Employee absenteeism

Sankepally vikram reddy

12/6/2018

Contents:

Chapter 1:

Introduction

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 it 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?
Build suitable model (both R and Python) to answer the above two questions

Data

Dataset Details:
Number of Attributes: 21
Missing Values : Yes
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)

Chapter 2:

Exploratory data analysis

Recoding factor variables

In order to view the distribution of classes in factor variables
We have to recode them back to their original form

Features to be recoded

Reason for abscence
Month of abscence
Seasons
Social drinker
Social Smoker
Disciplanary failure
Day of the week
Education

After recoding
The data looks like

```
Reason.for.absence                          Month.of.absence   Seasons
             physiotherapy                      Jul           summer
    infectious,parasitic diseases               Jul           summer
             blood donation                     Jul           summer
Diseases of the ear and mastoid process         Jul           summer
            blood donation                      Jul           summer
           blood donation                       Jul           summer

 Disciplinary.failure   Education   Social.drinker
       No               highschool       Yes
```

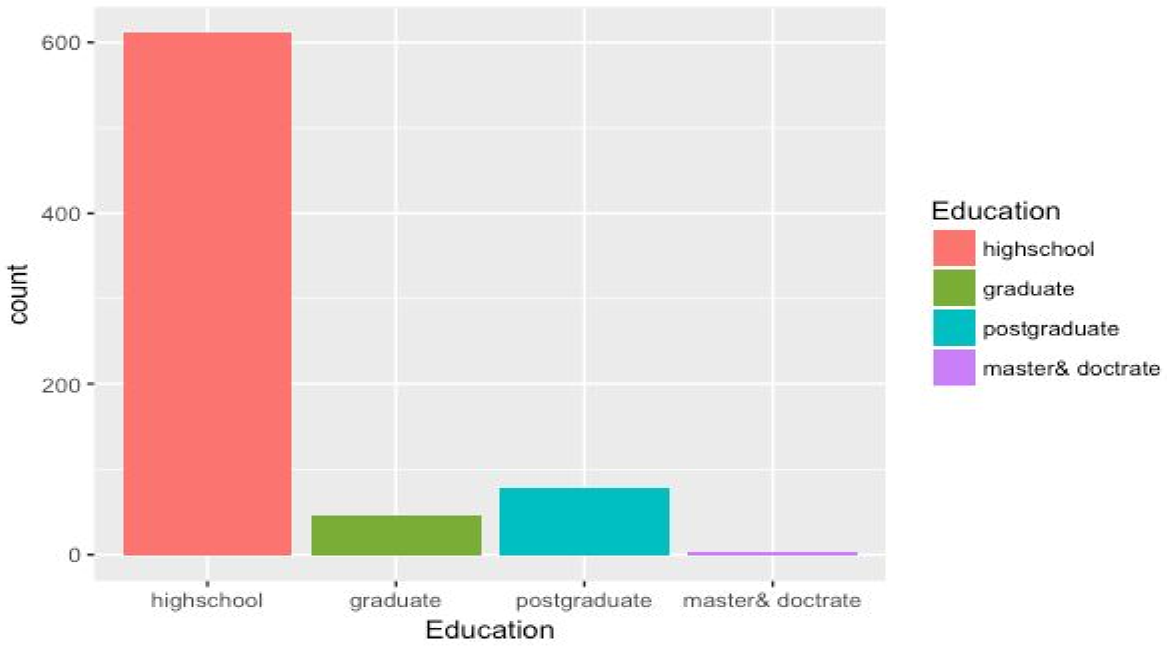
Yes	highschool	Yes
No	highschool	Yes
No	highschool	Yes
No	highschool	Yes
No	highschool	Yes

Social.smoker

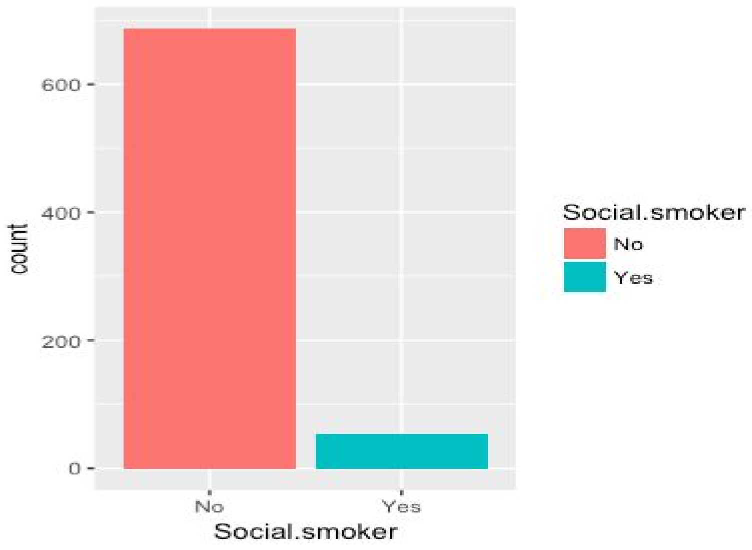
1	No
2	No
3	No
4	Yes
5	No
6	No

Data exploration:

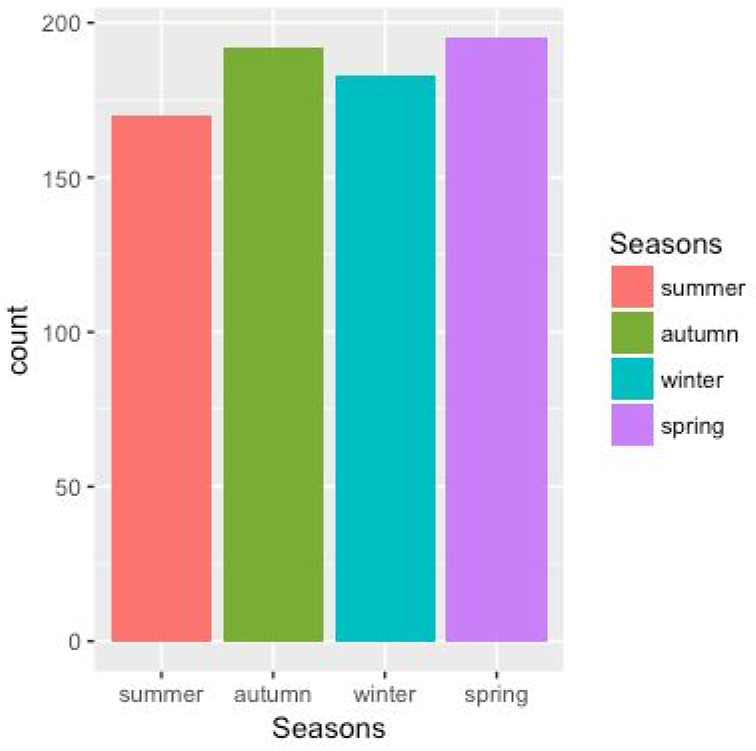
Distribution of education feature in the data



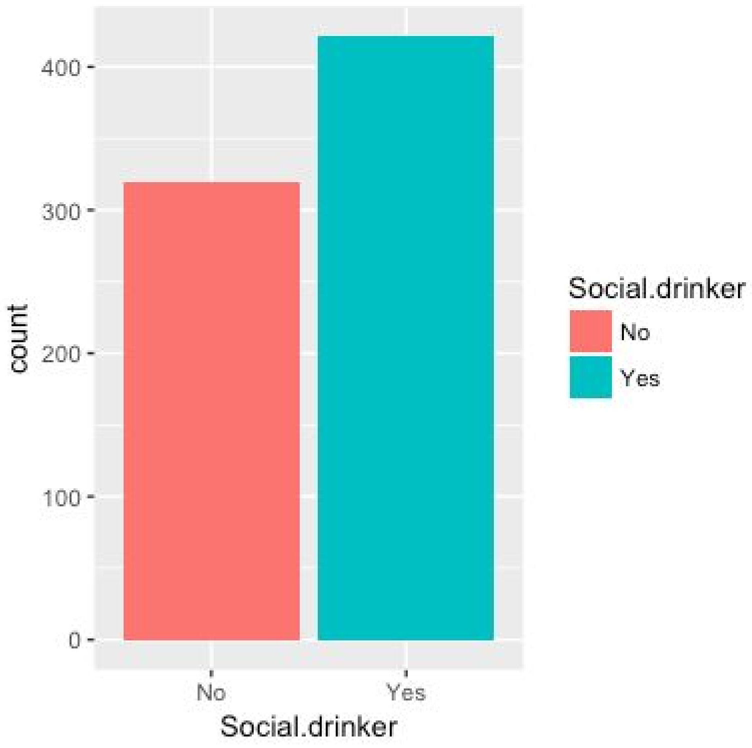
Distribution of Social.smoker feature in the data



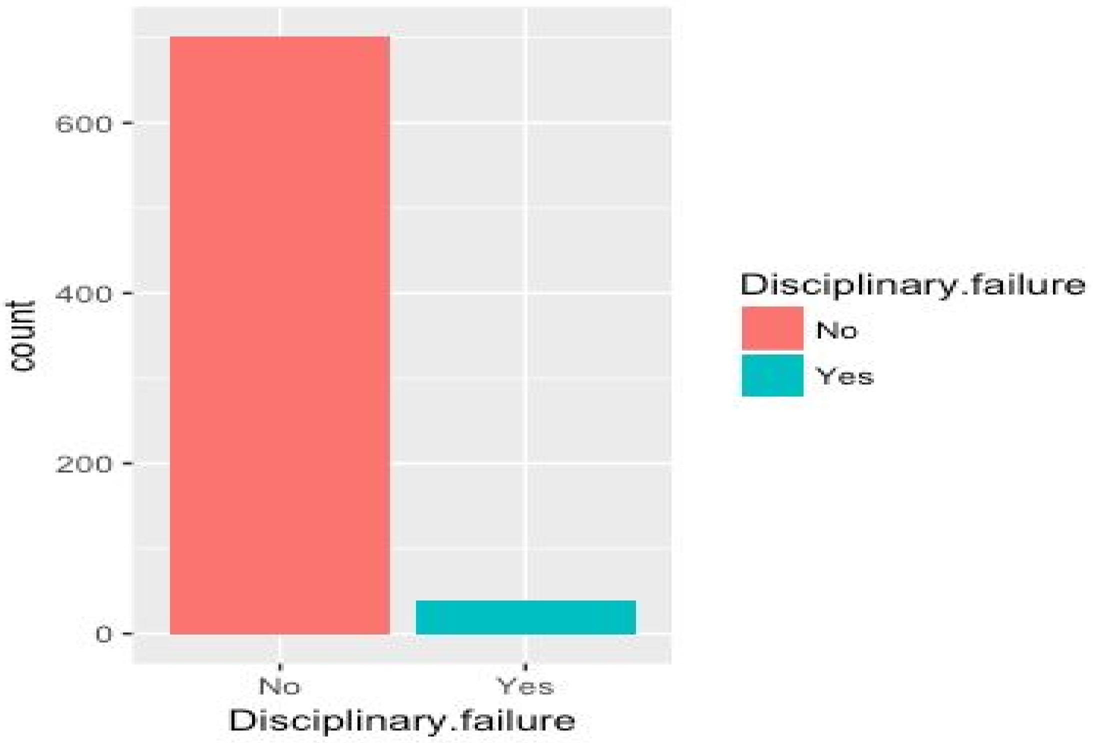
Distribution of Seasons feature in the data



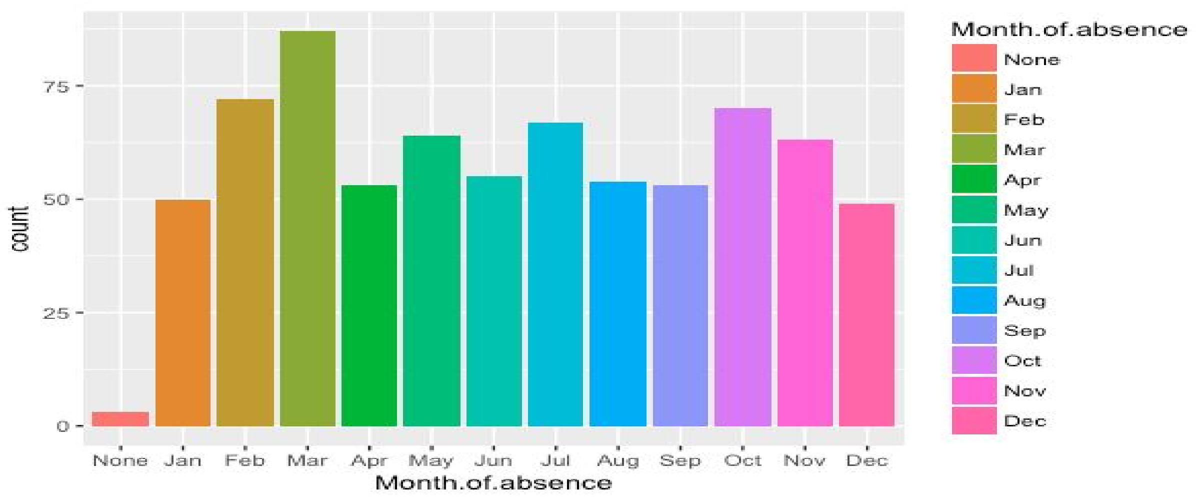
Distribution of social.drinker feature in the data



