Employee Absenteeism

Abhishek Hupele

24 July 2018

Contents

Introduction

1.1 Problem Statement

XYZ is a courier company. As we appreciate that human capital plays an important role in collection, transportation and delivery. The company is passing through genuine issue of Absenteeism. The company has shared its dataset and requested to have an answer on the following areas:

1. What changes company should bring to reduce the number of absenteeism?
2. How much losses every month can we project in 2011 if same trend of absenteeism continues?

1.2 Data

Dataset Characteristics: Timeseries Multivariant
Number of Attributes: 21
Missing Values : Yes

Numerical Variable :
‘Transportation expense', Distance from Residence to Work',‘Service time', 'Age', 'Work load Average/day ', 'Hit target','Son','Pet', 'Weight', 'Height', 'Body mass index','Absenteeism time in hours’

Categorical Variable :
‘ID', 'Reason for absence', 'Month of absence', 'Day of the week','Seasons', 'Disciplinary failure', 'Education','Social drinker','Social smoker’

Target Variable :
'Absenteeism time in hours’

It is regression problem, and we need to predict the features which are mostly contributing to absenteeism and suggest the possible changes to reduce absenteeism. Further, we need to predict the absenteeism expected in the same scenario.

Given below is a sample of the data set that we are using to predict the absenteeism of employees :

ID	Reason for absence	Month of absence	Day of the week	Seasons	Transportation expense	Distance from Residence to Work	Service time	Age	Work load Average /day
11	26	7	3	1	289	36	13	33	2,39,554
36	0	7	3	1	118	13	18	50	2,39,554
3	23	7	4	1	179	51	18	38	2,39,554
7	7	7	5	1	279	5	14	39	2,39,554
11	23	7	5	1	289	36	13	33	2,39,554
3	23	7	6	1	179	51	18	38	2,39,554
10	22	7	6	1		52	3	28	2,39,554

Hit target	Disciplinary failure	Education	Son	Social drinker	Social smoker	Pet	Weight	Height	Body mass index	Absenteeism time in hours
97	0	1	2	1	0	1	90	172	30	4
97	1	1	1	1	0	0	98	178	31	0
97	0	1	0	1	0	0	89	170	31	2
97	0	1	2	1	1	0	68	168	24	4
97	0	1	2	1	0	1	90	172	30	2
97	0	1	0	1	0	0	89	170	31	

Methodology

2.1 Exploratory Data Analysis

In EDA, we analyse dat set to summarize their main characteristics, often with visual methods. Explore the data, and possibly formulate hypotheses that could lead to new data collection and experiments.
Below showcasing some of the methods used for EDA -

a) Data Summary :

The data has dimensions (740 x 21), 740 observations and 21 attributes .

Numeric Values :

'Transportation expense', 'Distance from Residence to Work','Service time', 'Age', 'Work load Average/day ', 'Hit target','Son','Pet', 'Weight', 'Height', 'Body mass index','Absenteeism time in hours'

	Transportation expense	Distance from Residence to Work	Service time	Age	Work load Average/day	Hit target	Son	Pet	Weight	Height	Body mass index	Absenteeis time in hou
count	733.000000	737.000000	737.000000	737.000000	730.000000	734.000000	734.000000	738.000000	739.000000	726.000000	709.000000	718.0000
mean	221.035471	29.667571	12.565807	36.449118	271188.860274	94.587193	1.017711	0.746612	79.063599	172.152893	26.684062	6.9777
std	66.954179	14.848124	4.389813	6.480148	38981.880873	3.792705	1.094928	1.319726	12.868630	6.081065	4.292819	13.4769
min	118.000000	5.000000	1.000000	27.000000	205917.000000	81.000000	0.000000	0.000000	56.000000	163.000000	19.000000	0.0000
25%	179.000000	16.000000	9.000000	31.000000	244387.000000	93.000000	0.000000	0.000000	69.000000	169.000000	24.000000	2.0000
50%	225.000000	26.000000	13.000000	37.000000	264249.000000	95.000000	1.000000	0.000000	83.000000	170.000000	25.000000	3.0000
75%	260.000000	50.000000	16.000000	40.000000	284853.000000	97.000000	2.000000	1.000000	89.000000	172.000000	31.000000	8.0000
max	388.000000	52.000000	29.000000	58.000000	378884.000000	100.000000	4.000000	8.000000	108.000000	196.000000	38.000000	120.0000

Categorical Variable :

'ID', 'Reason for absence', 'Month of absence', 'Day of the week','Seasons', 'Disciplinary failure', 'Education','Social drinker','Social smoker'

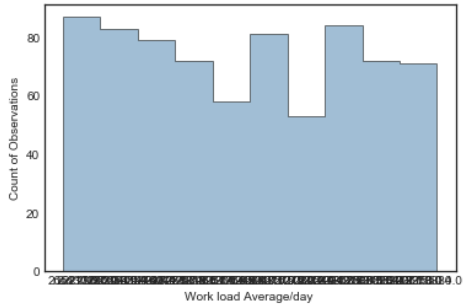
	ID	Reason for absence	Month of absence	Day of the week	Seasons	Disciplinary failure	Education	Social drinker	Social smoker
count	740.000000	737.000000	739.000000	740.000000	740.000000	734.000000	730.000000	737.000000	736.000000
mean	18.017568	19.188602	6.319350	3.914865	2.544595	0.053134	1.295890	0.567164	0.073370
std	11.021247	8.437493	3.435948	1.421675	1.111831	0.224453	0.676965	0.495805	0.260919
min	1.000000	0.000000	0.000000	2.000000	1.000000	0.000000	1.000000	0.000000	0.000000
25%	9.000000	13.000000	3.000000	3.000000	2.000000	0.000000	1.000000	0.000000	0.000000
50%	18.000000	23.000000	6.000000	4.000000	3.000000	0.000000	1.000000	1.000000	0.000000
75%	28.000000	26.000000	9.000000	5.000000	4.000000	0.000000	1.000000	1.000000	0.000000
max	36.000000	28.000000	12.000000	6.000000	4.000000	1.000000	4.000000	1.000000	1.000000

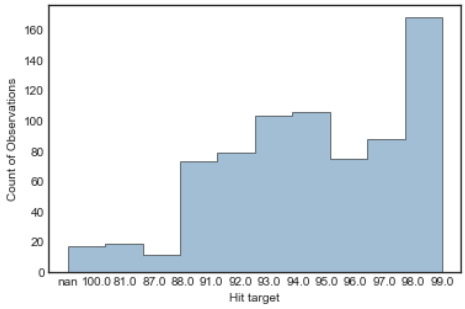
From ‘count’ statistics t is clear that there are some missing values in the data , ‘percentile’ values and ‘min’, ‘max’ values indicates outliers are present -outliers are important consideration in linear regression models as it can distort the result.

Also, note time related variables such as month, day of week and seasons. We can check for any cyclic seasonal, trend patterns with month of absence, days of week, seasons

b) Histograms :

Plotting data for numeric values gives a sense of distribution.



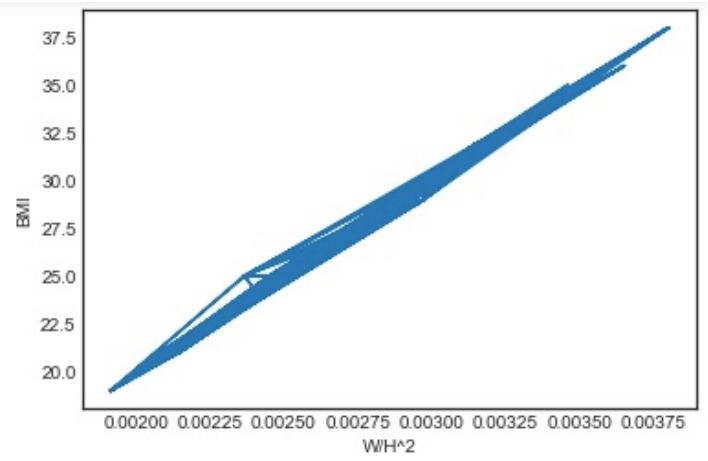


c) Frequency Distribution :

Day of the week		Disciplinary Failure		Seasons	
Values	Frequency	Values	Frequency	Values	Frequency
2	161	0	695	1	170
3	154	1	39	2	192
4	156			3	183
5	125			4	195
6	144				
Social drinker		Education		Social smoker	
Values	Frequency	Values	Frequency	Values	Frequency
0	319	1	601	0	682
1	418	2	46	1	54

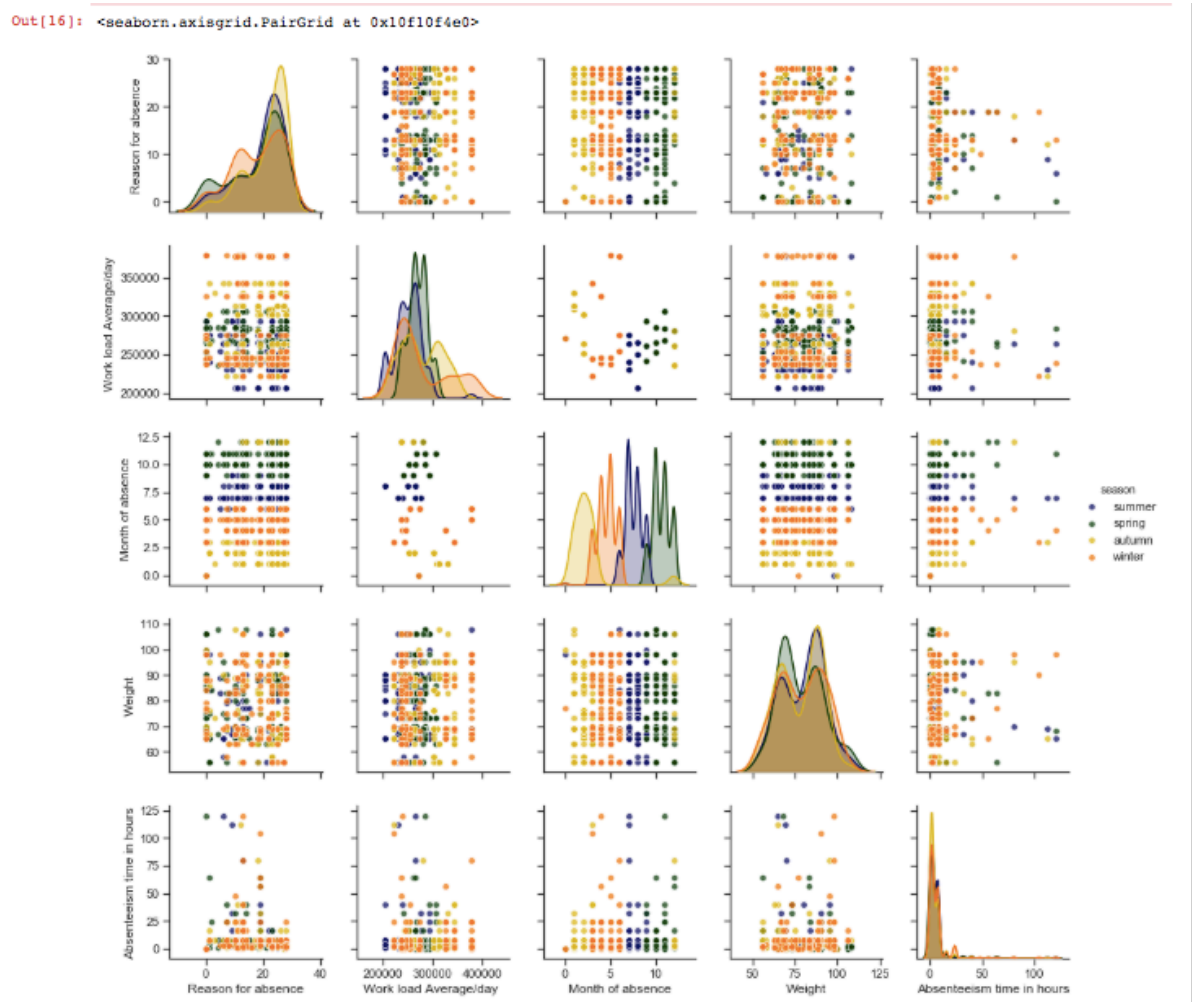
c) Scatter Plots :

Constructing scatter plots for variables helps us to see any relation.



