rf\_predictions})

print(df\_rf.head())

*#Calculate RMSE and R-squared value*

print("Root Mean Squared Error: "+str(RMSE(y\_test, rf\_predictions)))

print("R^2 Score(coefficient of determination) = "+str(r2\_score(y\_test, rf\_predictions)))

In this case the R2 square and RMSE has improved than decision tree but not acceptable. Still RMSE is high and R2 score is low.

* Now let’s use Linear Regression

*# Importing libraries for Linear Regression*

**from** **sklearn.linear\_model** **import** LinearRegression

*#Train the model*

lr\_model = LinearRegression().fit(X\_train , y\_train)

*#Perdict for test cases*

lr\_predictions = lr\_model.predict(X\_test)

*#Create data frame for actual and predicted values*

df\_lr = pd.DataFrame({'actual': y\_test, 'pred': lr\_predictions})

print(df\_lr.head())

*#Calculate RMSE and R-squared value*

print("Root Mean Squared Error: "+str(RMSE(y\_test, lr\_predictions)))

print("R^2 Score(coefficient of determination) = "+str(r2\_score(y\_test, lr\_predictions)))

In this case our RMSE and R2 score went to worst values.

As all the above three regression models didn’t work well ,

let’s use Dimension Reduction.

Dimension Reduction using PCA:

Let’s check the shape of our dataset that we used in our above three algorithms.

df.shape

(740, 116)

We see the rows as 740 and columns as 116

We apply PCA on our input data

*#Import library for PCA*

**from** **sklearn.decomposition** **import** PCA

*#Converting data to numpy array*

X = df.values

*#Data has 116 variables so no of components of PCA = 115*

pca = PCA(n\_components=115)

pca.fit(X)

*#Proportion of variance explained*

var= pca.explained\_variance\_ratio\_

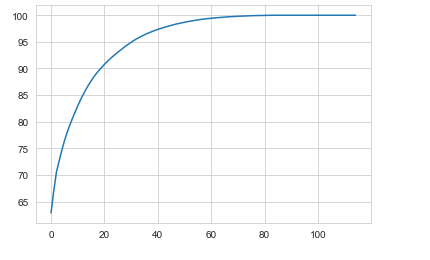
*#Cumulative scree plot*

var1=np.cumsum(np.round(pca.explained\_variance\_ratio\_, decimals=4)\*100)

*#Draw the plot*

plt.plot(var1)

plt.show()



As per the above plot we can conclude that we can reduce our components to 45 from 115.

Now let’s apply and see our result of models with PCA curated data.

*#Selecting 45 components since it explains almost 95+ % data variance*

pca = PCA(n\_components=45)

*#Fitting the selected components to the data*

pca.fit(X)

*#Splitting data into train and test data*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,target, test\_size=0.2, random\_state = 1)

Now Let’s apply the above three algorithms again on this dimension reduced dataset.

* Using DecisionTreeRegressor:

*#Build decsion tree using DecisionTreeRegressor*

dt\_model = DecisionTreeRegressor(random\_state = 1).fit(X\_train,y\_train)

*#Perdict for test cases*

dt\_predictions = dt\_model.predict(X\_test)

*#Create data frame for actual and predicted values*

df\_dt = pd.DataFrame({'actual': y\_test, 'pred': dt\_predictions})

print(df\_dt.head())

*#Calculate RMSE and R-squared value*

print("Root Mean Squared Error: "+str(RMSE(y\_test, dt\_predictions)))

print("R^2 Score(coefficient of determination) = "+str(r2\_score(y\_test, dt\_predictions)))

This time our R2 score has increased to 0.999 that is perfect and RMSE has also reduced to 0.1

* Let’s see the result of RandomForestRegressor:

*#Build random forest using RandomForestRegressor*

rf\_model = RandomForestRegressor(n\_estimators = 500, random\_state = 1).fit(X\_train,y\_train)

*#Perdict for test cases*

rf\_predictions = rf\_model.predict(X\_test)

*#Create data frame for actual and predicted values*

df\_rf = pd.DataFrame({'actual': y\_test, 'pred': rf\_predictions})

print(df\_rf.head())

*#Calculate RMSE and R-squared value*

print("Root Mean Squared Error: "+str(RMSE(y\_test, rf\_predictions)))

print("R^2 Score(coefficient of determination) = "+str(r2\_score(y\_test, rf\_predictions)))

Our result for RMSE and R2 score is almost same as decision tree above with slight improvement.

* Let’s check on Linear Regression:

*# Importing libraries for Linear Regression*

**from** **sklearn.linear\_model** **import** LinearRegression

*#Train the model*

lr\_model = LinearRegression().fit(X\_train , y\_train)

*#Perdict for test cases*

lr\_predictions = lr\_model.predict(X\_test)

*#Create data frame for actual and predicted values*

df\_lr = pd.DataFrame({'actual': y\_test, 'pred': lr\_predictions})

print(df\_lr.head())

*#Calculate RMSE and R-squared value*

print("Root Mean Squared Error: "+str(RMSE(y\_test, lr\_predictions)))

print("R^2 Score(coefficient of determination) = "+str(r2\_score(y\_test, lr\_predictions)))

This time result has improved to satisfaction.

actual pred

681 8.0 8.0

257 2.0 2.0

527 8.0 8.0

637 8.0 8.0

429 4.0 4.0

Root Mean Squared Error: 0.0015917054330183292

R^2 Score(coefficient of determination) = 0.999999788952107

We see Linear Regression to perform the best among other models.

Need to save the model which we have prepared so far.

To do that we need to pickle the model.

*#save the best model.*

**import** **pickle**

filename='Absenteeism.pkl'

M=open(filename,'wb')

pickle.dump(lr\_model,M)

M.close()

**To save your .Ipnyb file in form of executable, save the same as .py file**.

### ****To check model performance on totally new data set with same features.****

Now we have a totally new data set which has same feature as per previous data set but contain different values.

**Note –** To do that your executable file ‘**model’ and ‘.py’ file should be in same folder.**

**Check out my work @**

**https://nbviewer.jupyter.org/github/HemantPatar/Project-DYnamics-M20/blob/main/Project%2016%20%28%20Absenteeism%20Dataset%29.ipynb**

**Thank you ☺**