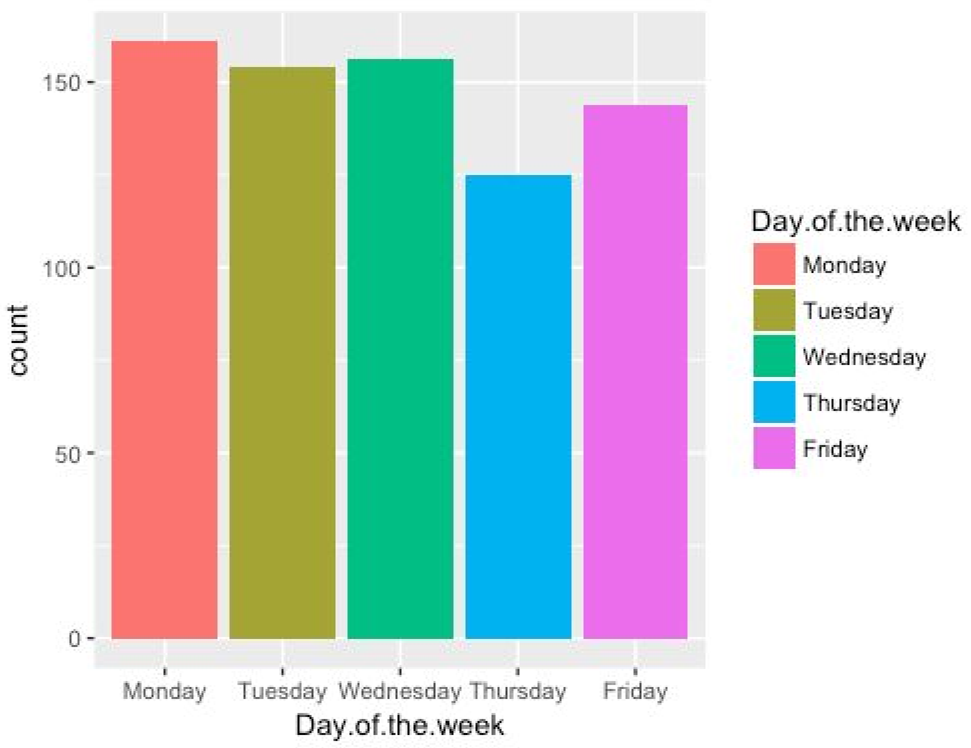
Distribution of Disciplinary.failure feature in the data



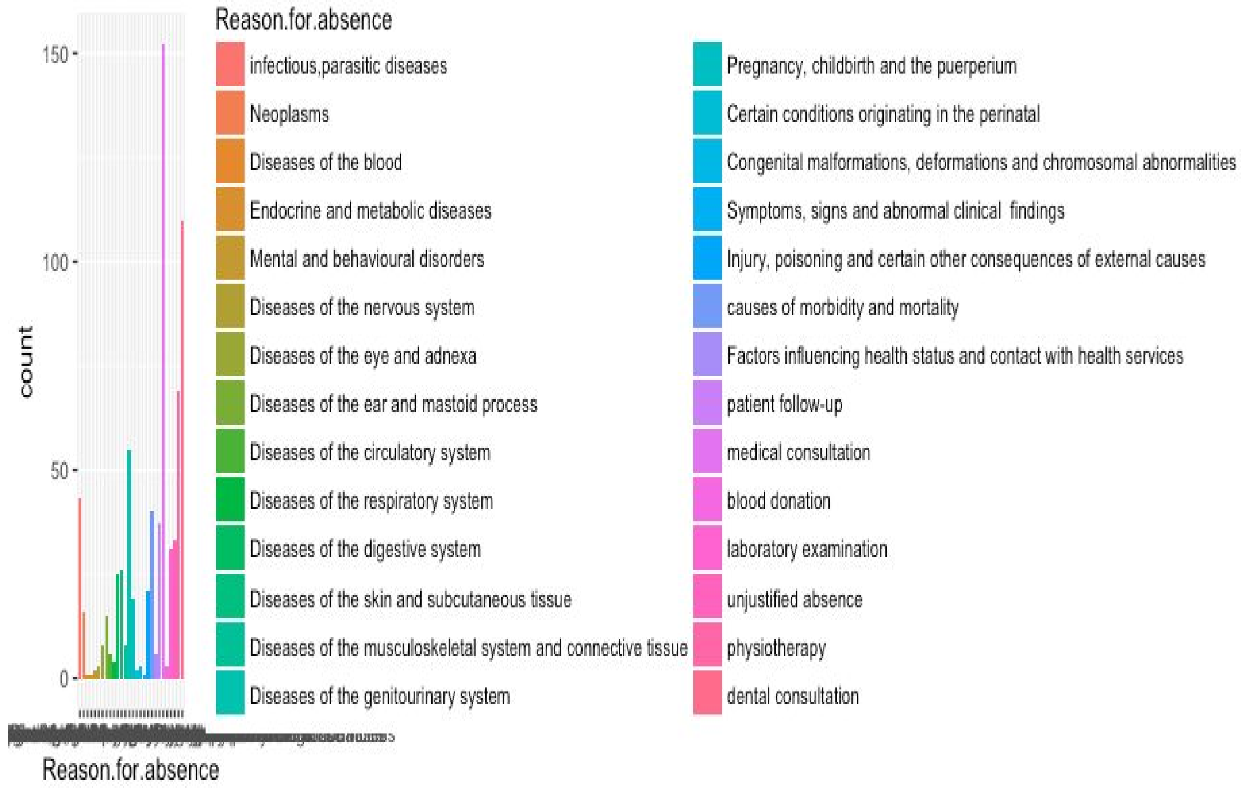
Distribution of month of absence feature



Distribution of day of the week feature

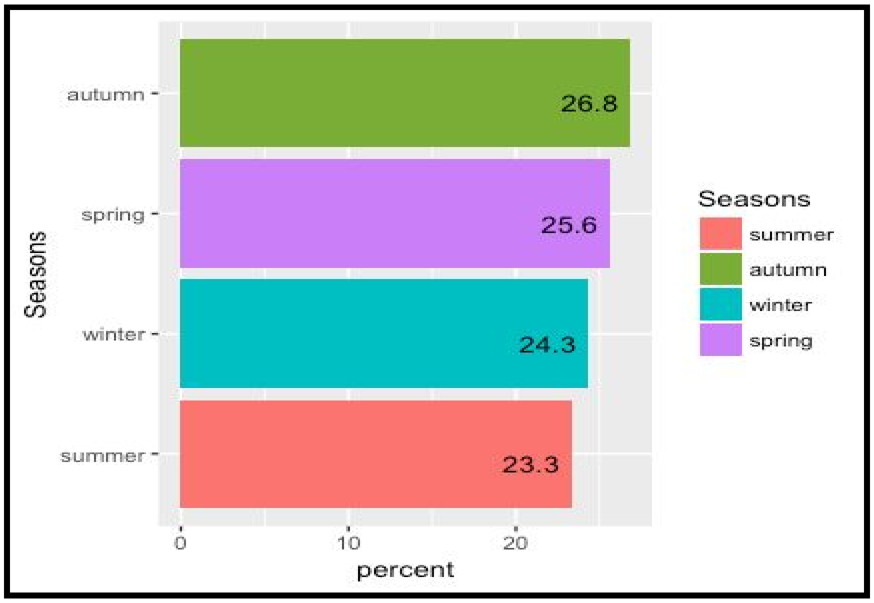

count
150
100
50
0
Monday
Tuesday
Wednesday
Thursday
Friday
Day.of.the.week
Day.of.the.week
Monday
Tuesday
Wednesday
Thursday
Friday

Distribution of reason of absence feature

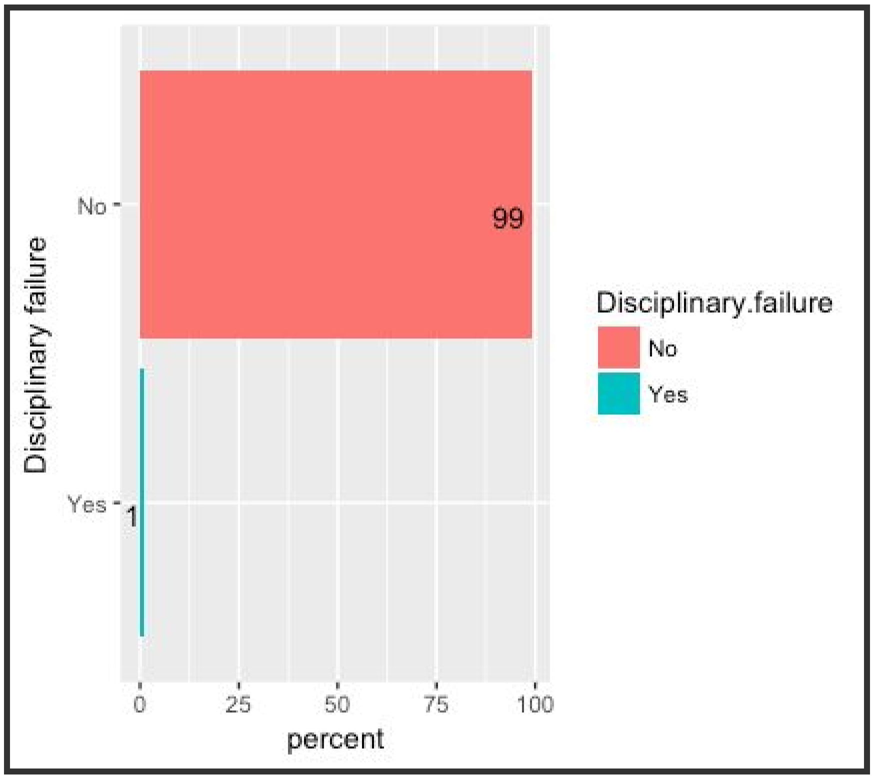


Percentage distribution of factor variables who are absent (absenteeism hours >0)