We know that Body Mass Index is a simple calculation using a person's height and weight. The formula is BMI = kg/m^2 where kg is a person's 'Weight' in kilograms and m^2 is their 'Height' in metres squared. Same is evident from scatter plot plotted using this formula. Below is the plot showing relation between 'Body mass index' , 'Weight', 'Height' using ratio - BMI = kg/m^2 , showing linear relationship.

We can make one more exploratory plot, the pairplot, to visualize the relationships between variables. This plots all the variables against each other in scatterplots allowing us to inspect correlations between features.



d) Table Summary :

Absenteeism time in hours					
Day of the week	2	3	4	5	6
Month of absence					
0.0	NaN	0.0	0.0	NaN	0.0
1.0	88.0	45.0	36.0	42.0	11.0
2.0	47.0	51.0	68.0	62.0	66.0
3.0	262.0	213.0	120.0	67.0	87.0
4.0	61.0	31.0	234.0	36.0	120.0
5.0	118.0	22.0	122.0	26.0	104.0
6.0	120.0	47.0	127.0	65.0	44.0
7.0	236.0	325.0	53.0	47.0	63.0
8.0	86.0	72.0	15.0	66.0	33.0
9.0	117.0	47.0	81.0	8.0	31.0
10.0	75.0	51.0	108.0	71.0	35.0
11.0	127.0	159.0	86.0	40.0	51.0
12.0	32.0	243.0	41.0	23.0	43

2.2 Pre Processing

2.2.1 Assigning Levels/Categories to Categorical Variables

We observe that the variables are numeric in data set. But, we have also seen categorical variables in data as mentioned in 'Data Summary' represented by the numbers. Therefore, we need to convert these variables into categorical.

2.2.2 Missing Value Analyses

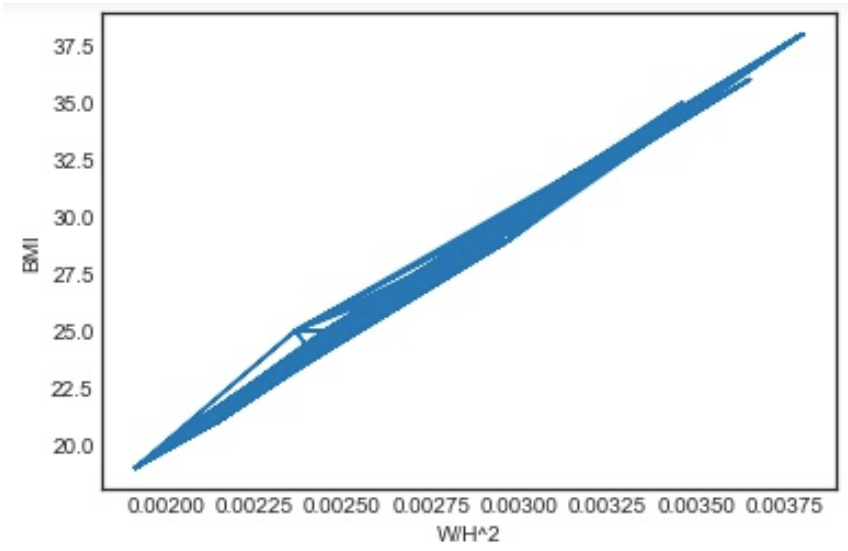
We have seen indication of missing values present in EDA section with 'count' values. Let's see the extent of missing values.

	miss_value	miss_percentage
Body mass index	31	4.19
Absenteeism time in hours	22	2.97
Height	14	1.89
Work load Average/day	10	1.35
Transportation expense	7	0.95
Son	6	0.81
Hit target	6	0.81
Service time	3	0.41
Age	3	0.41
Distance from Residence to Work	3	0.41
Pet	2	0.27
Weight	1	0.14
Reason for absence	0	0.00
Disciplinary failure	0	0.00
Education	0	0.00
Seasons	0	0.00
Social drinker	0	0.00

Social smoker	0	0.00
Day of the week	0	0.00
Month of absence	0	0.00

We know that **Body Mass** Index is a simple **calculation** using a person's height and weight. The **formula** is **BMI** = kg/m^2 where kg is a person's 'Weight' in kilograms and m^2 is their 'Height' in metres squared. Same is evident from scatter plot plotted using this formula. We can thus use it to predict the missing values. Also the mean error is low for predicting this value using the formula **BMI** = kg/m^2 .

error_pred_BMI = —0.006686090091033748
error_pred_Height = -0.03809632194196376
error_pred_Weght = 0.014961058344639218



Observe that 'Body mass index' , 'Height' and 'Weight' comprises of 46 of the missing observations. Which can be imputed with greater accuracy using BMI ratio as stated.

Thus we predict missing clues for the 'Body mass index' , 'Height' and 'Weight' using **BMI** = kg/m^2.

New Missing Value data is

	miss_value	miss_percentage
Absenteeism time in hours	22	2.97
Work load Average/day	10	1.35
Education	10	1.35
Transportation expense	7	0.95
Son	6	0.81
Disciplinary failure	6	0.81
Hit target	6	0.81
Social smoker	4	0.54
Service time	3	0.41
Age	3	0.41
Distance from Residence to Work	3	0.41
Reason for absence	3	0.41
Social drinker	3	0.41
Pet	2	0.27
Height	2	0.27
Body mass index	2	0.27
Month of absence	1	0.14
Seasons	0	0.00
Day of the week	0	0.00
Weight	0	0.00

22 observations are for ‘Absenteeism time in hours’, as it is target variable and we are supposed to predict it ,imputing it may pollute the model, the, dropping these 22 observations.

Missing Percentage is less than 1.35% and so we choose to impute rest of the data. Using KNN imputation to impute the missing values as it gives closest value compared to mean and median methods.

2.2.3 Outlier Analysis

An outlier is a pattern which is dissimilar with respect to the rest of the patterns in the data set.
If we remove outliers by IQR method , observations are reduce to 516 which is 30% less than original 740 observations.
Therefore, we will go for imputing the outliers. We have replaced values greater than 0.99 quantile with value of 0.99 quantile and replaced values less than 0.01 quantile with value of 0.01 quantile.

2.2.4 Feature Scaling

In order to reduce unwanted variation either within or between variables, we perform either normalisation or standardisation. As all the variables are not normally distributed, we will bring the continuous variables within range(0,1) by performing normalisation.

$$z = \frac{\mathrm{x} - min(x)}{[\max(x) - min(x)]}$$

Where x is an original value, z is the normalised value

Used **min-max scaling** for each independent variable column and have brought all of them in to same range(0,1), **excluded target variable.**

2.2.5 Dummy Variable/One hot encoding of factor variables

For categorical variables where no such ordinal relationship exists, the integer encoding is not enough.
In fact, using this encoding and allowing the model to assume a natural ordering between categories may result in poor performance or unexpected results (predictions halfway between categories).
In this case, a one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

2.2.6 Feature Engineering:

Feature engineering is the process of using **domain knowledge** of the data to create features that make machine learning algorithms work. If feature engineering is

done correctly, it increases the predictive power of machine learning algorithms by creating features from raw data that help facilitate the machine learning process. Feature Engineering is an art.
Steps which are involved while solving any problem in machine learning are as follows:

- Gathering data.
- Cleaning data.
- Feature engineering.
- Defining model.
- Training, testing model and predicting the output.

Feature engineering is the most important art in machine learning which creates the huge difference between a good model and a bad model. Let's see what feature engineering covers.
Suppose, we are given a data "flight date time vs status". Then, given the date-time data, we have to predict the status of the flight.

	Date_Time_Combined	Status
0	2018-02-14 20:40	Delayed
1	2018-02-15 10:30	On Time
2	2018-02-14 07:40	On Time
3	2018-02-15 18:10	Delayed
4	2018-02-14 10:20	On Time

Flight Date Time Data

As the status of the flight depends on the hour of the day, not on the date-time. We will create the new feature "Hour_Of_Day". Using the "Hour_Of_Day" feature, the machine will learn better as this feature is directly related to the status of the flight.

	Hour_Of_Day	Status
0	20	Delayed
1	10	On Time
2	7	On Time
3	18	Delayed
4	10	On Time

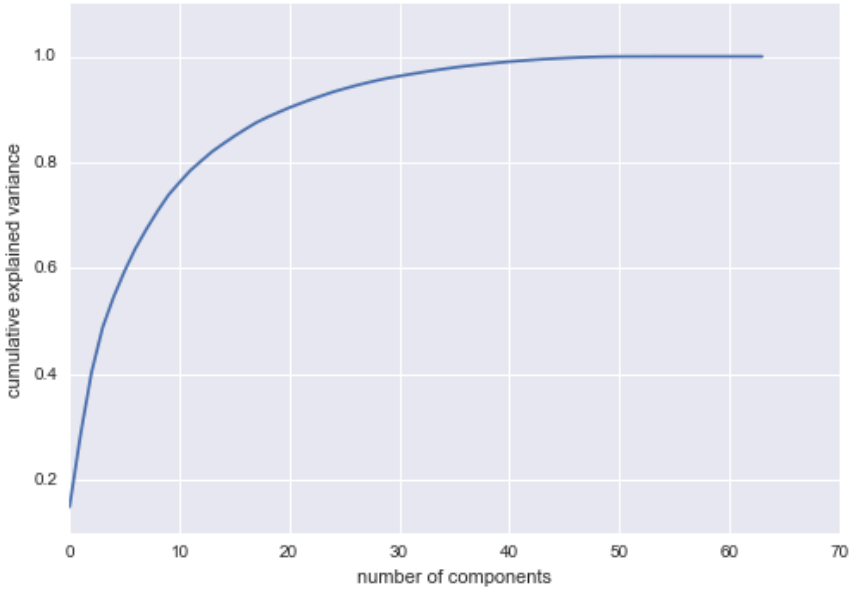
Flight Hour Of Day Data

Here, creating the new feature “Hour_Of_Day” is the feature engineering.