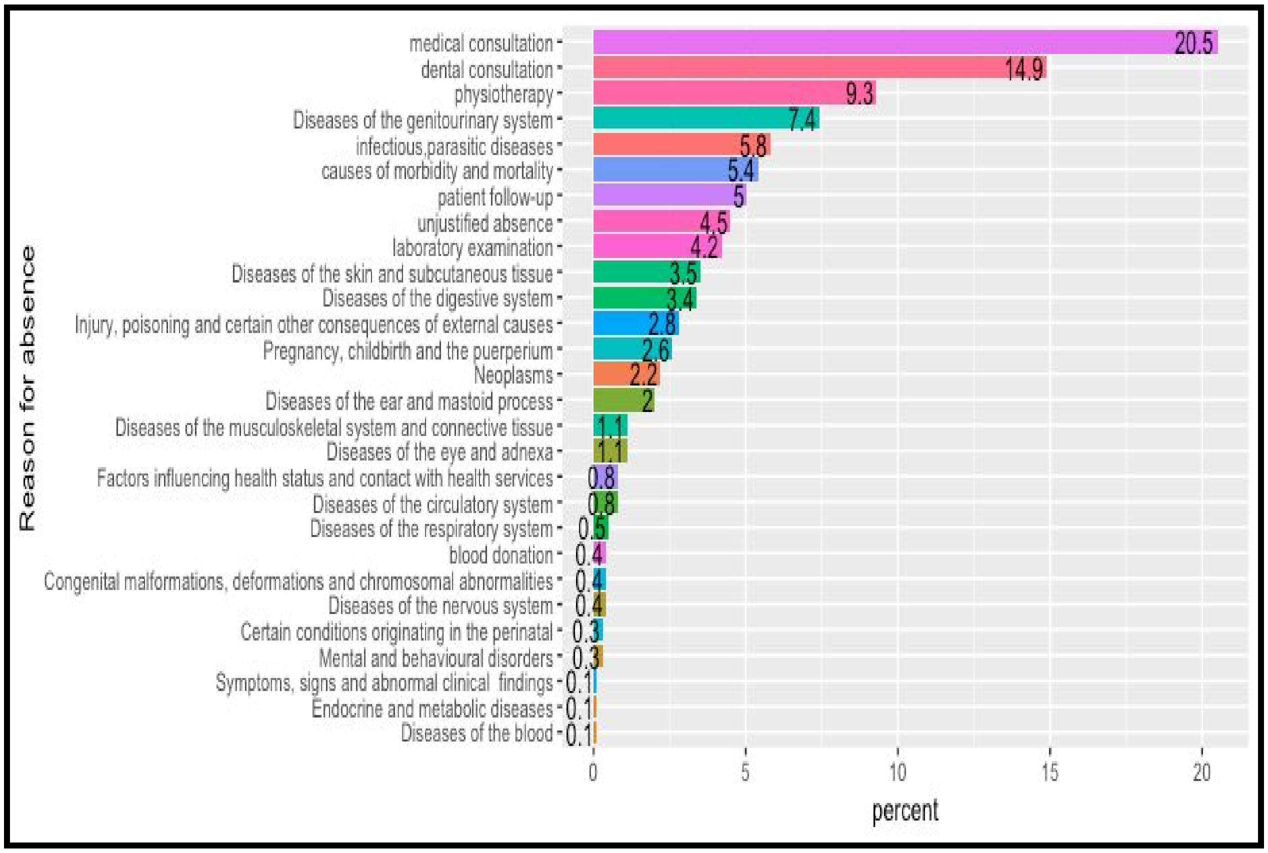
Seasons



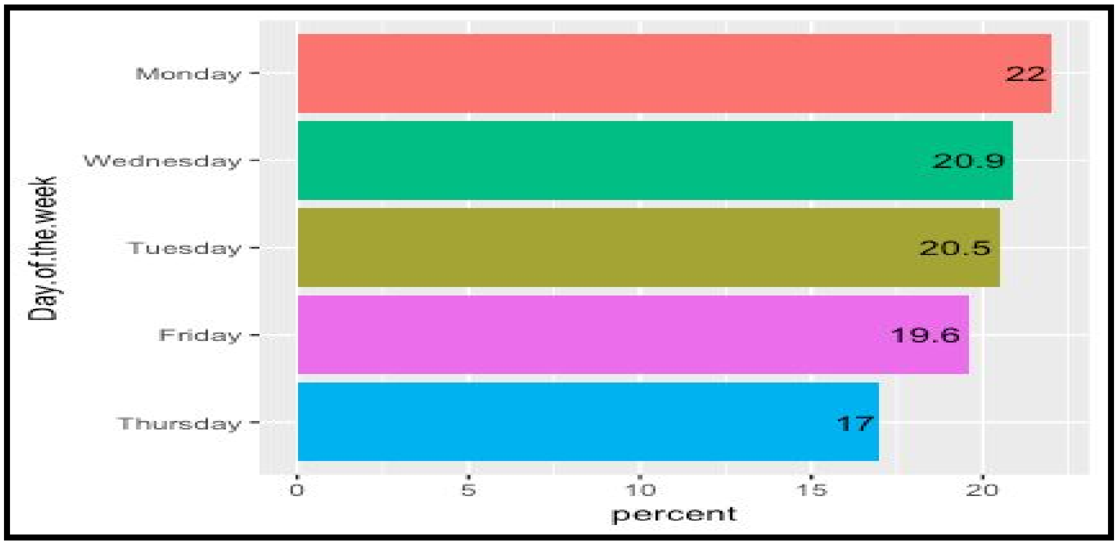
Disciplinary failure



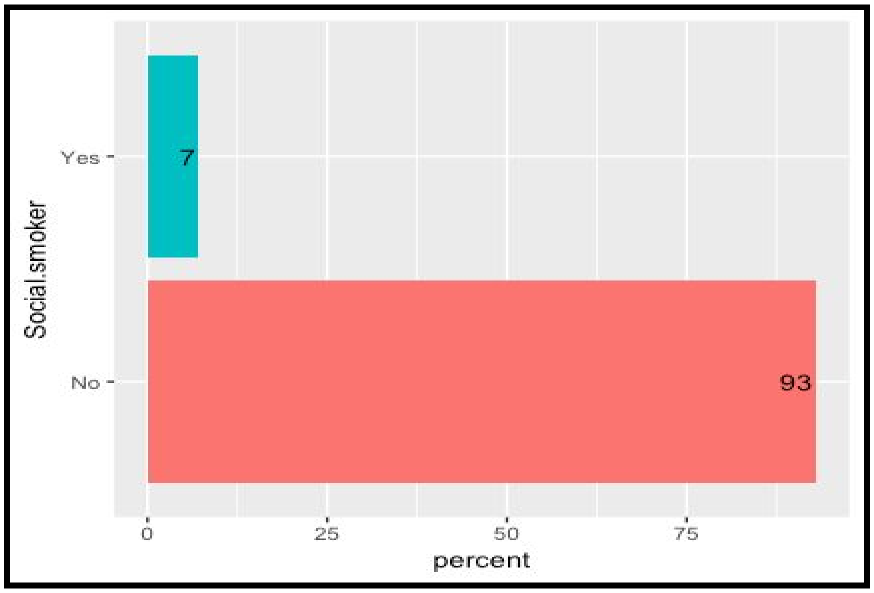
Reason for absence



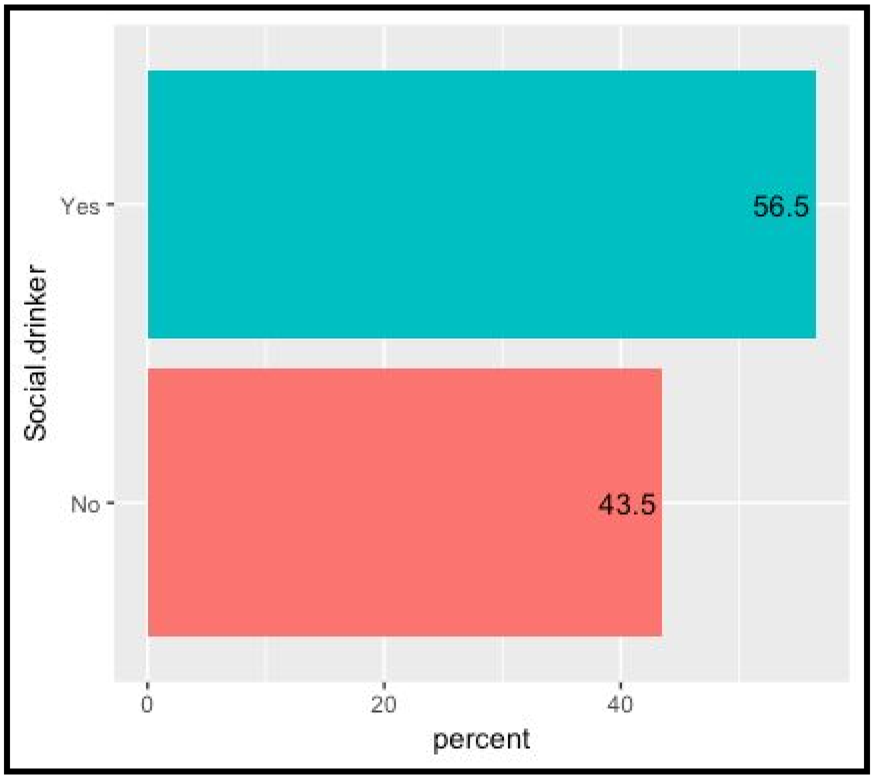
Day of the week



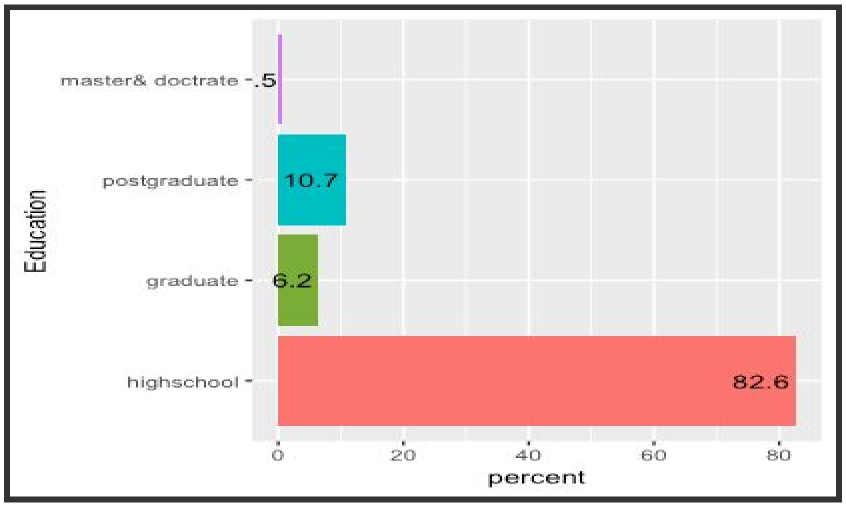
Social smoker



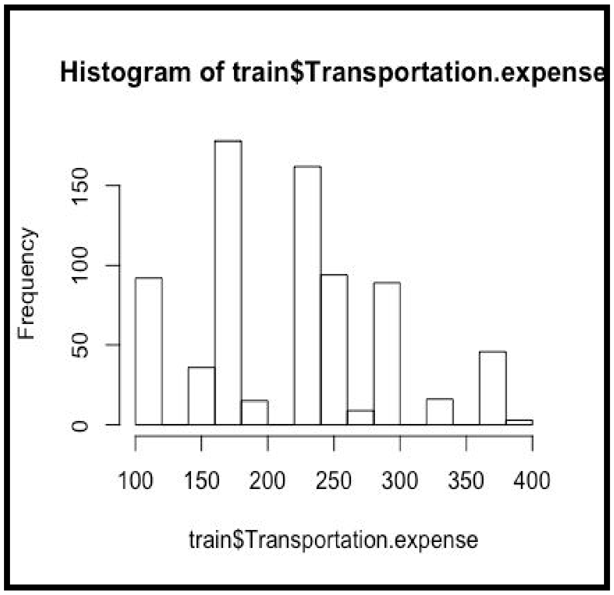
Social.drinker

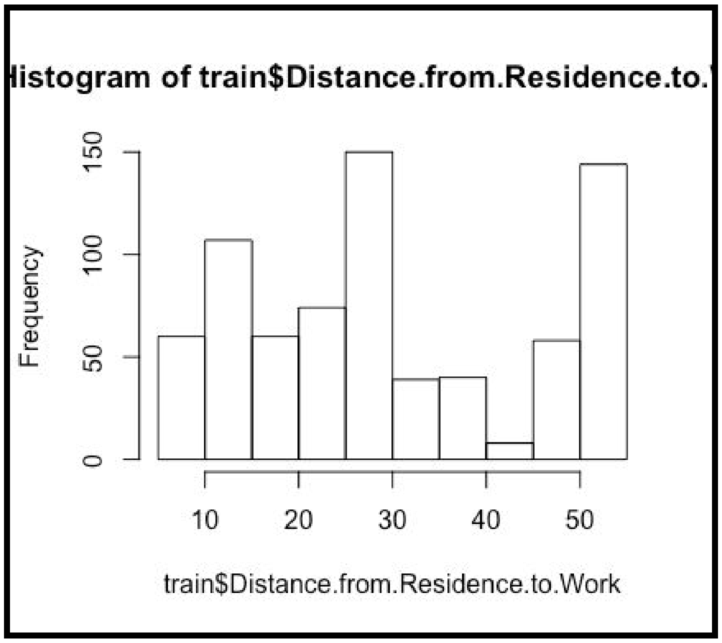


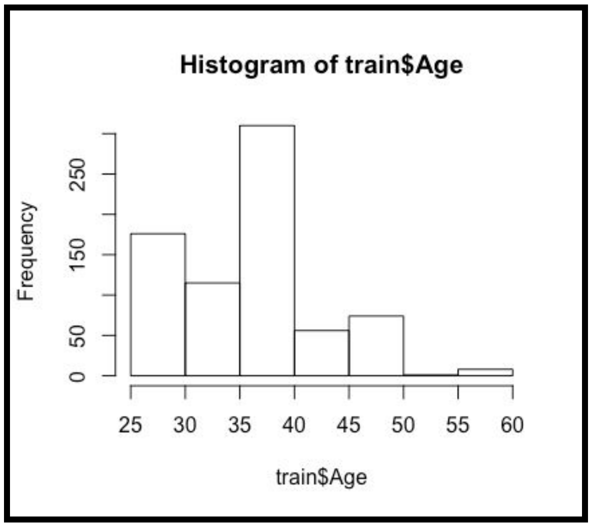
Education

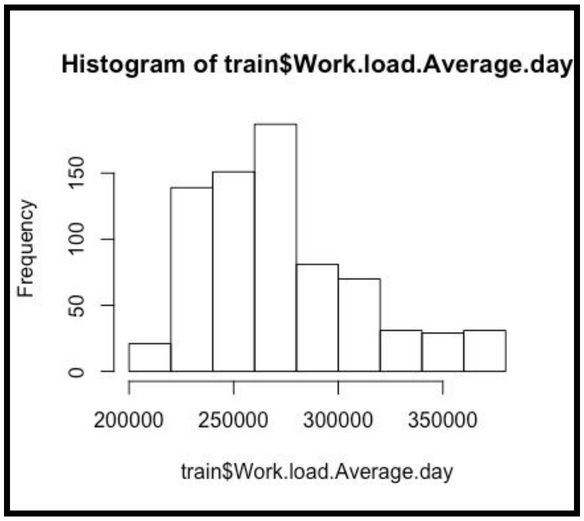


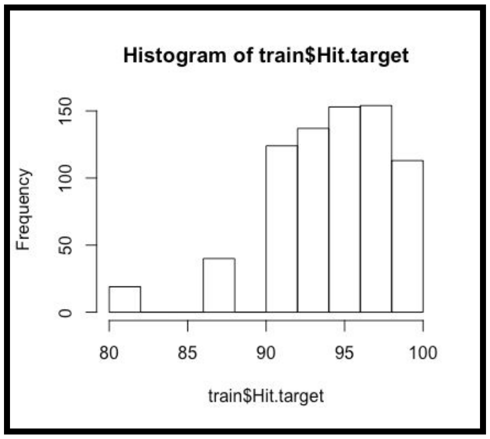
Histogram distribution of continuous variables:

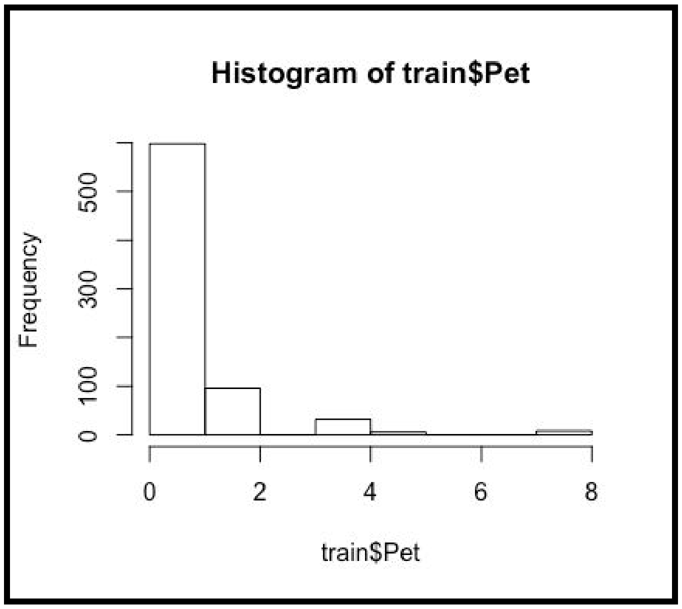


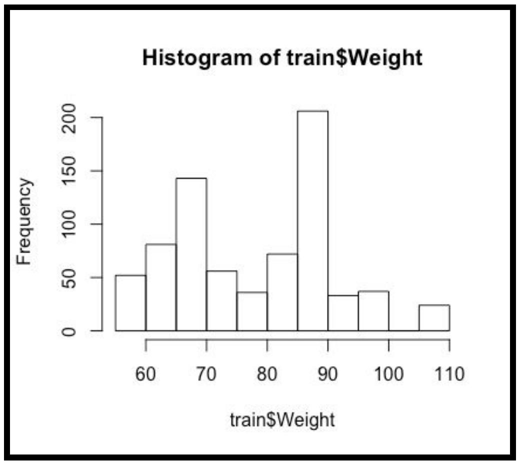

listogram of train$Distance.from.Residence.to.
Frequency
0
50
100
150
10
20
30
40
50
train$Distance.from.Residence.to.Work

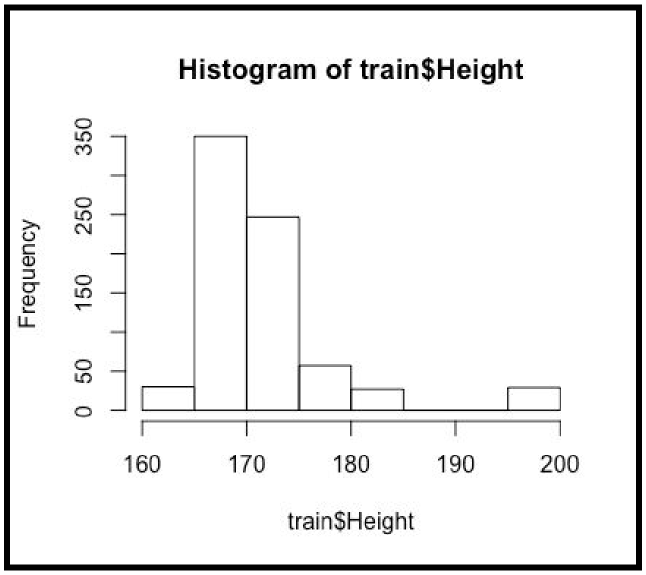

Histogram of train$Age
Frequency
train$Age
0
50
150
250
25
30
35
40
45
50
55
60

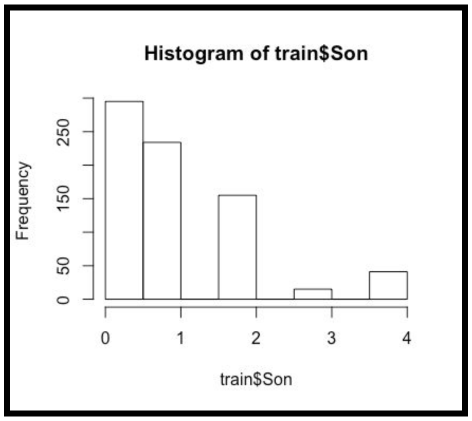

Histogram of train$Work.load.Average.day
Frequency
train$Work.load.Average.day
0
50
100
150
200000
250000
300000
350000

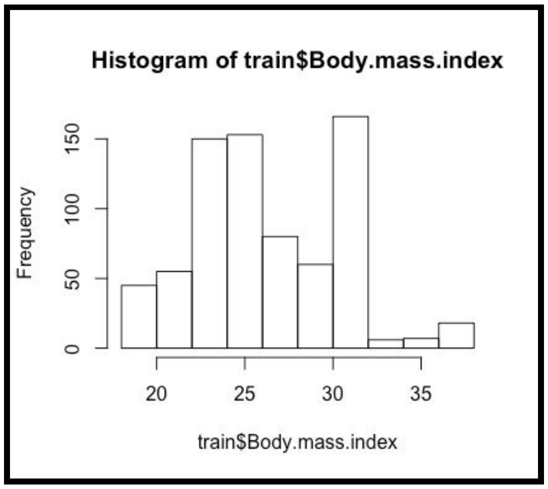

Histogram of train$Hit.target
Frequency
train$Hit.target
0
50
100
150
80
85
90
95
100

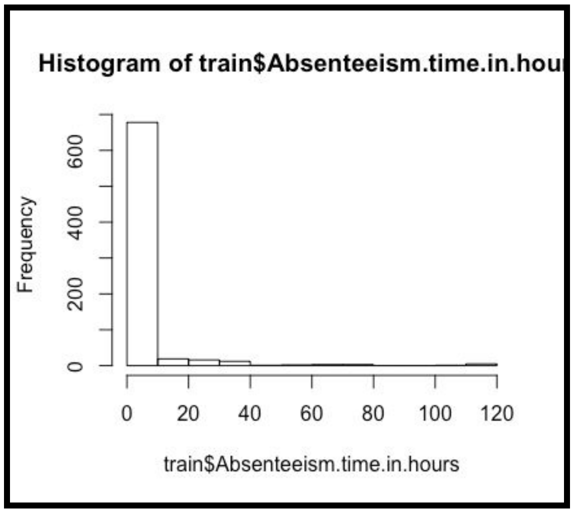

Histogram of train$Pet
Frequency
train$Pet
0
100
300
500
0
2
4
6
8

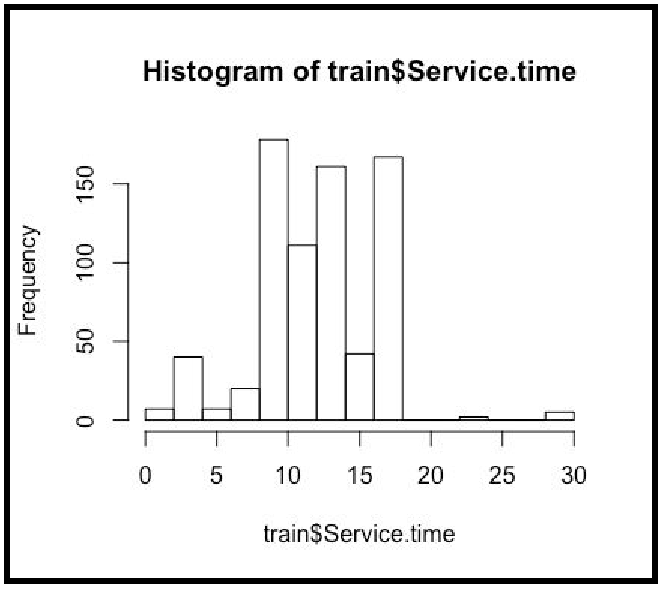

Histogram of train$Weight
Frequency
0
50
100
150
200
60
70
80
90
100
110
train$Weight


Histogram of train$Height
Frequency
0
50
150
250
350
160
170
180
190
200
train$Height


Histogram of train$Son
Frequency
0
50
150
250
0
1
2
3
4
train$Son

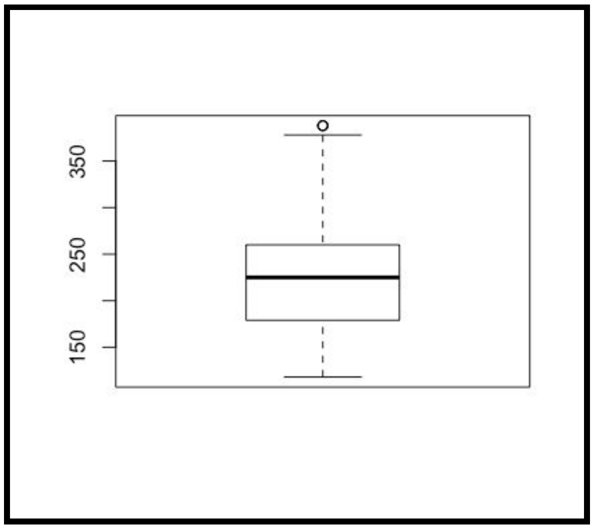

Histogram of train$Body.mass.index
Frequency
0
50
100
150
20
25
30
35
train$Body.mass.index


Histogram of train$Absenteeism.time.in.hou
Frequency
0
200
400
600
0
20
40
60
80
100
120
train$Absenteeism.time.in.hours

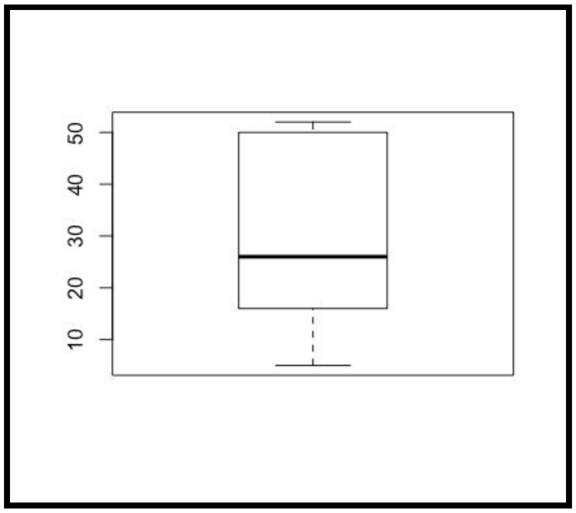

Histogram of train$Service.time
Frequency
0
50
100
150
0
5
10
15
20
25
30
train$Service.time