Using :
a) Boruta package in R
b) SeleckKBest package in Python

Principal component analysis :

It is a fast and flexible unsupervised method for dimensionality reduction in data, Using PCA for dimensionality reduction involves zeroing out one or more of the smallest principal components, resulting in a lower-dimensional projection of the data that preserves the maximal data variance.
A vital part of using PCA in practice is the ability to estimate how many components are needed to describe the data. This can be determined by looking at the cumulative *explained variance ratio* as a function of the number of components:



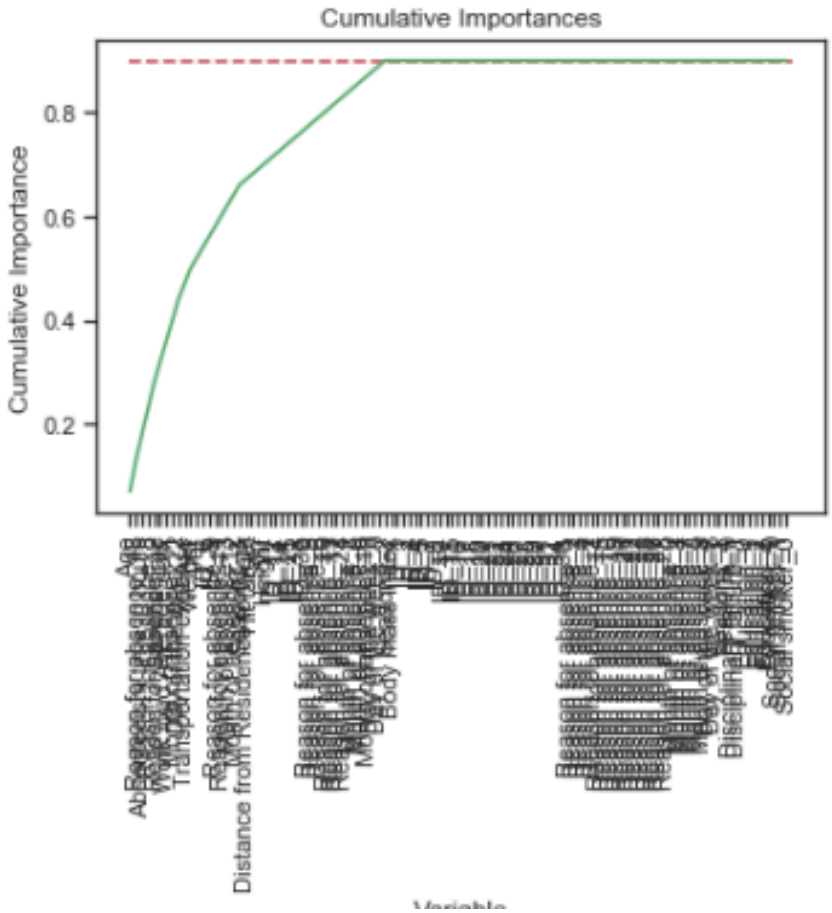
Boruta is a feature selection algorithm. Precisely, it works as a wrapper algorithm around Random Forest. This package derive its name from a demon in Slavic mythology who dwelled in pine forests.
We know that feature selection is a crucial step in predictive modeling. This technique achieves supreme importance when a data set comprised of several variables is given for model building.
Boruta can be your algorithm of choice to deal with such data sets. Particularly when one is interested in understanding the mechanisms related to the variable of interest, rather than just building a black box predictive model with good prediction accuracy.

How does it work?
Below is the step wise working of boruta algorithm:

1. Firstly, it adds randomness to the given data set by creating shuffled copies of all features (which are called shadow features).
2. Then, it trains a random forest classifier on the extended data set and applies a feature importance measure (the default is Mean Decrease Accuracy) to evaluate the importance of each feature where higher means more important.
3. At every iteration, it checks whether a real feature has a higher importance than the best of its shadow features (i.e. whether the feature has a higher Z score than the maximum Z score of its shadow features) and constantly removes features which are deemed highly unimportant.
4. Finally, the algorithm stops either when all features gets confirmed or rejected or it reaches a specified limit of random forest runs.

We can also make a cumulative importance graph that shows the contribution to the overall importance of each additional variable. The dashed line is drawn at 95% of total importance accounted for.

```
Number of features for 90% importance: 43
```

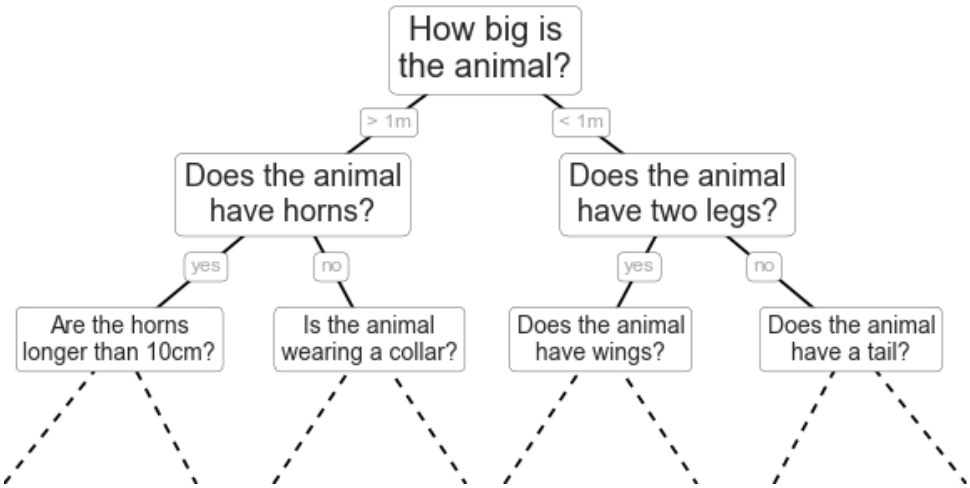


These stats definitely prove that some variables are much more important to our problem than others! First, let's make a quick graph to represent the relative differences in feature importances.

2.4. Model Selection

(i) Decision Tree , Random Forest :

Random forests are an example of an *ensemble learner* built on decision trees. For this reason we'll start by discussing decision trees themselves.
Decision trees are extremely intuitive ways to classify or label objects: you simply ask a series of questions designed to zero-in on the classification. For example, if you wanted to build a decision tree to classify an animal you come across while on a hike, you might construct the one shown here:



The binary splitting makes this extremely efficient: in a well-constructed tree, each question will cut the number of options by approximately half, very quickly narrowing the options even among a large number of classes. The trick, of course, comes in deciding which questions to ask at each step. In machine learning implementations of decision trees, the questions generally take the form of axis-aligned splits in the data: that is, each node in the tree splits the data into two groups using a cutoff value within one of the features.

A simple decision tree built on this data will iteratively split the data along one or the other axis according to some quantitative criterion, and at each level assign the label of the new region according to a majority vote of points within it.

This notion—that multiple overfitting estimators can be combined to reduce the effect of this overfitting—is what underlies an ensemble method called *bagging*. Bagging makes use of an ensemble (a grab bag, perhaps) of parallel estimators, each of which over-fits the data, and averages the results to find a better classification. An ensemble of randomized decision trees is known as a *random forest*.

(ii) Regression

Regression analysis is a form of predictive modelling technique which investigates the relationship between a dependent (target) and independent variable (s) (predictor). This technique is used for forecasting, time series modelling and finding the causal effect relationship between the variables. For example, relationship between rash driving and number of road accidents by a driver is best studied through regression.

Regression analysis is an important tool for modelling and analyzing data. Here, we fit a curve / line to the data points, in such a manner that the differences between the distances of data points from the curve or line is minimized. I'll explain this in more details in coming sections.

There are multiple benefits of using regression analysis. They are as follows:

1. It indicates the significant relationships between dependent variable and independent variable.

2. It indicates the strength of impact of multiple independent variables on a dependent variable.

There are various kinds of regression techniques available to make predictions. These techniques are mostly driven by three metrics (number of independent variables, type of dependent variables and shape of regression line).

1. Linear Regression

It is one of the most widely known modeling technique. Linear regression is usually among the first few topics which people pick while learning predictive modeling. In this technique, the dependent variable is continuous, independent variable(s) can be continuous or discrete, and nature of regression line is linear.
Linear Regression establishes a relationship between dependent variable (Y) and one or more independent variables (X) using a best fit straight line (also known as regression line).
It is represented by an equation Y=a+b*X + e, where a is intercept, b is slope of the line and e is error term. This equation can be used to predict the value of target variable based on given predictor variable(s).

2. Ridge Regression

Ridge Regression is a technique used when the data suffers from multicollinearity (independent variables are highly correlated). In multicollinearity, even though the least squares estimates (OLS) are unbiased, their variances are large which deviates the observed value far from the true value. By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors.
Above, we saw the equation for linear regression. Remember? It can be represented as:

$$y=a+ b*x$$

This equation also has an error term. The complete equation becomes:
y=a+b*x+e (error term), [error term is the value needed to correct for a prediction error between the observed and predicted value]
=> y=a+y= a+ b1x1+ b2x2+....+e, for multiple independent variables.
In a linear equation, prediction errors can be decomposed into two sub components. First is due to the biased and second is due to the variance. Prediction error can occur due to any one of these two or both components. Here, we'll discuss about the error caused due to variance.
Ridge regression solves the multicollinearity problem through shrinkage parameter λ (lambda). Look at the equation below.

$$= \underset{\beta \in \mathbf{R}^p}{\operatorname{argmin}} \underbrace{\|y - X\beta\|_2^2}_{\text{Loss}} + \lambda \underbrace{\|\beta\|_2^2}_{\text{Penalty}}$$

3. Lasso Regression

Similar to Ridge Regression, Lasso (Least Absolute Shrinkage and Selection Operator) also penalizes the absolute size of the regression coefficients. In addition, it is capable of reducing the variability and improving the accuracy of linear regression models. Look at the equation below:

$$= \underset{\beta \in \mathbf{R}^p}{\operatorname{argmin}} \underbrace{\|y - X\beta\|_2^2}_{\text{Loss}} + \lambda \underbrace{\|\beta\|_1}_{\text{Penalty}}$$

Lasso regression differs from ridge regression in a way that it uses absolute values in the penalty function, instead of squares. This leads to penalizing (or equivalently constraining the sum of the absolute values of the estimates) values which causes some of the parameter estimates to turn out exactly zero. Larger the penalty applied, further the estimates get shrunk towards absolute zero. This results to variable selection out of given n variables.

4.L**east-angle regression (LARS)**

It is an algorithm for fitting linear regression models to high-dimensional data, developed by Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani.[1]
Suppose we expect a response variable to be determined by a linear combination of a subset of potential covariates. Then the LARS algorithm provides a means of producing an estimate of which variables to include, as well as their coefficients.
Instead of giving a vector result, the LARS solution consists of a curve denoting the solution for each value of the L1 norm of the parameter vector. The algorithm is similar to forward stepwise regression, but instead of including variables at each step, the estimated parameters are increased in a direction equiangular to each one's correlations with the residual.

2.5 Model Evaluation

R :

Linear Regression :
500 samples
13 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 450, 449, 451, 450, 449, 450, ...
Resampling results:

RMSE	Rsquared	MAE
9.358796	0.2434889	4.861968

Tuning parameter 'intercept' was held constant at a value of TRUE

Random Forest :

550 samples
13 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 495, 494, 495, 495, 495, 494, ...
Resampling results across tuning parameters:

mtry	RMSE	Rsquared	MAE
2	9.353175	0.1987974	4.658850
7	9.757714	0.2041148	4.835090
13	9.830951	0.2001058	4.881805

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 2.

Decision Tree :
550 samples
13 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 495, 495, 495, 495, 495, 495, ...
Resampling results across tuning parameters:

cp	RMSE	Rsquared	MAE
0.02688929	9.823503	0.10806305	5.375228

```
 0.02784565   9.823503  0.10806305  5.375228
 0.06138268  10.272001  0.01678803  5.482391

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was cp = 0.02784565.
```

PCA _ Linear Regression :

```
PC23        9.528e-04  3.989e-05  2.389e+01  < 2e-16 ***
PC24       -3.837e-03  4.075e-05 -9.417e+01  < 2e-16 ***
PC25       -2.349e-03  4.138e-05 -5.676e+01  < 2e-16 ***
PC26        2.951e-03  4.301e-05  6.863e+01  < 2e-16 ***
PC27        3.398e-03  4.399e-05  7.725e+01  < 2e-16 ***
PC28       -3.299e-03  4.495e-05 -7.341e+01  < 2e-16 ***
PC29       -2.735e-03  4.630e-05 -5.908e+01  < 2e-16 ***
PC30       -1.626e-03  4.915e-05 -3.309e+01  < 2e-16 ***
PC31        2.251e-03  5.116e-05  4.400e+01  < 2e-16 ***
PC32        8.102e-04  5.308e-05  1.526e+01  < 2e-16 ***
PC33        2.124e-03  5.659e-05  3.753e+01  < 2e-16 ***
PC34       -9.060e-04  5.935e-05 -1.527e+01  < 2e-16 ***
PC35        5.861e-05  6.133e-05  9.560e-01   0.3397
PC36        7.388e-04  6.272e-05  1.178e+01  < 2e-16 ***
PC37        7.159e-05  6.412e-05  1.117e+00   0.2647
PC38       -3.926e-04  6.752e-05 -5.815e+00 1.07e-08 ***
PC39       -8.419e-04  7.036e-05 -1.196e+01  < 2e-16 ***
PC40       -1.828e-04  7.678e-05 -2.381e+00   0.0176 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0002389 on 509 degrees of freedom
Multiple R-squared:      1,  Adjusted R-squared:      1
F-statistic: 2.613e+10 on 40 and 509 DF,  p-value: < 2.2e-16

[1] 0.0002532975
```

PCA_Decision Tree

```
n= 550

node), split, n, deviance, yval
      * denotes terminal node

 1) root 550 5.966408e+04  6.6231330
   2) PC1>=-21.38557 529 1.093996e+04  4.8779600
     4) PC1>=0.3239187 341 5.907187e+02  2.2178680
       8) PC1>=3.975932 204 1.028063e+02  1.3032470
        16) PC1>=5.124953 113 2.145714e+01  0.7371172
          32) PC1>=6.341281 29 0.000000e+00  0.0000000 *
          33) PC1< 6.341281 84 2.603469e-01  0.9915981 *
        17) PC1< 5.124953 91 1.599352e-01  2.0062420 *
```

```
      9) PC1< 3.975932 137 6.314897e+01  3.5797870
       18) PC1>=2.998676 71 4.184048e-01  3.0129410 *
       19) PC1< 2.998676 66 1.537573e+01  4.1895750 *
     5) PC1< 0.3239187 188 3.559627e+03  9.7029140
      10) PC1>=-6.601393 160 8.810371e+00  8.0009230 *
      11) PC1< -6.601393 28 4.388571e+02 19.4285700
       22) PC1>=-13.38321 16 0.000000e+00 16.0000000 *
       23) PC1< -13.38321 12 0.000000e+00 24.0000000 *
   3) PC1< -21.38557 21 6.527667e+03 50.5848600
     6) PC1>=-41.38144 12 1.866667e+02 36.6666700 *
     7) PC1< -41.38144 9 9.169473e+02 69.1424600 *
[1] 2.369151
```

PCA_Lasso Regression

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was fraction = 0.9.

```
           Length Class      Mode
call          4   -none-     call
actions      41   -none-     list
allset       40   -none-     numeric
beta.pure  1640   -none-     numeric
vn           40   -none-     character
mu            1   -none-     numeric
normx        40   -none-     numeric
meanx        40   -none-     numeric
lambda        1   -none-     numeric
L1norm       41   -none-     numeric
penalty      41   -none-     numeric
df           41   -none-     numeric
Cp           41   -none-     numeric
sigma2        1   -none-     numeric
xNames       40   -none-     character
problemType   1   -none-     character
tuneValue     1   data.frame list
obsLevels     1   -none-     logical
param         0   -none-     list
[1] 1.341679
```

Python :

Decision Tree :
RMSE : 13.924707527894094
Model Score : -0.10208269812771098

Random Forest :
RMSE : 10.943016363130383
Model Score : -0.10208269812771098

Linear Regression :
RMSE : 10.047347044404601

Ridge Regression :
RMSE : 9.942875323147323
Model Score : 0.090161993699275 24

Lasso Regression :
RMSE : 10.244734307061808
Model Score : 0.03407927080390738

Lars Regression
RMSE : 10.066464784072709
Model Score : 0.0674029352486113

PCA_Decision Tree:
RMSE : 1.9521298140264163
Model Score : -0.10208269812771098

PCA_Random Forest :
RMSE : 1.296222028496432
Model Score : 0.9845368174631971

PCA_Linear Regression :
RMSE : 7.348232275761557

PCA_Ridge Regression :
RMSE : 0.0006038134689983856
Model Score : 0.9999999966445894

PCA_Lasso Regression :
RMSE : 0.2558915515924606

```
Model Score : 0.9993973682869337
```

PCA_Lars Regression :

```
RMSE :  0.0005605063708145348
Model Score : 0.9999999971086465
```

Best Fit : Model using PCA is giving the best results

2.6 What changes company should bring to reduce the number of absenteeism?

In order to find out the changes the company should make , we must find the features/variables that causes the absenteeism.
This problem is more about finding the causation factors of Absenteeism. Once we find them and the nature of relationship the company can suitably make changes to reduce Absenteeism.

Suggestions :

1. We have 51% of absenteeism for blood donations, medical checkups et.. It is recommended that Company should provide medical checkups once in 6 months for employees so that the absenteeism could be reduced. In-house medical consultant should be appointed for avoiding absenteeism due to non-serious medical issues .
2. 10% of employees are absent due to transportation expenses. So ,company must provide transportation facility and ask for little fee from employee so that this could be profitable for company help in reducing absenteeism. Company should provide necessary transportation for the people who stay(**>20 km**) far from workplace.
3. People who has low level educational qualification are being more absent.so company should talk to the workers regarding what the exact problem is.
4. Age group between 25 and 40 has almost 75% absent hours.Its the age between youth and middle adulthood age. Company should take up some activities for employee engagement and family events can be planned yearly/half-yearly.
5. People having **>70kg** weight has more absent hours.company should make sure that workers should not put on more weight as it leads to some serious issues.
6. Meal is important part of employees requirement, it affects health, weight, medical conditions, company can plan for canteen facilities at minimal rates and provide healthy food to improve well-being of employee health.

Distribution of factor variables who are absent (absenteeism hours >0)

Seasons :

Values	Frequency	% Freq
4	195	26.351351
2	192	25.945946
3	183	24.72973
1	170	22.972973

Disciplinary failure :

Values	Frequency	% Freq
0	701	94.729730
1	39	5.270270

Reason for absence :

People who has medical consultation ,dental consultation,physiotherapy,diseases of the genitourinary system contribute more than 50% absenteeism hours. So company should take necessary care to the workers who are suffering with above diseases to reduce number of absenteeism absenteeism

Values	Frequency	% Freq
23	149	20.135135
28	110	14.864865
27	71	9.594595
13	55	7.432432
0	43	5.810811
19	40	5.405405
22	37	5
26	33	4.459459
25	31	4.189189
11	26	3.513514
10	25	3.378378
18	21	2.837838
14	19	2.567568
1	16	2.162162
7	15	2.027027
12	8	1.081081
6	8	1.081081

Values	Frequency	% Freq
21	6	0.810811
8	6	0.810811
9	4	0.540541
24	3	0.405405
5	3	0.405405
16	3	0.405405
15	2	0.27027
4	2	0.27027
20	1	0.135135
3	1	0.135135
2	1	0.135135
17	1	0.135135

Day of the week :

Values	Frequency	% Freq
2	161	21.756757
4	156	21.081081
3	154	20.810811
6	144	19.459459
5	125	16.891892

Social smoker :

Values	Frequency	% Freq
0	686	92
1	54	7

Social drinker :

People who are social drinker have more number of absent hours

Values	Frequency	% Freq
1	419	56.621622
0	321	43.378378

Education :

People who have very low qualification level have more absent hours than others

Values	Frequency	% Freq
1	609	82.297297
3	79	10.675676
2	46	6.216216
4	4	0.540541
0	2	0.27027

Pet :

Values	Frequency	% Freq
0	460	62.162162
1	138	18.648649
2	96	12.972973
4	32	4.324324
6	8	1.081081
5	6	0.810811

Age :

Employees whose age is between 28 to 40 has more absent hours almost 75% belongs to this age group

Values	Frequency	% Freq
28	124	16.756757
38	113	15.27027
37	78	10.540541
40	59	7.972973

Values	Frequency	% Freq
33	51	6.891892
36	50	6.756757
30	46	6.216216
50	37	5
41	32	4.324324
34	29	3.918919
47	24	3.243243
43	24	3.243243
31	22	2.972973
32	13	1.756757
39	9	1.216216
56	8	1.081081
29	7	0.945946
48	6	0.810811
49	5	0.675676
46	2	0.27027
53	1	0.135135