Boxplot of data

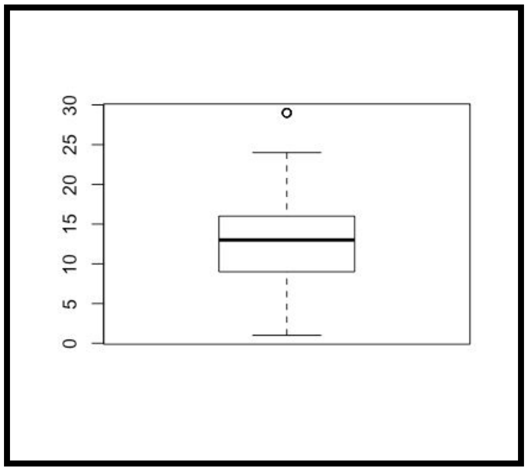
Transportation.expense



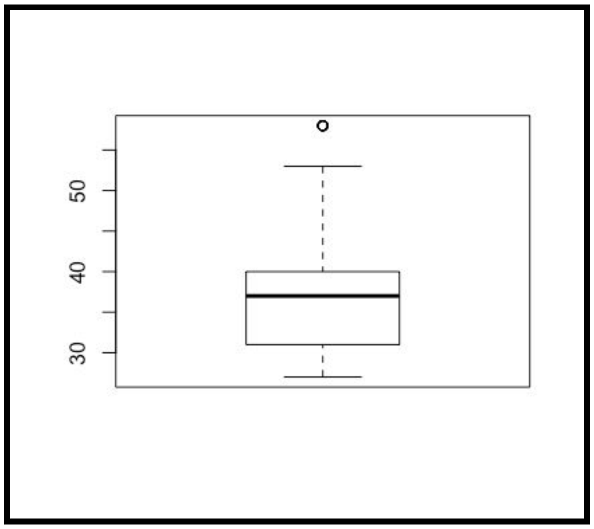
Distance.from.Residence.to.Work



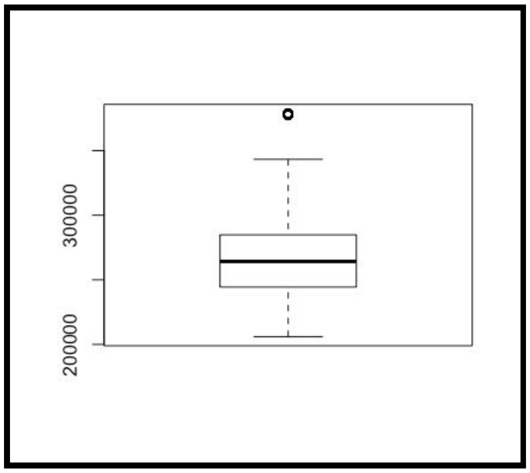
Service time



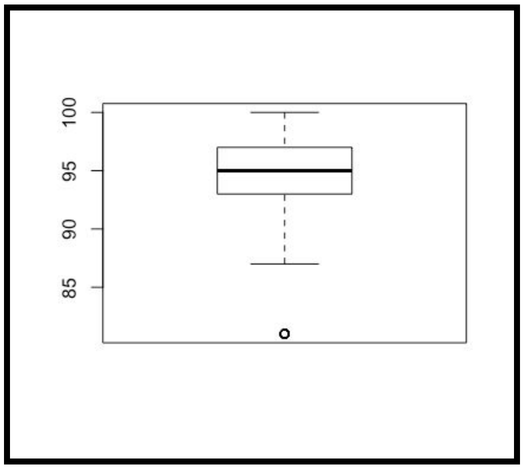
Age



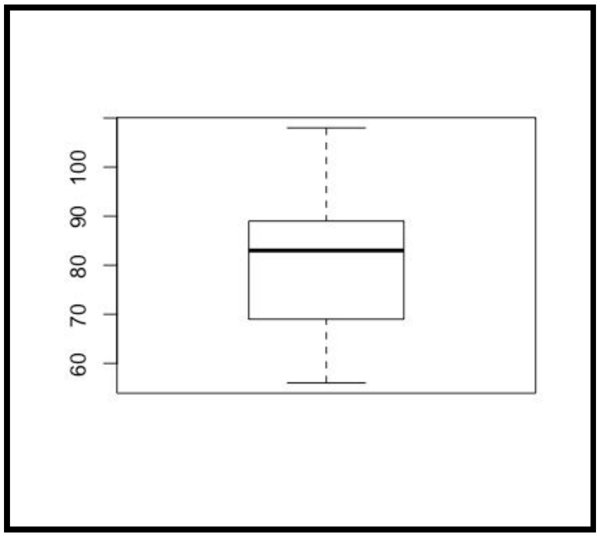
Workload



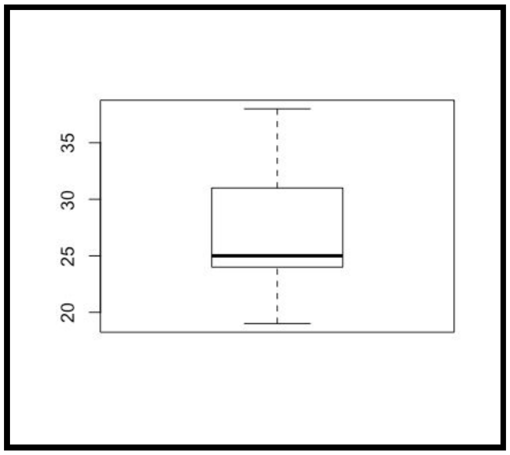
Hit target



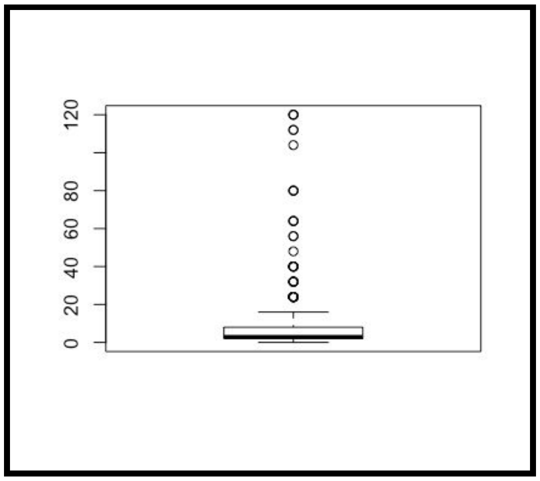
Weight



Body mass index



Absenteeism(target)
Outliers are there in the data



Anova test:
An ANOVA test is a way to find out if survey or experiment results are significant. In other words, they help you to figure out if you need to reject the null hypothesis or accept the alternate hypothesis. Basically, you're testing groups to see if there's a difference between them. Examples of when you might want to test different groups

When there are two or more categorical factors in our model, we again may want to test various single degrees-of-freedom hypotheses that compare various levels of the two or more factors in the model. As with the one-way ANOVA model, how you parameterize your two-way or higher model affects how you go about performing individual tests.

ANOVA assume continuous values in the dependent variable, but categorical variables as the independent variables

P-value
To determine whether any of the differences between the means are statistically significant, compare the **p-value** to your significance level to assess the null hypothesis. The null hypothesis states that the population means are all equal.

Usually, a significance level of 0.05 works well.

Reason.for.absence.************** null hypothesis is rejected**

```
Anova Table (Type II tests)

Response: da$Absenteeism.time.in.hours
                  Sum Sq  Df F value    Pr(>F)
Reason.for.absence  23308  26  5.1772 1.039e-14 ***
Residuals          104414 603
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Month.of.absence

```
Anova Table (Type II tests)
Response: da$Absenteeism.time.in.hours
                 Sum Sq Df F value  Pr(>F)
Month.of.absence   3253  12  1.5479   0.1022
Residuals        127313 727
```

Social.smoker

```
Anova Table (Type II tests)

Response: da$Absenteeism.time.in.hours
              Sum Sq    Df      F value  Pr(>F)
Social.smoker    206   1  1.1671     0.2803
Residuals     130359 738
Anova Table (Type II tests)
```

Social.drinker ******* null hypothesis is rejected**

```
Response: da$Absenteeism.time.in.hours
            Sum Sq  Df F value                      Pr(>F)
Social.drinker    517   1  2.9311         0.08731 .
Residuals      130049 738
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Disciplinary.failure

```
Anova Table (Type II tests)

Response: da$Absenteeism.time.in.hours
                 Sum Sq  Df F value                            Pr(>F)
Disciplinary.failure    454   1  2.5755           0.109
Residuals           130111 738
```

Chapter 3:

Data preprocessing:

Missing value imputation

In statistics, imputation is the process of replacing missing data with substituted values. When substituting for a missing value point, it is known as missing value imputation

There are two ways to deal with missing values
1.Removing empty places
2.Imputing missing values with median or mean of the column

I have opted the second one and it has given me good results

Outlier removal

There exist some outliers in the model without replacing that outliers it has given me unacceptable results

So as the number of outliers in target variable is very less
i have replaced values greater than 0.95 quantile with value of 0.95 quantile
I have replaced values less than 0.05 quantile with value of 0.05 quantile

Data scaling

As the independent variables in the data have different ranges.passing the data with that ranges has produced worst results
As we know model does not identify how it was measured and in what unit it was measured ,it just identifies the numbers as points

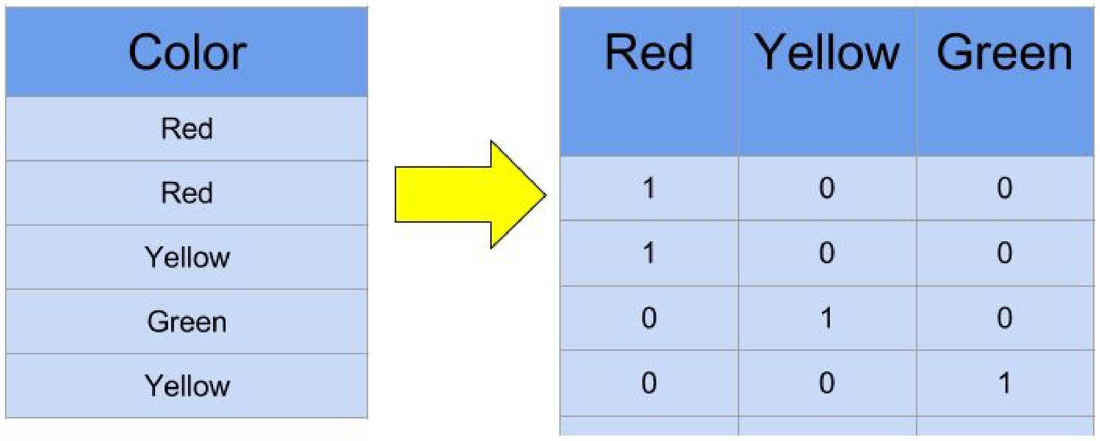
So i have brought the range of each column to **0 to 1 (except target variable)**
I have done **min max scaling** for each column and have brought all of them in to same range

One hot encoding of factor variables

One hot encoding is a representation of categorical variables as binary vectors. This first requires that the categorical values be mapped to integer values.

Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1
I have done one hot encoding of categorical data

So the number of binary columns depend on number of factor levels in the raw column
Now all the data is converted to numeric features
By one hot encoding we will convert the factor to binary columns which is numeric in nature
For example



Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Removing correlated features from the data:

Correlation is a statistical technique that can show whether and how strongly pairs of variables are related.

The main result of a correlation is called the **correlation coefficient** . It ranges from -1.0 to +1.0. The closer r is to +1 or -1, the more closely the two variables are related.

If r is close to 0, it means there is no relationship between the variables. If r is positive, it means that as one variable gets larger the other gets larger. If r is negative it means that as one gets larger, the other gets smaller (usally called an inverse correlation).

While correlation coefficients are normally reported as r = (a value between -1 and +1), squaring them makes then easier to understand. The square of the coefficient (or r square) is equal to the percent of the variation in one variable that is related to the variation in the other. After squaring r, ignore the decimal point. An r of .5 means 25% of the variation is related (.5 squared =.25). An r value of .7 means 49% of the variance is related (.7 squared = .49).

I have removed highly correlated features (having high correlation coefficient [>0.95])

Chapter 4

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.

I have used Boruta package in r
And selectKbest in python

Boruta

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

seleckKBest:

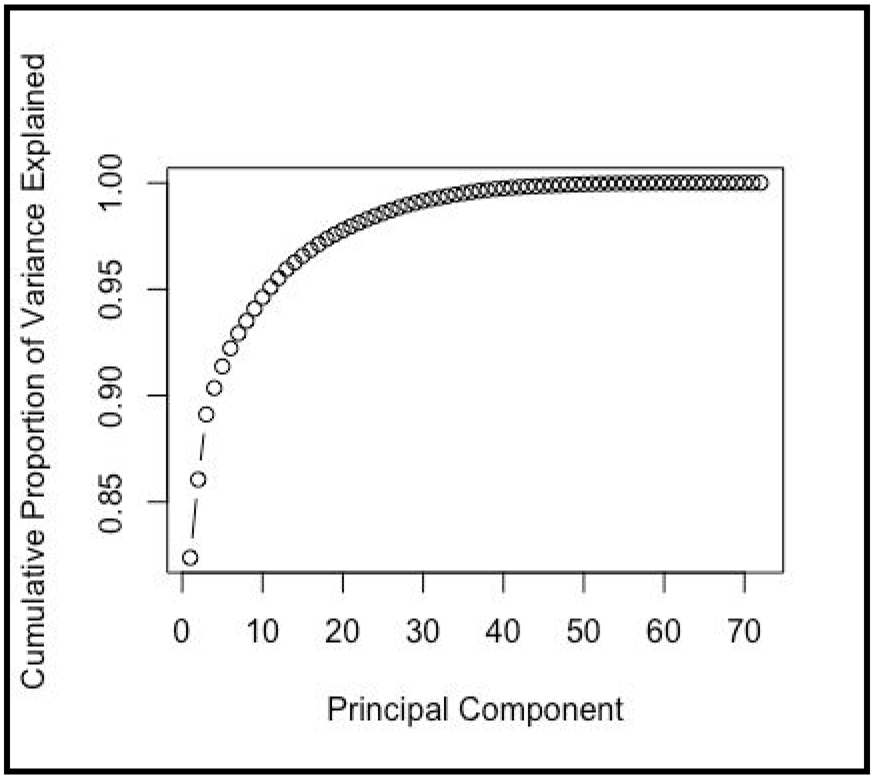
SelectKBest selects the top k features that have maximum relevance with the target variable. It takes two parameters as input arguments, "k" (obviously) and the score function to rate the relevance of every feature with the target variable. For example, for a regression problem, you can supply

Select features according to the k highest scores. The score function must return an array of scores, one for each feature (additionally, it can also return p-values, but these are neither needed nor required). SelectKBest then simply retains the first k features with the highest scores.

Principal component analysis:

As we have more features in the data(new columns obtained from one hot encoding) the dimension will be large.
So i have used Principal component analysis to reduce the dimension of the data keeping the variance unchanged
Principal component analysis (PCA) is a technique used to emphasize variation and bring out strong patterns in a dataset. It's often used to make data easy to explore and visualize.

Principal component analysis is a technique for feature extraction — so it combines our input variables in a specific way, then we can drop the least important variables while still retaining the most valuable parts of all of the variables! As an added benefit, each of the new variables after PCA are all independent of one another. This is a benefit because the assumptions of a linear model require our independent variables to be independent of one another. If we decide to fit a linear regression model with these new variables (see principal component regression below), this assumption will necessarily be satisfied.



From the above plot ~40 variables clearly explains the almost 100% of variance in the data

So PCA has reduced 72 variables to 40 without compromising the variance in the data

Sampling methods
K-fold repeated CV

- k-fold cross-validation randomly divides the data into k blocks of roughly equal size. Each of the blocks is left out in turn and the other k-1 blocks are used to train the model. The held out block is predicted and these predictions are summarized into some type of performance measure (e.g. accuracy, root mean squared error (RMSE), etc.). The k estimates of performance are averaged to get the overall resampled estimate. k is 10 or sometimes 5.

- Repeated k-fold CV does the same as above but more than once. For example, five repeats of 10-fold CV would give 50 total resamples that are averaged. Note this is not the same as 50-fold CV.

Chapter 5:

Multiple Linear Regression:
Multiple linear regression (MLR) is a statistical technique that uses several explanatory variables to predict the outcome of a response variable.
The goal of multiple linear regression (MLR) is to model the relationship between the explanatory and response variables.
Multiple linear regression (MLR) is used to determine a mathematical relationship among a number of random variables.
In other terms, MLR examines how multiple independent variables are related to one dependent variable.
Once each of the independent factors have been determined to predict the dependent variable, the information on the multiple variables can be used to cr
eate an accurate prediction on the level of effect they have on the outcome variable. The model creates a relationship in the form of a straight line (linear) that best approximates all the individual data points.

The model for MLR, given n observations, is:
$y_i = B_0 + B_1x_{i1} + B_2x_{i2} + ... + B_px_{ip} + E$ where i = 1,2, ..., n

Decision tree

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, decision tree algorithm can be used for solving regression and classification problems too.
The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by learning decision rules inferred from prior data(training data).

The understanding level of Decision Trees algorithm is so easy compared with other classification algorithms. The decision tree algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label.

Decision Tree Algorithm Pseudocode

1. Place the best attribute of the dataset at the root of the tree.
2. Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for an attribute.

3. Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree.

Random forest

A group of decision tree is nothing but a random forest

The Random Forest is one of the most effective machine learning models for predictive analytics, making it an industrial tool for machine learning.

Background process

The random forest model is a type of additive model that makes predictions by combining decisions from a sequence of base models. More formally we can write this class of models as:

$$g(x)=f0(x)+f1(x)+f2(x)+...$$

where the final model g is the sum of simple base models f_i. Here, each base classifier is a simple decision tree. This broad technique of using multiple models to obtain better predictive performance is called model ensembling. In random forests, all the base models are constructed independently using a different subsample of the data.

Lasso regression:

Lasso is another extension built on regularized linear regression, but with a small twist

The only difference from Ridge regression is that the regularization term is in absolute value. But this difference has a huge impact.Lasso method overcomes the disadvantage of Ridge regression by not only punishing high values of the coefficients β but actually setting them to zero if they are not relevant. Therefore, we might end up with fewer features included in the model than you started with, which is a huge advantage.

Ridge regression:

Ridge regression is an extension for linear regression. It’s basically a regularized linear regression model. The λ parameter is a scalar that should be learned as well, using a method called cross validation that will be discussed in another post.

A super important fact we need to notice about ridge regression is that it enforces the β coefficients to be lower, but it does not enforce them to be zero. That is, it will not get rid of irrelevant features but rather minimize their impact on the trained model.

Chapter 6:

Model results with feature selection:

Linear Regression

```
500 samples
 26 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 449, 450, 451, 450, 449, 449, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.6869893  0.4567036  0.5198062

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

Random forest

```
Random Forest

550 samples
 26 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 496, 495, 495, 495, 494, 496, ...
Resampling results across tuning parameters:

  mtry  RMSE       Rsquared   MAE
   2    0.7467292  0.3877560  0.5872196
  14    0.7179312  0.4141166  0.5374368
  26    0.7252704  0.4084543  0.5413293

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 14.
```

Decision tree
CART

```
550 samples
 26 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 495, 494, 495, 495, 495, 496, ...
Resampling results across tuning parameters:

  cp          RMSE       Rsquared   MAE
  0.02410396  0.7207862  0.3937296  0.5436170
  0.06108719  0.7441314  0.3550788  0.5822147
  0.11057921  0.8565608  0.2563823  0.7007582

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was cp = 0.02410396.
```

After removing statistically insignificant variables i have passed these variables to the model(but it did not improve my results)
New variables

```
new=c('Reason.for.absence.1','Reason.for.absence.9',
  'Reason.for.absence.13','Reason.for.absence.19','Reason.for.absence.23',

'Reason.for.absence.25','Reason.for.absence.27','Reason.for.absence.28','Work.load.Average.day' ,'Disciplinary.failure.1' )
```

Linear Regression

```
740 samples
 10 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
```

```
Summary of sample sizes: 666, 666, 666, 665, 666, 666, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.6937021  0.4013057  0.5265712

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

Ridge Regression

```
740 samples
 10 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 667, 666, 667, 666, 665, 667, ...
Resampling results across tuning parameters:

  lambda  RMSE       Rsquared   MAE
  0e+00   0.6947670  0.3969520  0.5271024
  1e-04   0.6947668  0.3969514  0.5271143
  1e-01   0.6961806  0.3945434  0.5364535

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was lambda = 1e-04.
```

Random Forest

```
550 samples
 10 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 495, 495, 496, 496, 494, 494, ...
Resampling results across tuning parameters:

  mtry  RMSE       Rsquared   MAE
   2    0.7333247  0.4110335  0.5932519
   6    0.7065413  0.4221801  0.5387507
  10    0.7465085  0.3726460  0.5708008
```

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 6.

Decision tree
CART

```
550 samples
 10 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 494, 494, 494, 495, 496, 496, ...
Resampling results across tuning parameters:

  cp          RMSE       Rsquared   MAE
  0.01720696  0.7218776  0.3950959  0.5473875
  0.06108719  0.7455863  0.3538757  0.5831339
  0.11057921  0.8682911  0.2473977  0.7124916
```

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was cp = 0.01720696.

PCA results:
I have applied PCA to my data and it has given me very good results

Linear Regression

```
550 samples
 40 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 495, 495, 497, 495, 495, 494, ...
Resampling results:

  RMSE          Rsquared  MAE
  0.0003649108  1         0.0001897937

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

Ridge Regression

```
550 samples
 40 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 495, 494, 495, 494, 495, 497, ...
Resampling results across tuning parameters:

  lambda  RMSE          R-squared  MAE
  0e+00   0.0003720586  1.0000000  0.0001906134
  1e-04   0.0003922922  1.0000000  0.0002138447
  1e-01   0.0731254004  0.9999014  0.0537276425

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was lambda = 0.
```

Random Forest

```
550 samples
 40 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 495, 495, 495, 496, 494, 495, ...
Resampling results across tuning parameters:

  mtry  RMSE        Rsquared   MAE
   2    3.48931619  0.8015626  2.30151504
  21    0.46298332  0.9947362  0.24544708
  40    0.02402738  0.9999369  0.00416798

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 40.
```

Decision tree

CART

```
550 samples
 40 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 494, 496, 494, 495, 495, 497, ...
Resampling results across tuning parameters:

  cp          RMSE      Rsquared   MAE
  0.03859736  1.232329  0.9501225  0.8017971
  0.19782997  2.142400  0.8308691  1.7485723
  0.73891643  4.617210  0.6913729  3.5226207

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was cp = 0.03859736.
```

The lasso

```
550 samples
 40 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 494, 496, 495, 494, 495, 495, ...
Resampling results across tuning parameters:

  fraction  RMSE       Rsquared   MAE
  0.1       5.0830384  0.9999767  3.7081262
  0.5       2.7804990  0.9999767  2.0270920
  0.9       0.4785491  0.9999767  0.3460578

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

Python results:

Before applying PCA

model_results(Ridge_model)

```
test-RMSE of data with outliers
10.1997039544
coefficient of determination R^2 of the prediction
0.0415500192233
test-RMSE of outlier removed data(clean)
4.7402286468
coefficient of determination R^2 of the prediction
0.249558527724
```

model_results(rf_model)

```
test-RMSE of data with outliers
11.4892695485
coefficient of determination R^2 of the prediction
```

-0.216127633426
test-RMSE of outlier removed data(clean)
4.64730377075
coefficient of determination R^2 of the prediction
0.278692633047

model_results(lasso_model)
test-RMSE of data with outliers
10.3809895377
coefficient of determination R^2 of the prediction
0.00717700698425
test-RMSE of outlier removed data(clean)
5.47381776305
coefficient of determination R^2 of the prediction
-0.000688513854519

model_results(DT_model)
test-RMSE of data with outliers
12.323464562
coefficient of determination R^2 of the prediction
-0.399136095773
test-RMSE of outlier removed data(clean)
5.22906547483
coefficient of determination R^2 of the prediction
0.0867989433428

model_results(lin_reg_model)
test-RMSE of data with outliers
10.3987053503
coefficient of determination R^2 of the prediction
0.00378548555666
test-RMSE of outlier removed data(clean)
4.78595415439
coefficient of determination R^2 of the prediction
0.235010782318

model_results(lars_model)
test-RMSE of data with outliers
10.3987053503

coefficient of determination R^2 of the prediction
0.00378548555666
test-RMSE of outlier removed data(clean)
4.78595415439
coefficient of determination R^2 of the prediction
0.235010782318

Python PCA results:

pca_model_results(Ridge_model)
RMSE on trained data of PCA model
0.000710964495848
test-RMSE PCA model
0.000668459390511
coefficient of determination R^2 of the prediction
0.999999985077

pca_model_results(rf_model)
RMSE on trained data of PCA model
0.0299210673317
test-RMSE PCA model
0.0183803655523
coefficient of determination R^2 of the prediction
0.999988716949

pca_model_results(DT_model)
RMSE on trained data of PCA model
0.0
test-RMSE PCA model
0.0
coefficient of determination R^2 of the prediction
1.0

pca_model_results(lin_reg_model)
RMSE on trained data of PCA model
0.000604623781034
test-RMSE PCA model
0.000587157470621

coefficient of determination R^2 of the prediction
0.999999988486

pca_model_results(lars_model)
RMSE on trained data of PCA model
0.000604623781034
test-RMSE PCA model
0.000587157470621
coefficient of determination R^2 of the prediction
0.999999988486

Comparing results of the models

Results in r(feature selection using ‘Boruta’)

Model	RMSE	R-squared
linear	7.342354	0.7442699~ 74.4%
Randomforest	6.015931	0.8158162 ~ 81.5%
Decision tree	6.556025	0.7657387 ~ 76.57%

PCA results in R

Model	RMSE	R-squared
linear	0.0003649108	1 ~ 100%
ridge	0.0003720586	1 ~ 100%
Randomforest	0.02402738	0.9999369 ~ 100%
Decision tree	0.05726378	0.9998252 ~ 100%
Lasso	0.4785491	0.9999768 ~ 100%

Python results(feature selection using 'selectKbest')

Model	RMSE	R-square
linear	4.78595415439	0.235010782318 ~ 23.5%
ridge	4.7402286468	0.249558527724 ~ 24.9%
Randomforest	4.64730377075	0.278692633047 ~ 27.8%
Decision tree	5.22906547483	0.0867989433428 ~ 8.6%
Lasso	5.47381776305	-0.000688513854519 ~0%
lars	4.78595415439	0.235010782318 ~ 23.5%

PCA results in python

Model	RMSE	R-Square
linear	0.000604623781034	0.999999988486~ 100%
ridge	0.000710964495848	0.999999985077 ~ 100%
Randomforest	0.0299210673317	0.999988716949 ~ 100%
Decision tree	0	1~ 100%
Lasso	0.170942829236	0.999024066961 ~ 100%
lars	0.000587157470621	0.999999988486 ~ 100%

So over all PCA has produced the best results for the above problem

NOTE:RMSE that I have provided in the above tables is of training data.
RMSE of test data is less than the training data (the PCA models has achieved good results and it has not overfitted) with r_squared value of 100%.