Weight :

Employee with weight above 80 kgs has more absenteeism hours (>52%)

Values	Frequency	% Freq
89	113.00000	15.27027
69	85.000000	11.486486
65	61.000000	8.243243
83	55.000000	7.432432
56	46.000000	6.216216
90	40.000000	5.405405
73	37	5
98	35.00000	4.72973

Values	Frequency	% Freq
67	30.000000	4.054054
95	29.000000	3.918919
88	29.000000	3.918919
80	24.000000	3.243243
86	24.000000	3.243243
106	24.000000	3.243243
63	20.000000	2.702703
75	19.000000	2.567568
84	16.000000	2.162162
70	15.000000	2.027027
68	13.000000	1.756757
58	7.000000	0.945946
77	6.000000	0.810811
94	4.000000	0.540541
76	3.000000	0.405405
79	3.000000	0.405405
100	2.00000	0.27027

Distance from Residence to Work :

Employee who stay more than 20 km away from the company account for 66% foo absenteeism

Values	Frequency	% Freq
26	126	17.027027
51	120	16.216216
10	61	8.243243
25	55	7.432432
50	45	6.081081
36	40	5.405405
31	37	5
13	34	4.594595

Values	Frequency	% Freq
12	29	3.918919
16	26	3.513514
11	26	3.513514
52	24	3.243243
22	20	2.702703
20	19	2.567568
17	15	2.027027
29	14	1.891892
15	9	1.216216
14	9	1.216216
49	8	1.081081
42	7	0.945946
27	7	0.945946
48	5	0.675676
35	2	0.27027
45	1	0.135135
47	1	0.135135

Work load Average/day :

Values	Frequency	% Freq
222196	36	4.864865
264249	33	4.459459
237656	32	4.324324
343253	29	3.918919
265017	28	3.783784
284853	25	3.378378
268519	23	3.108108
284031	22	2.972973

Values	Frequency	% Freq
244387	22	2.972973
251818	21	2.837838
308593	21	2.837838
205917	21	2.837838
241476	21	2.837838
326452	20	2.702703
230290	20	2.702703
246288	20	2.702703
275089	19	2.567568
253957	19	2.567568
294217	19	2.567568
239554	19	2.567568
236629	19	2.567568
302585	18	2.432432
265615	18	2.432432
253465	17	2.297297
306345	16	2.162162
377550	16	2.162162
275312	16	2.162162
246074	16	2.162162
313532	15	2.027027
280549	15	2.027027
378884	15	2.027027
249797	15	2.027027
239409	13	1.756757
261306	13	1.756757
261756	12	1.621622
264604	12	1.621622
330061	11	1.486486
271219	3	0.405405

Values	Frequency	% Freq
243498	1	0.135135
244216	1	0.135135
318579	1	0.135135
308230	1	0.135135
253661	1	0.135135
254001	1	0.135135
254371	1	0.135135
258767	1	0.135135
261321	1	0.135135
262898	1	0.135135

Hit target :

Employees with achieving >90 hit target are accounting for 85% of absenteeism

Values	Frequency	% Freq
93	104	14.054054
99	102	13.783784
97	89	12.027027
92	80	10.810811
96	75	10.135135
95	74	10
98	66	8.918919
91	45	6.081081
94	35	4.72973
88	28	3.783784
81	19	2.567568
87	12	1.621622
100	11	1.486486

2.7 How much losses every month can we project in 2011 if same trend of absenteeism continues?

```
sum(pca.lm.prediction)
[1] 1315.677
> sum(pca.lasso.prediction)
[1] 1310.353
> sum(pca.dt.prediction)
[1] 1291.65
```

Total of 1300 absent hours predicted if it follows same trend in 2011.

Python Code

EDA_Code :

```
# coding: utf-8

# In[60]:


## Problem Statement
#1. What changes company should bring to reduce the number of absenteeism?
#2. How much losses every month can we project in 2011 if same trend of
#absenteeism continues?


# In[61]:


## Set Working Directory
import os
os.getcwd()
# os.chdir('Desktop/Project_2_Employee Absenteeism')


# In[62]:


# set path for python as need basis
#import sys
#sys.path.insert(0, '/local/python/lib/python2.7/site-packages/')


# In[63]:


##Import Libararies
import pandas as pd
import numpy as np
from fancyimpute import KNN
import matplotlib.pyplot as plt
import seaborn as sns


# In[64]:


##Import Data
data = pd.read_excel('Absenteeism_at_work_Project.xls')
data_org = data.copy()
```

```
col = data.columns
# Numerical and Categorical Variables
n_col = ['Transportation expense', 'Distance from Residence to Work','Service time',
        'Age', 'Work load Average/day ', 'Hit target','Son','Pet', 'Weight', 'Height',
        'Body mass index','Absenteeism time in hours']
c_col = ['ID', 'Reason for absence', 'Month of absence', 'Day of the week','Seasons',
'Disciplinary failure',
        'Education','Social drinker','Social smoker',]


# In[65]:


##Explanatory Data Analysis
# Data Summary
data.shape
data[n_col].describe()
data[c_col].describe()


# In[66]:


data_org


# In[67]:


#Histogram - Numerical Variables
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
plt.style.use('seaborn-white')
count = 1
for i in n_col:
    plt.figure()
            plt.hist(data[[i]],alpha = 0.5,histtype='stepfilled',color =
'steelblue',edgecolor='black')
    plt.xlabel(i)
    plt.ylabel('Count of Observations')


# In[68]:


#Histogram - Categorical Variables
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
plt.style.use('seaborn-white')
```

```
count = 1
for i in c_col:
    plt.figure()
            plt.hist(data[[i]],alpha = 0.5,histtype='stepfilled',color =
'steelblue',edgecolor='none')
    plt.xlabel(i)
    plt.ylabel('Count of Observations')


# In[69]:


d = data.pivot_table(index='Month of absence',aggfunc={'Absenteeism time in
hours':sum})
#data.groupby(['Month of absence', 'Day of the week'])['Absenteeism time in
hours'].aggregate('sum').unstack()
d = d.reset_index()
plt.plot(d['Month of absence'],d['Absenteeism time in hours'])


# In[70]:


#Freq distribution for category Variable
import pandas as pd
for i in c_col:
                                                                    df1 =
pd.Series(data[i]).value_counts().reset_index().sort_values('index').reset_index(drop
=True)
    df1.columns = ['Values', 'Frequency']
    print(i)
    print(df1)


# In[71]:


data.pivot_table(index='Month of absence', columns = 'Day of the week',
aggfunc={'Absenteeism time in hours':sum})


# In[72]:


# Scatter Plot
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
plt.style.use('seaborn-white')
plt.figure()
```

```
plt.plot(data['Distance from Residence to Work'],data['Transportation expense']**3,'o')
plt.xlabel('Distance from Residence to Work')
plt.ylabel('Transportation expense')
plt.figure()
plt.plot(data['Weight']/data['Height']**2,data['Body mass index'])
plt.xlabel('W/H^2')
plt.ylabel('BMI')
plt.figure()
plt.plot(data['Disciplinary failure'],data['Absenteeism time in hours'],'o')
plt.figure()
plt.plot(data['Service time'],data['Hit target'],'o')
plt.figure()
plt.plot(data['Service time'],data['Age'])
plt.figure()
plt.plot(data['Weight'],data['Body mass index'])
plt.figure()
plt.plot(1/data['Height']**2,data['Body mass index'])


# In[73]:


# Scatter Pair Plots
# Create columns of seasons for pair plotting colors
# Seasons (summer (1), autumn (2), winter (3), spring (4))
seasons = data['Seasons']
for i in range(len(seasons)):
    if seasons[i] == 1:
        seasons[i] = 'summer'
    if seasons[i] == 2:
        seasons[i] = 'autumn'
    if seasons[i] == 3:
        seasons[i] = 'winter'
    if seasons[i] == 4:
        seasons[i] = 'spring'

# Will only use six variables for plotting pairs
reduced_features = data[['Reason for absence', 'Work load Average/day ', 'Month of
absence',
                         'Weight','Absenteeism time in hours']].copy()
reduced_features['season'] = seasons
# Use seaborn for pair plots
import seaborn as sns
sns.set(style="ticks", color_codes=True);
# Create a custom color palete
palette = sns.xkcd_palette(['dark blue', 'dark green', 'gold', 'orange'])
# Make the pair plot with a some aesthetic changes
sns.pairplot(reduced_features, hue = 'season', diag_kind = 'kde', palette= palette,
plot_kws=dict(alpha = 0.7),
```

```
                diag_kws=dict(shade=True))
```

In[74]:

```
data = data_org
#data.head()
```

In[75]:

```
# Assigning Levels/Categories to Categorical Variables
#for i in c_col:
#    data[i] = pd.Categorical(data[i])
#    print(i)
#    data[i] = data[i].cat.codes
#data['Reason for absence'].unique()
# Above Code replaces 'nan' with '-1', replacing '-1' with 'nan' to impute
#for i in c_col:
#    data[i] = data[i].replace(-1,np.nan)
#print(sorted(data.Seasons.unique()), end = ' ')
```

In[76]:

```
# Identify Columnwise Missing Values
missing_val = pd.DataFrame(data.isnull().sum())
missing_val.reset_index()
missing_val = missing_val.rename(columns = {'index' : 'variables', 0:'miss_value'})
row,col = data.shape
missing_val['miss_percentage'] = round((missing_val['miss_value']/row)*100,2)
missing_val = missing_val.sort_values('miss_percentage',ascending=False)
# Select sample data and find best method to predict missing values
missing_val
```