Selecting the best fitted model:

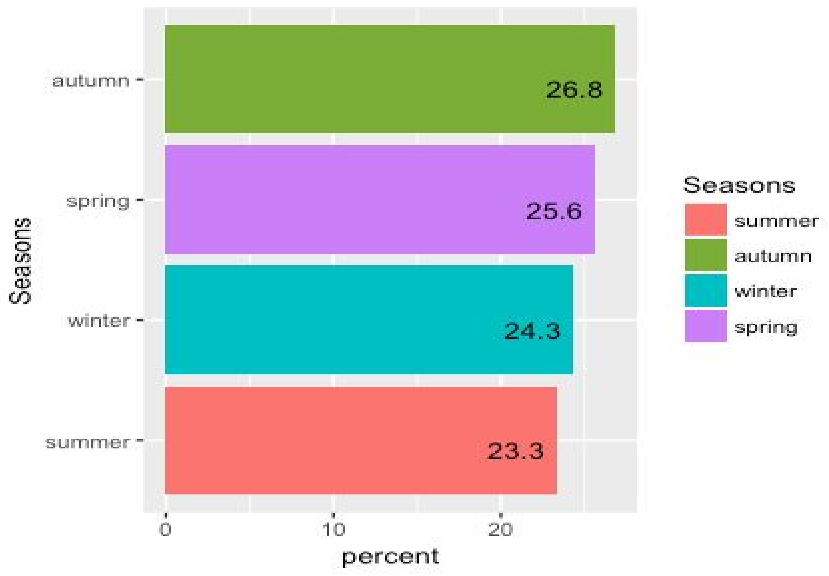
Every model using PCA is giving the best results

Chapter 7:

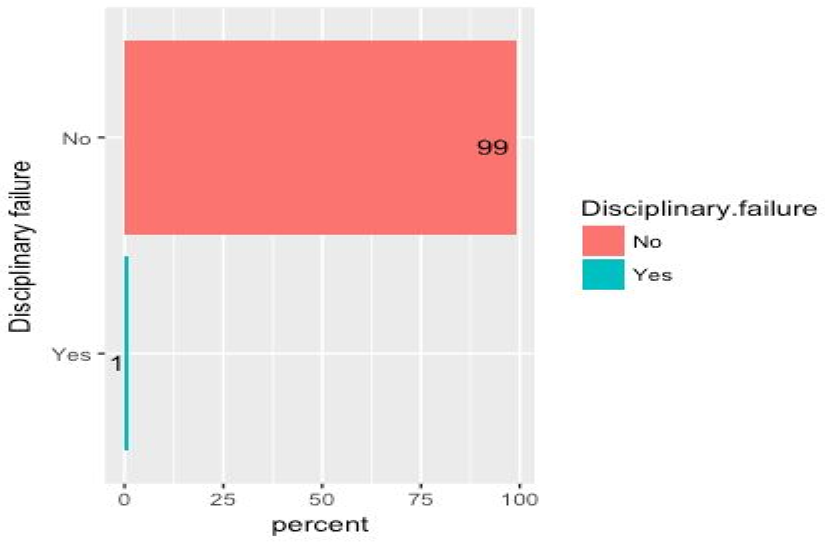
1.What changes company should bring to reduce the number of absenteeism?

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

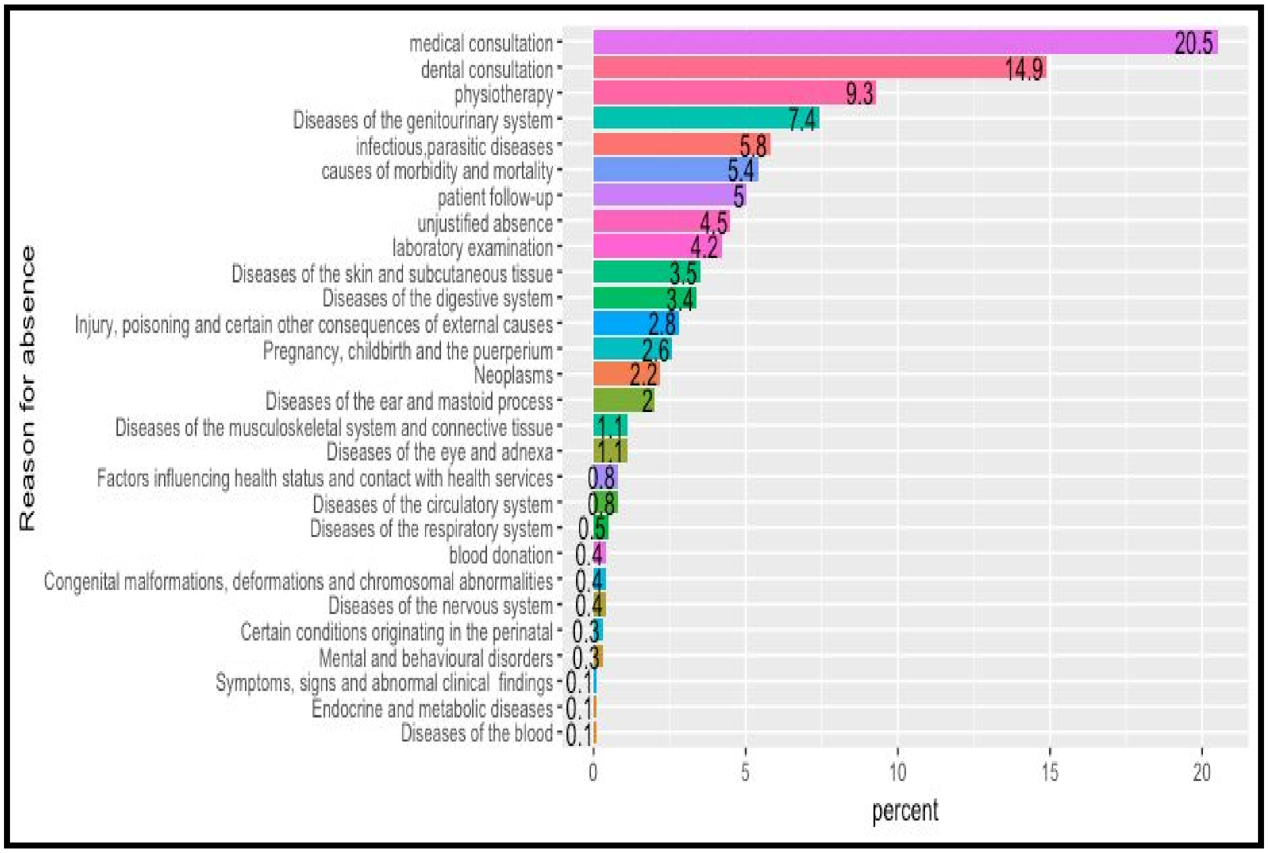
Seasons



Disciplinary failure

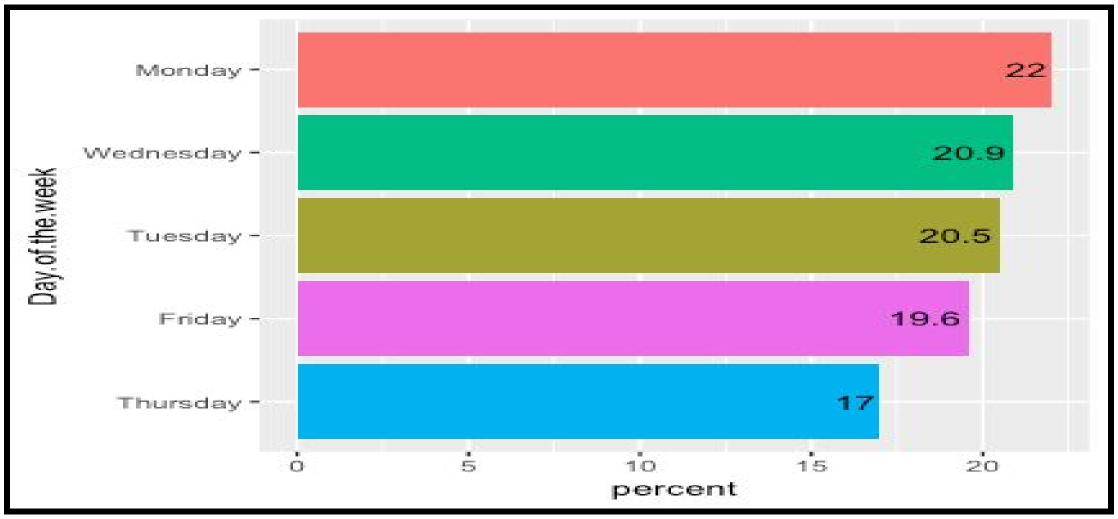


Reason for absence

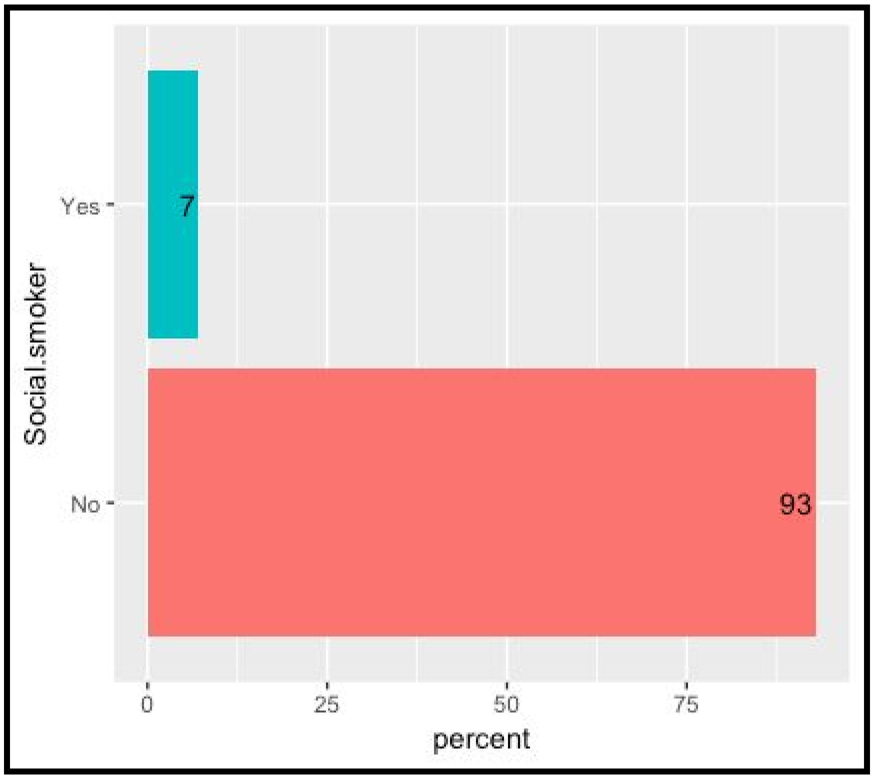


******People who has medical consultation ,dental consultation,physiotherapy,diseases of the genitourinary system contribute more than 50% absenteeism hours
So company should take necessary care to the workers who are suffering with above diseases to reduce number of absenteeism absenteeism