In[77]:

```
# impute BMI, Weight,Height
temp = data[['Weight', 'Height', 'Body mass index']]
temp['pred_BMI'] = (data['Weight']/(data['Height']**2))*10000
temp['error_BMI'] = temp['Body mass index']-temp['pred_BMI']
print(temp['error_BMI'].dropna().mean())
data['Body mass index'].fillna((data['Weight']/(data['Height']**2))*10000,inplace =
True)
```

```
# impute Height
temp['pred_Height'] = ((data['Weight']/data['Body mass index'])*10000)**0.5
temp['error_Height'] = temp['Height']-temp['pred_Height']
print(temp['error_Height'].dropna().mean())
data['Height'].fillna(((data['Weight']/data['Body mass index'])*10000)**0.5,inplace = True)
# impute Weight
temp['pred_Weight'] = ((data['Height']**2)*data['Body mass index'])/10000
temp['error_Weight'] = temp['Weight']-temp['pred_Weight']
print(temp['error_Weight'].dropna().mean())
data['Weight'].fillna(((data['Height']**2)*data['Body mass index'])/10000,inplace = True)


# In[78]:


data.head()


# In[79]:


# Selecting best method to impute missing value
# Crete dummy missing value
temp_data = data
temp_data['Body mass index'].iloc[22] # 27.0
temp_data['Body mass index'].iloc[22] = np.NaN
#temp_data['Body mass index'].iloc[22]
temp_data['Body mass index'].mean() #26.68
temp_data['Body mass index'].iloc[22] = np.NaN
temp_data['Body mass index'].median() #25
temp_data['Body mass index'].iloc[22] = np.NaN
temp_data = pd.DataFrame(KNN(k=3).complete(temp_data),columns = temp_data.columns)
temp_data['Body mass index'].iloc[22] # 27
# Best match by KNN
data = temp_data
# Check any Columnwise Missing Values
pd.DataFrame(data.isnull().sum())


# In[ ]:


# Dropping the 22 missing 'Absenteeism time in hours' observations
#data = data.iloc[data['Absenteeism time in hours'].dropna().index,:]
#data.shape
```

```
# In[ ]:


#Remove Outliers using IQR
#for i in n_col:
#    q25,q75 = np.percentile(data[i],[25,75])
#   iqr = q75-q25
#   max = q75+1.5*iqr
#    min = q25-1.5*iqr
#    data = data.drop(data[data.loc[:,i]<min].index)
#    data = data.drop(data[data.loc[:,i]>max].index)
# data.shape (516,21)


# In[80]:


# Impute outliers
df = data
for col in n_col:
    percentiles = df[col].quantile([0.01,0.99]).values
    df[col][df[col] <= percentiles[0]] = percentiles[0]
    df[col][df[col] >= percentiles[1]] = percentiles[1]
data = df
data.shape


# In[81]:


data_wo_scal = data


# In[82]:


# Impute Outliers
#data = temp_data
#for i in col_name:
#    q25,q75 = np.percentile(data[i],[25,75])
#    iqr = q75-q25
#    max = q75+1.5*iqr
#    min = q25-1.5*iqr
#    data[i][data[i]>max] = np.NaN
#    data[i][data[i]<min] = np.NaN

#data = pd.DataFrame(KNN(k=3).complete(data),columns = data.columns)
#data.shape
```

In[83]:

```
#Feature Scaling
for i in n_col[:11]:
    data[i] = (data[i]-data[i].min())/(data[i].max()-data[i].min())
data.shape, #data.head()
```

In[84]:

```
# Correlation Analysis
rho = data[n_col].corr()
rho
```

In[85]:

```
#Corelation Analysis
#Selecting Continuos Variable
data_corr = data.loc[:,n_col]
f,ax = plt.subplots(figsize = (7,5))
corr = data_corr.corr()

import seaborn as sns
sns.heatmap(corr,mask = np.zeros_like(corr, dtype = np.bool),cmap =
sns.diverging_palette(220,10,as_cmap=True),
            square = True, ax =ax)
```

In[86]:

```
from scipy.stats import chi2_contingency
for i in c_col:
    print(i)
        chi2, p, dof, ex = chi2_contingency(pd.crosstab(data['Absenteeism time in
hours'],data[i]))
    print(p)
```

In[87]:

```
# Feature Importance Matrix Random Forest
```

```
import pandas as pd
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(n_estimators=300).fit(data.iloc[:, 0:20],data.iloc[:,20])
importances = list(rf_model.feature_importances_)
feature_importances = [(feature, round(importance, 2)) for feature, importance in zip(list(data.columns),
                                                                                      importances)]
# Sort the feature importances by most important first
feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse = True)
# Print out the feature and importances
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in feature_importances]


# In[88]:


# lImportance plotting
x_values = list(range(len(importances)))
# Make a bar chart
plt.bar(x_values, importances, orientation = 'vertical', color = 'r', edgecolor = 'k', linewidth = 1.2)
# Tick labels for x axis
plt.xticks(x_values, data.columns, rotation='vertical')
# Axis labels and title
plt.ylabel('Importance'); plt.xlabel('Variable'); plt.title('Variable Importances');


# In[89]:


# List of features sorted from most to least important
sorted_importances = [importance[1] for importance in feature_importances]
sorted_features = [importance[0] for importance in feature_importances]
# Cumulative importances
cumulative_importances = np.cumsum(sorted_importances)
# Make a line graph
plt.plot(x_values, cumulative_importances, 'g-')
# Draw line at 95% of importance retained
plt.hlines(y = 0.95, xmin=0, xmax=len(sorted_importances), color = 'r', linestyles = 'dashed')
# Format x ticks and labels
plt.xticks(x_values, sorted_features, rotation = 'vertical')
# Axis labels and title
```

```
plt.xlabel('Variable'); plt.ylabel('Cumulative Importance'); plt.title('Cumulative Importances');
# Find number of features for cumulative importance of 95%
# Add 1 because Python is zero-indexed
print('Number of features for 95% importance:', np.where(cumulative_importances > 0.95)[0][0] + 1)


# In[90]:


df1 = data.applymap(int)
for i in c_col:
    df1[i].astype('str')
temp = pd.DataFrame(df1['Absenteeism time in hours'])
temp = temp.join(data[n_col[0:-1]])
for i in c_col:
    d = pd.get_dummies(df1[i],prefix = i)
    temp = temp.join(d)
data_hotencod = temp
data_hotencod.shape


# In[91]:


data_wo_scal=data_wo_scal.applymap(int)
data.head()


# In[92]:


#df1 = pd.Series(data_wo_scal['Reason for absence']).value_counts().reset_index().sort_values('index').reset_index(drop=True)
#data_wo_scal_original =pd.read_excel('Absenteeism_at_work_Project .xls')
#df1 = pd.Series(data_wo_scal['Seasons']).value_counts().reset_index().sort_values('index').reset_index(drop=True)
#df1 = pd.Series(data_wo_scal['Disciplinary failure']).value_counts().reset_index().sort_values('index').reset_index(drop=True)
#df1 = pd.Series(data_wo_scal['Reason for absence']).value_counts().reset_index().sort_values('index').reset_index(drop=True)
#df1 = pd.Series(data_wo_scal['Day of the week']).value_counts().reset_index().sort_values('index').reset_index(drop=True)
#df1 = pd.Series(data_wo_scal['Social smoker']).value_counts().reset_index().sort_values('index').reset_index(drop=True)
#df1 = pd.Series(data_wo_scal['Social drinker']).value_counts().reset_index().sort_values('index').reset_index(drop=True)
```

```
# d f 1 =
pd.Series(data_wo_scal['Education']).value_counts().reset_index().sort_values('inde
x').reset_index(drop=True)
# d f 1 =
pd.Series(data_wo_scal['Pet']).value_counts().reset_index().sort_values('index').rese
t_index(drop=True)
# d f 1 =
pd.Series(data_wo_scal['Age']).value_counts().reset_index().sort_values('index').res
et_index(drop=True)
# d f 1 =
pd.Series(data_wo_scal['Weight']).value_counts().reset_index().sort_values('index').r
eset_index(drop=True)
#df1 = pd.Series(data_wo_scal['Distance from Residence to
Work']).value_counts().reset_index().sort_values('index').reset_index(drop=True)
#df1 = pd.Series(data_wo_scal['Work load Average/day
']).value_counts().reset_index().sort_values('index').reset_index(drop=True)
df1 = pd.Series(data_wo_scal['Hit
target']).value_counts().reset_index().sort_values('index').reset_index(drop=True)

df1.columns = ['Values', 'Frequency']
df1['% Freq'] = df1['Frequency']/(len(data_wo_scal))*100

print(df1.sort_values('% Freq',ascending=False))


# In[97]:


# Hot Encoded/dummy Varibles Data
# Feature Importance Matrix Random Forest for
import pandas as pd
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(n_estimators=300).fit(data_hotencod.iloc[:,
1:],data_hotencod.iloc[:,0])
importances = list(rf_model.feature_importances_)
feature_importances = [(feature, round(importance, 2)) for feature, importance in
zip(list(data_hotencod.columns),
                                                    importances)]
# Sort the feature importances by most important first
feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse =
True)
# Print out the feature and importances
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in feature_importances]
```

In[99]:

```
# lImportance plotting
x_values = list(range(len(importances)))
# Make a bar chart
plt.bar(x_values, importances, orientation = 'vertical', color = 'r', edgecolor = 'k', linewidth = 1.2)
# Tick labels for x axis
plt.xticks(x_values, data_hotencod.columns, rotation='vertical')
# Axis labels and title
plt.ylabel('Importance'); plt.xlabel('Variable'); plt.title('Variable Importances');
```

In[105]:

```
# List of features sorted from most to least important
sorted_importances = [importance[1] for importance in feature_importances]
sorted_features = [importance[0] for importance in feature_importances]
# Cumulative importances
cumulative_importances = np.cumsum(sorted_importances)
# Make a line graph
plt.plot(x_values, cumulative_importances, 'g-')
# Draw line at 90% of importance retained
plt.hlines(y = 0.90, xmin=0, xmax=len(sorted_importances), color = 'r', linestyles = 'dashed')
# Format x ticks and labels
plt.xticks(x_values, sorted_features, rotation = 'vertical')
# Axis labels and title
plt.xlabel('Variable'); plt.ylabel('Cumulative Importance'); plt.title('Cumulative Importances');
# Find number of features for cumulative importance of 90%
# Add 1 because Python is zero-indexed
print('Number of features for 90% importance:', np.where(cumulative_importances > 0.90)[0][0] + 1)
```

In[103]:

```
cumulative_importances>0.90
```

Model_Code :

```
# coding: utf-8

# In[1]:


#2. How much losses every month can we project in 2011 if same trend of absenteeism continues?


# In[2]:


##Import Libararies
import pandas as pd
import numpy as np
from fancyimpute import KNN
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.formula.api import ols
from sklearn.decomposition import PCA
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import Ridge, Lasso, LinearRegression,Lars
from sklearn.ensemble import  RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression


# In[3]:


##Import Data
data = pd.read_excel('Absenteeism_at_work_Project.xls')
data_org = data
col = data.columns
# Numerical and Categorical Variables
n_col = ['Transportation expense', 'Distance from Residence to Work','Service time',
        'Age', 'Work load Average/day ', 'Hit target','Son','Pet', 'Weight', 'Height',
        'Body mass index','Absenteeism time in hours']
c_col = ['ID', 'Reason for absence', 'Month of absence', 'Day of the week','Seasons',
'Disciplinary failure',
        'Education','Social drinker','Social smoker',]
```

In[4]:

```
# Identify Columnwise Missing Values
missing_val = pd.DataFrame(data.isnull().sum())
missing_val.reset_index()
missing_val = missing_val.rename(columns = {'index' : 'variables', 0:'miss_value'})
row,col = data.shape
missing_val['miss_percentage'] = round((missing_val['miss_value']/row)*100,2)
missing_val = missing_val.sort_values('miss_percentage',ascending=False)
# Select sample data and find best method to predict missing values
#missing_val
# impute BMI
temp = data[['Weight', 'Height', 'Body mass index']]
temp['pred_BMI'] = (data['Weight']/(data['Height']**2))*10000
temp['error_BMI'] = temp['Body mass index']-temp['pred_BMI']
#print(temp['error_BMI'].dropna().mean())
data['Body mass index'].fillna((data['Weight']/(data['Height']**2))*10000,inplace =
True)
# impute Height
temp['pred_Height'] = ((data['Weight']/data['Body mass index'])*10000)**0.5
temp['error_Height'] = temp['Height']-temp['pred_Height']
#print(temp['error_Height'].dropna().mean())
data['Height'].fillna(((data['Weight']/data['Body mass index'])*10000)**0.5,inplace =
True)
# impute Weight
temp['pred_Weight'] = ((data['Height']**2)*data['Body mass index'])/10000
temp['error_Weight'] = temp['Weight']-temp['pred_Weight']
#print(temp['error_Weight'].dropna().mean())
data['Weight'].fillna(((data['Height']**2)*data['Body mass index'])/10000,inplace =
True)
# Selecting best method to impute missing value
# Crete dummy missing value
temp_data = data
temp_data['Body mass index'].iloc[22] # 27.0
temp_data['Body mass index'].iloc[22] = np.NaN
#temp_data['Body mass index'].iloc[22]
temp_data['Body mass index'].mean() #26.68
temp_data['Body mass index'].iloc[22] = np.NaN
temp_data['Body mass index'].median() #25
temp_data['Body mass index'].iloc[22] = np.NaN
temp_data = pd.DataFrame(KNN(k=3).complete(temp_data),columns =
temp_data.columns)
temp_data['Body mass index'].iloc[22] # 27
# Best match by KNN
data = temp_data
# Check any Columnwise Missing Values
#pd.DataFrame(data.isnull().sum())
```

In[5]:

```
# Impute outliers
df = data
for col in n_col:
    percentiles = df[col].quantile([0.01,0.99]).values
    df[col][df[col] <= percentiles[0]] = percentiles[0]
    df[col][df[col] >= percentiles[1]] = percentiles[1]
data = df
#data.shape
```

In[6]:

```
df1 = data.applymap(int)
for i in c_col[1:]:
    df1[i].astype('str')
temp = pd.DataFrame(df1['Absenteeism time in hours'])
temp = temp.join(data[n_col[0:-1]])
for i in c_col[1:]:
    d = pd.get_dummies(df1[i],prefix = i)
    temp = temp.join(d)
data_hotencod = temp
data_hotencod.shape
```

In[7]:

```
# Hot Encoded/dummy Varibles Data
# Feature Importance Matrix Random Forest for
import pandas as pd
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(n_estimators=300).fit(data_hotencod.iloc[:,
1:],data_hotencod.iloc[:,0])
importances = list(rf_model.feature_importances_)
feature_importances = [(feature, round(importance, 2)) for feature, importance in
zip(list(data_hotencod.columns),
                                                    importances)]
# Sort the feature importances by most important first
feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse =
True)
```

```
# Print out the feature and importances
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in feature_importances]


# In[8]:


# lImportance plotting
x_values = list(range(len(importances)))
# Make a bar chart
plt.bar(x_values, importances, orientation = 'vertical', color = 'r', edgecolor = 'k', linewidth = 1.2)
# Tick labels for x axis
plt.xticks(x_values, data.columns, rotation='vertical')
# Axis labels and title
plt.ylabel('Importance'); plt.xlabel('Variable'); plt.title('Variable Importances');


# In[9]:


# List of features sorted from most to least important
sorted_importances = [importance[1] for importance in feature_importances]
sorted_features = [importance[0] for importance in feature_importances]
# Cumulative importances
cumulative_importances = np.cumsum(sorted_importances)
# Make a line graph
plt.plot(x_values, cumulative_importances, 'g-')
# Draw line at 90% of importance retained
plt.hlines(y = 0.90, xmin=0, xmax=len(sorted_importances), color = 'r', linestyles = 'dashed')
# Format x ticks and labels
plt.xticks(x_values, sorted_features, rotation = 'vertical')
# Axis labels and title
plt.xlabel('Variable'); plt.ylabel('Cumulative Importance'); plt.title('Cumulative Importances');
# Find number of features for cumulative importance of 90%
# Add 1 because Python is zero-indexed
print('Number of features for 90% importance:', np.where(cumulative_importances > 0.90)[0][0] + 1)


# In[10]:


#selecting top 15 features
new = SelectKBest(f_regression,k=15).fit_transform(data_hotencod.iloc[:, 1:],data_hotencod.iloc[:,0])
```

```
X_train, X_test, y_train, y_test = train_test_split(new,data_hotencod.iloc[:,0], test_size=0.2,random_state=42)


# In[11]:


#Decision Tree
dt_model=DecisionTreeRegressor(random_state=3).fit(X_train,y_train)
pred = dt_model.predict(X_test)
print('RMSE : ',np.sqrt(mean_squared_error(y_test,pred)))
#print('Model Score :', rf_model.score(X_test, y_test))


# In[12]:


#Random forest
rf_model = RandomForestRegressor(n_estimators=700).fit(X_train,y_train)
pred = rf_model.predict(X_test)
print('RMSE : ',np.sqrt(mean_squared_error(y_test,pred))),
print('Model Score :', rf_model.score(X_test, y_test))


# In[13]:


#Linear Regression
import statsmodels.api as sm
lr_model = sm.OLS(y_train,X_train).fit()
pred = lr_model.predict(X_test)
print('RMSE : ',np.sqrt(mean_squared_error(y_test,pred)))
#print('Model Score :', lr_model.score(X_test, y_test))#lr_model.summary()


# In[14]:


#Ridge Regression
ridge_model=Ridge().fit(X_train,y_train)
pred = ridge_model.predict(X_test)
print('RMSE : ',np.sqrt(mean_squared_error(y_test,pred)))
print('Model Score :', ridge_model.score(X_test, y_test))


# In[15]:


#Lasso regression
```

```
lasso_model=Lasso().fit(X_train,y_train)
pred = lasso_model.predict(X_test)
print('RMSE : ',np.sqrt(mean_squared_error(y_test,pred)))
print('Model Score :', lasso_model.score(X_test, y_test))


# In[16]:


#Lars Regression
lars_model=Lars().fit(X_train,y_train)
pred = lars_model.predict(X_test)
print('RMSE : ',np.sqrt(mean_squared_error(y_test,pred))),print('Model Score :',
lars_model.score(X_test, y_test))


# In[17]:


#data_hotencod.columns


# In[18]:


#data=data_hotencod.drop(['ID'],axis=1)
#pca = PCA(n_components=73)
#pca.fit(data_hotencod.values)
#The amount of variance that each PC explained
#var= pca.explained_variance_ratio_
#Cumulative Variance
var1=np.cumsum(np.round(pca.explained_variance_ratio_,decimals=4)*100)
#graph of the variance
#plt.plot(var1)


# In[ ]:


##############################
## from the above plot
#The plot above shows that ~ 40 components explains around 99% variance in the
data set.
#By using PCA we have reduced 72 predictors to 40 without compromising on
explained variance.
##############################
#Looking at above plot I'm taking 40 variables
pca = PCA(n_components=37)
#now fitting the selected components to the data
```

```
pca.fit(data_hotencod.values)
#PCA selected features
X1=pca.fit_transform(data_hotencod.values)
#splitting train and test data
X_train_pca, X_test_pca, y_train_pca, y_test_pca =
train_test_split(X1,data_hotencod['Absenteeism time in hours'],
                                                test_size=0.2,random_state=42)


# In[ ]:


# PCA_Decision Tree
dt_model=DecisionTreeRegressor(random_state=3).fit(X_train_pca,y_train_pca)
pred = dt_model.predict(X_test_pca)
print('RMSE : ',np.sqrt(mean_squared_error(y_test_pca,pred)))
print('Model Score :', rf_model.score(X_test, y_test))


# In[ ]:


#PCA_Random forest
rf_model = RandomForestRegressor(n_estimators=700).fit(X_train_pca,y_train_pca)
pred = rf_model.predict(X_test_pca)
print('RMSE : ',np.sqrt(mean_squared_error(y_test,pred))),
print('Model Score :', rf_model.score(X_test_pca, y_test_pca))


# In[ ]:


#Linear Regression
import statsmodels.api as sm
lr_model = sm.OLS(y_train_pca,X_train_pca).fit()
pred = lr_model.predict(X_test_pca)
print('RMSE : ',np.sqrt(mean_squared_error(y_test_pca,pred)))
#print('Model Score :', lr_model.score(X_test_pca, y_test_pca))#lr_model.summary()


# In[ ]:


#Ridge Regression
ridge_model=Ridge().fit(X_train_pca,y_train_pca)
pred = ridge_model.predict(X_test_pca)
print('RMSE : ',np.sqrt(mean_squared_error(y_test_pca,pred)))
print('Model Score :', ridge_model.score(X_test_pca, y_test_pca))
```

In[ ]:

```
#Lasso regression
lasso_model=Lasso().fit(X_train_pca,y_train_pca)
pred = lasso_model.predict(X_test_pca)
print('RMSE : ',np.sqrt(mean_squared_error(y_test_pca,pred)))
print('Model Score :', lasso_model.score(X_test_pca, y_test_pca))
```

In[ ]:

```
#Lars Regression
lars_model=Lars().fit(X_train_pca,y_train_pca)
pred = lars_model.predict(X_test_pca)
print('RMSE : ',np.sqrt(mean_squared_error(y_test,pred))),print('Model Score :',
lars_model.score(X_test_pca, y_test_pca))
```

R Code :

`

```{r}
# Load Libraries
rm(list = ls())
#  install.packages("corrgram",  lib="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")
#  install.packages("sampling",  lib="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")
# install.packages("DataCombine", lib="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")
#  install.packages("caret",  lib="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")
#install.packages(c("forcats", "DataExplorer", "ggthemes","grid","gridExtra","factoextra","FactoMineR"))
#install.packages("DataExplorer",  lib="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")
#install.packages("psych")
library(psych)
library(ggplot2)
library(corrgram)
library(sampling)
library(corrgram)
library(class)
library(e1071)
library(caret)
library(DataCombine)
library(caret)
library(randomForest)
library(inTrees)
library(C50)
library(dplyr)
library(forcats)
library(plyr)
library(DataExplorer)
library(ggthemes)
library(grid)
library(gridExtra)
library(factoextra)
library(FactoMineR)
```


```{r}
# Import Data
data = read.csv('Absenteeism_at_work_Project_csv.csv', header = TRUE)
train = data = data.frame(data)
data_org=train
```

```
# list of Columns
col = colnames(train)
# list of numeric and categorical columns
n_col = c('Transportation.expense', 'Distance.from.Residence.to.Work','Service.time',
        'Age', 'Work.load.Average/day ', 'Hit.target','Son','Pet', 'Weight', 'Height',
        'Body.mass.index','Absenteeism.time.in.hours')
c_col = c('ID', 'Reason.for.absence', 'Month.of.absence', 'Day.of.the.week','Seasons',
'Disciplinary.failure',
        'Education','Social.drinker','Social.smoker')
```

```{r}
str(train) # Work.load.Average.day is 'Factor Variable'
train$Work.load.Average.day = as.integer(train$Work.load.Average.day)

# Convert catefory variables from numeric to factor type
for (i in c_col) {
   train[,i]=as.factor((train[,i]))
}

numeric_index = sapply(train,is.numeric)
numeric_data = train[,numeric_index]
cnames_n = colnames(numeric_data)

factor_index = sapply(train, is.factor)
factor_data = train[,factor_index]
cnames_f = colnames(factor_data)
```

```{r}
multi.hist(train[,numeric_index], main = NA, dcol = c("blue", "red"), dlty = c("solid",
"solid"), bcol = "linen")
summary(train)
```
```{r}
pairs.panels(train)
```

```{r}
# Missing Value Analysis
# Create datframe with missing percentage
missing_val = data.frame(apply(train,2,function(x){sum(is.na(x))}))
# Convert row names into column
missing_val$Columns = row.names(missing_val)
row.names(missing_val) = NULL
# Rename the variable name
names(missing_val)[1] = 'missing_percentage'
#Calculate percentage
```

```
missing_val$missing_percentage = (missing_val$missing_percentage/
nrow(missing_val))*100
# Arrange in descending order
missing_val = missing_val[order(-missing_val$missing_percentage),]
#Rearranging the columns
missing_val = missing_val[,c(2,1)]
```

```{r}
library(DMwR)
train = knnImputation(train, k =3)
```

```{r}
summary(train)
boxplot(train)
```

```{r}
for (i in 1:length(cnames_n)){
  assign(paste0("plot",i), ggplot(aes_string(y = (cnames_n[i]), x =
"Absenteeism.time.in.hours"), data =subset(train))+
 stat_boxplot(geom = "errorbar", width = 0.5) +
 geom_boxplot(outlier.colour="red", fill = "blue" ,outlier.shape=18,
 outlier.size=1, notch=FALSE) +
 theme(legend.position="bottom")+
 labs(y=cnames_n[i],x="Absenteeism.time.in.hours")+
 ggtitle(paste("Box plot of responded for",cnames_n[i])))
}
#boxplot of outliers
gridExtra::grid.arrange(plot1,plot2,plot3,ncol=3)
gridExtra::grid.arrange(plot4,plot5,plot6,ncol=3)
gridExtra::grid.arrange(plot7,plot8,plot9,ncol=3)
gridExtra::grid.arrange(plot10,plot11,plot12,ncol=3)
```

```{r}
for (i in cnames_f) {
  print(i)
  print(chisq.test(table(unlist(train[21]),unlist(train[i]))))
}
```

```{r}
# Capping and Flooring Outliers
for (i in cnames_n) {
  percentile = quantile(train[i], c(0.01, 0.99),na.rm = TRUE)
```

```
  train[i][train[i]<percentile[1]]=percentile[1]
  train[i][train[i]>percentile[2]]=percentile[2]
}
#train_deleted = subset(train, select = -c(area.code, phone.number, total.day.minutes, total.eve.minutes, total.night.minutes, total.intl.minutes))
#numeric_index = sapply(train_deleted,is.numeric)
#numeric_data = train_deleted[,numeric_index]
#cnames = colnames(numeric_data)
```

```{r}

for ( i in cnames_n[1:11]) {
  train[,i] = (train[,i] - min(train[,i]))/(max(train[,i]) - min(train[,i]))
}

```

```{r}
#install.packages("dummies",   lib="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")
library(dummies)
train_t = train
for (i in c_col[2:9]){
  temp = data.frame(dummy(train[,i]))
  train = cbind(train,temp)
  train[,i] = NULL
}

```

```{r}

corrgram(train[-1], order = F, upper.panel = panel.pie, text.panel = panel.txt, main = 'CorrelationPlot')
#symnum(cor(train[-1])
high_corr =  findCorrelation(cor(train[-1]), cutoff=0.95)
```

```{r}
#removing highly correlated columns
new_data = train[,-c(1,high_corr)]
new_train=cbind(train['ID'],new_data)
test=new_train[550:740,-1]
```

```{r}
#combining the clean data with the factor variables of original data
#temp_train=cbind(train_t[c_col],new_data)
#temp_train = new_data
#temp_train$Month.of.absence = as.numeric(temp_train$Month.of.absence)
#new_train <- subset(temp_train, Month.of.absence <11)
#new_test <- subset(temp_train, Month.of.absence>10)
#new_train$Month.of.absence = as.factor(new_train$Month.of.absence)
#new_test$Month.of.absence = as.factor(new_test$Month.of.absence)
```

```{r}
#install.packages('DAAG')
library(DAAG)
#feature selection using boruta package
install.packages("Boruta",  lib="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")
library(Boruta)
#new_train$Absenteeism.time.in.hours=log(new_train$Absenteeism.time.in.hours)
boruta.train=Boruta(Absenteeism.time.in.hours~., data = new_train[,-1], doTrace = 2)
selected_features=getSelectedAttributes(boruta.train, withTentative = F)
set.seed(123)
#creating formula from boruta selected features
formula=as.formula(paste("Absenteeism.time.in.hours~",paste(selected_features,collapse = "+")))
```
```{r}
# Linear Model
# Cross-Validation
train_control <- trainControl(method = "repeatedcv",number = 10)
options(warn=-1)
# Linear Model
lm_model<-  train(formula,data=new_train[1:500,-1],metric="RMSE", method="lm",trControl=train_control)
pred=predict(lm_model,test)
# Importance_Variable
lm_importance <- varImp(lm_model, scale=FALSE)
# Summary Importance
print(lm_model)
plot(lm_importance)
```

```{r}
#Random Forest
rf_model<-    train(formula,data=new_train[1:550,-1], metric="RMSE",method="rf",trControl=train_control)
pred=predict(rf_model,test)
# summarize model
```

```
print(rf_model)
# plot model
plot(rf_model)
```

```{r}
#decision tree model
dt_model<-    train(formula,data=new_train[1:550,-1],
metric="RMSE",method="rpart",trControl=train_control)
pred=predict(dt_model,test)
print(dt_model)
# plot importance
plot(dt_model)
```

```{r}
print(boruta.train)
```

```{r}
getSelectedAttributes(boruta.train, withTentative = F)
boruta.df <- attStats(boruta.train)
class(boruta.df)
print(boruta.df)
```

```{r}
# PCA, Principal component analysis
pca_train <- train[1:550,-1]
pca_test <- train[551:740,-1]
pc_analysis <- prcomp(pca_train)
#outputs the mean of variables prin_comp$center
pc_analysis$center
pc_analysis$scale
dim(pc_analysis$x)
biplot(pc_analysis, scale = 0)
```

```{r}
#compute standard deviation of each principal component
std_dev <- pc_analysis$sdev
#compute variance
pr_var <- std_dev^2
#proportion of variance explained
prop_varex <- pr_var/sum(pr_var)
#plot
```

```
plot(prop_varex, xlab = "Principal Component",ylab = "Proportion of Variance Explained",type = "b")
#cumulative plot
plot(cumsum(prop_varex), xlab = "Principal Component",ylab = "Cumulative Proportion of Variance Explained",type = "b")
```

```{r}
#add a training set with principal components
train.data <- data.frame(Absenteeism.time.in.hours =pca_train$Absenteeism.time.in.hours, pc_analysis$x)
#we are interested in first 40 PCAs as we have seen from the graph
# and the target variable ,so in total 41(including target variable)
train.data <- train.data[,1:41]

 #transform test into PCA
 test.data=predict(pc_analysis, newdata = pca_test)
 test.data= as.data.frame(test.data)
 #select the first 40 components
 test.data=test.data[,1:40]
```

```{r}
#linear regression
set.seed(123)
train_control=trainControl(method = "repeatedcv",number = 10,repeats = 6)
pca_lm_model=train(Absenteeism.time.in.hours~.,data=train.data,metric="RMSE",method="lm",trControl=train_control)
print(pca_lm_model)
print(summary(pca_lm_model))
#make prediction on test data
pca.lm.prediction = predict(pca_lm_model, test.data)
# absent rate in every month 2011 if same trend of absenteeism continues
temp = data_org[551:740,-1]
pca.lm.trend=data.frame(data_org[551:740,-1]$Month.of.absence,pca.lm.prediction)
ggplot(pca.lm.trend, aes(x = data_org.551.740...1..Month.of.absence ,y = pca.lm.prediction) ) + geom_bar(stat ='identity',fill='blue')
#finding RMSE on test data
print(RMSE(pca.lm.prediction,pca_test$Absenteeism.time.in.hours))
```

```{r}
#Decision tree
set.seed(123)
train_control = trainControl(method = "repeatedcv", number = 10, repeats = 6)
pca_dt_model=train(Absenteeism.time.in.hours~.,data=train.data,metric="RMSE",
method="rpart",tuneLength =10, trControl=train_control)
```

```
print(pca_dt_model)
print(summary(pca_dt_model))
#make prediction on test data
pca.dt.prediction = predict(pca_dt_model, test.data)

# absent rate in every month 2011 if same trend of absenteeism continues
pca.dt.trend=data.frame(data_org[551:740,-1]$Month.of.absence,pca.dt.prediction)
ggplot(pca.dt.trend, aes(x = data_org.551.740...1..Month.of.absence ,y = pca.dt.prediction) ) + geom_bar(stat = 'identity')
#finding RMSE on test data
print(RMSE(pca.dt.prediction,pca_test$Absenteeism.time.in.hours))
```

```{r}
#Lasso regression
#install.packages("elasticnet", lib="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")
library(elasticnet)
set.seed(123)
train_control = trainControl(method = "repeatedcv",number = 10,repeats = 6)
pca_lasso_model=train(Absenteeism.time.in.hours~.,data=train.data,metric="RMSE", method="lasso",tuneLength = 10,trControl=train_control)
print(pca_lasso_model)
print(summary(pca_lasso_model))
#make prediction on test data
pca.lasso.prediction = predict(pca_lasso_model, test.data)
# absent rate in every month 2011 if same trend of absenteeism continues
pca.lasso.trend=data.frame(data_org[551:740,-1]$Month.of.absence,pca.lasso.prediction)
ggplot(pca.lasso.trend, aes(x =data_org.551.740...1..Month.of.absence , y = pca.lasso.prediction) ) +geom_bar(stat = 'identity')
#finding RMSE on test data
print(RMSE(pca.lasso.prediction,pca_test$Absenteeism.time.in.hours))
```