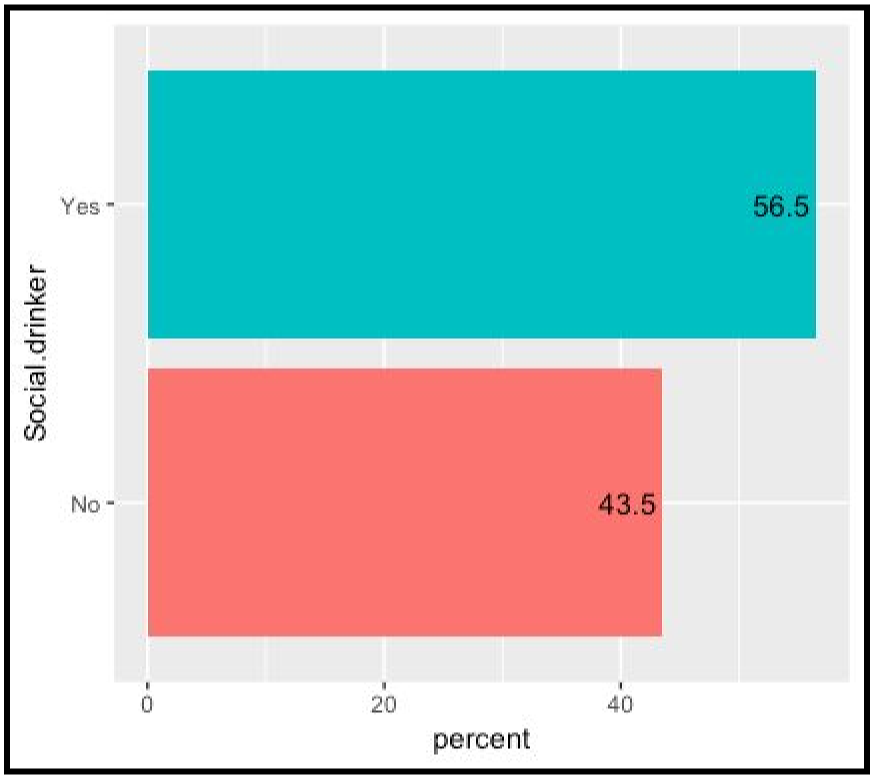
Day of the week



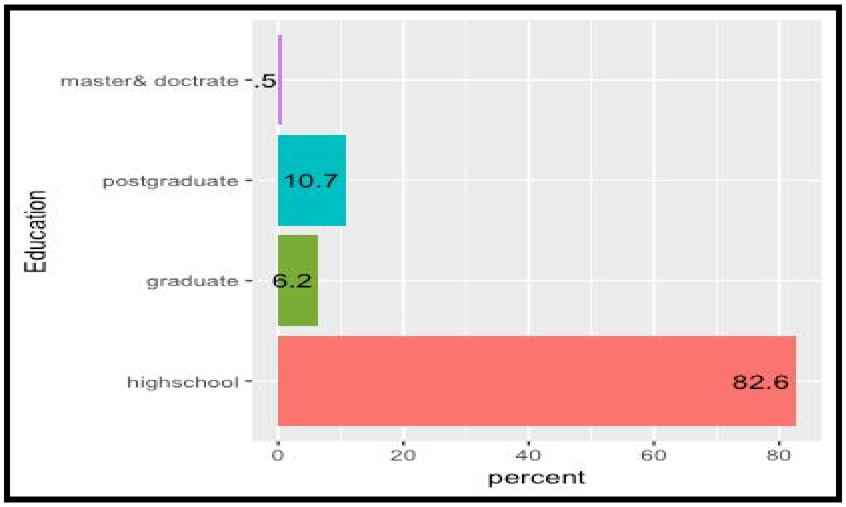
Social smoker



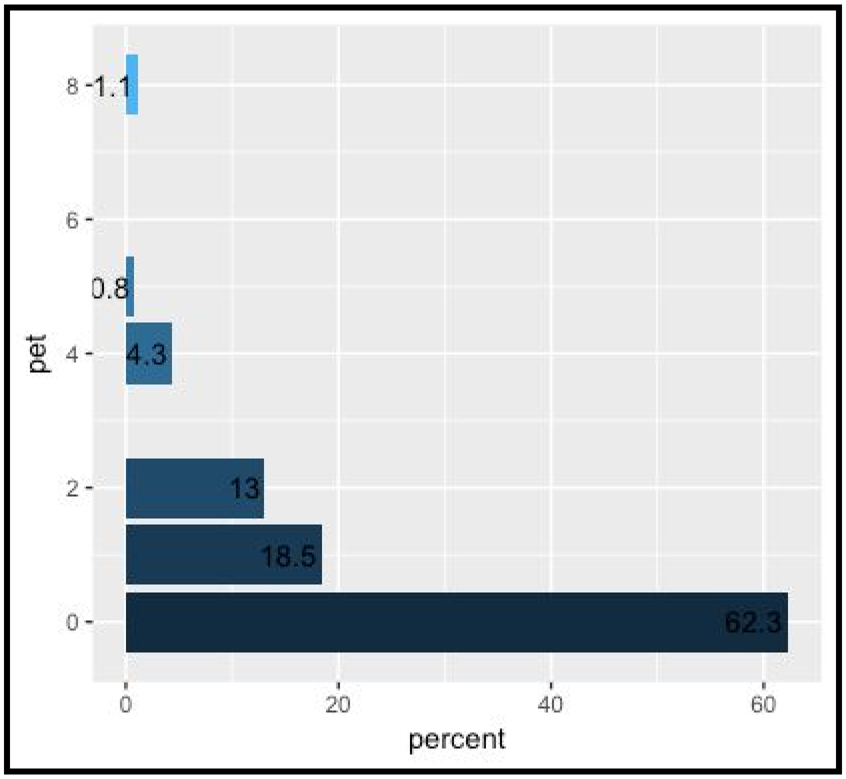
Social.drinker

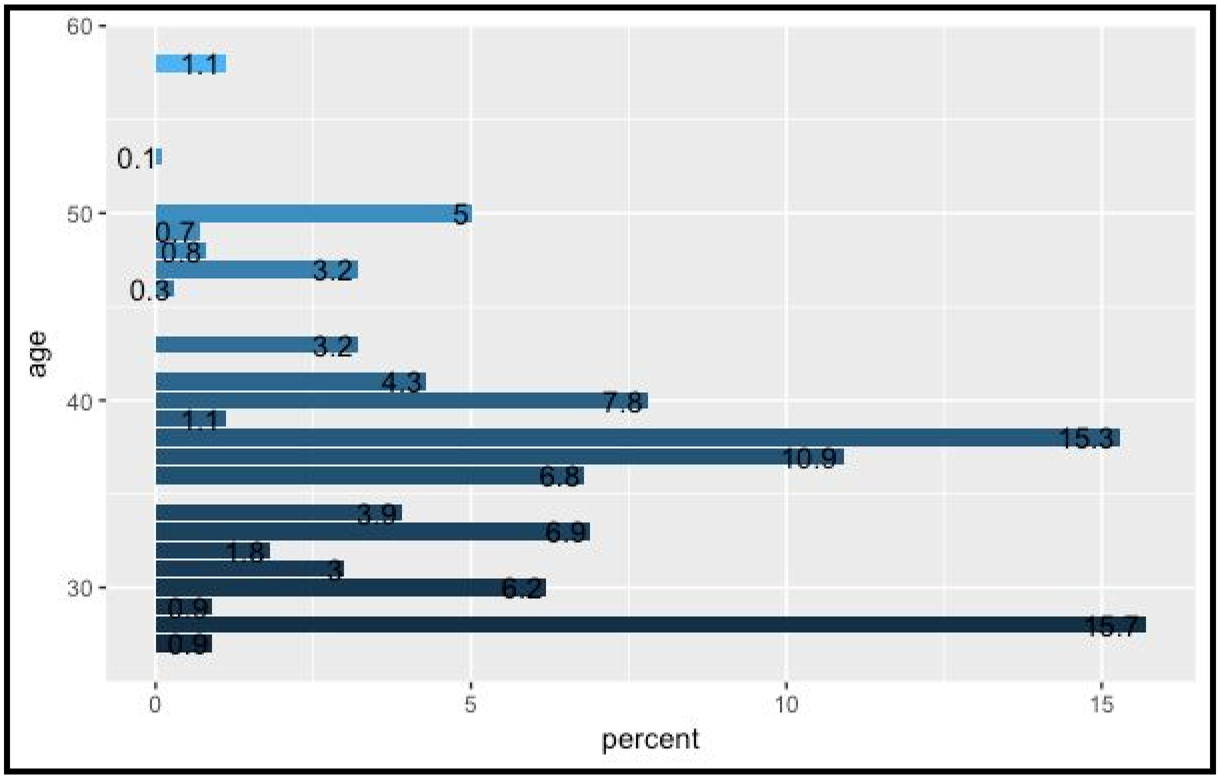


******People who are social drinker have more number of absent hours

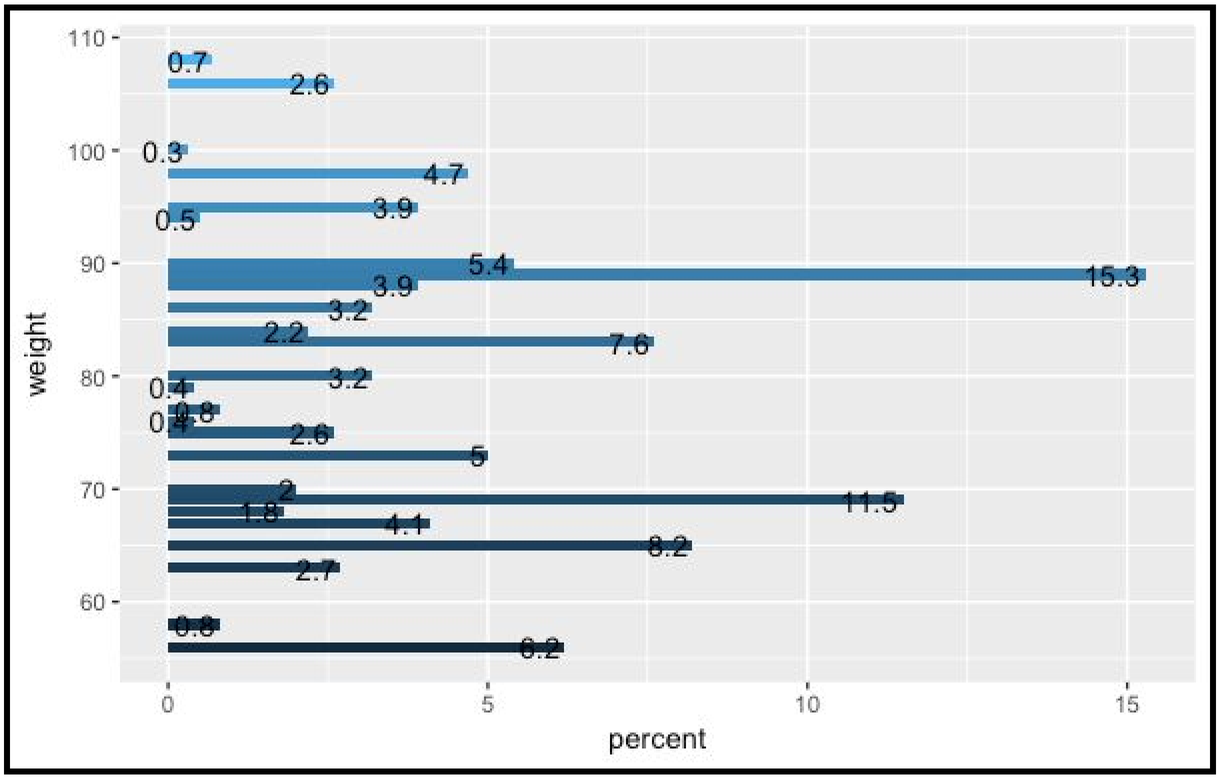


******People who have very low qualification level have more absent hours than others

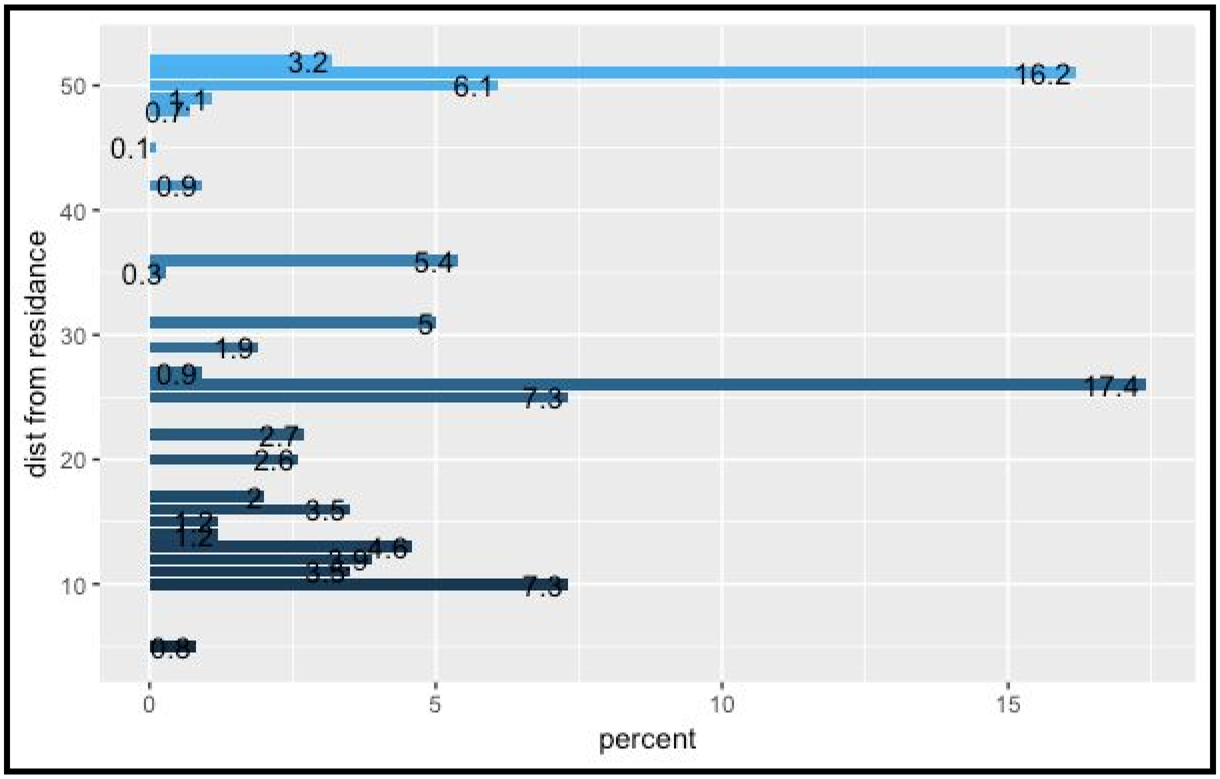




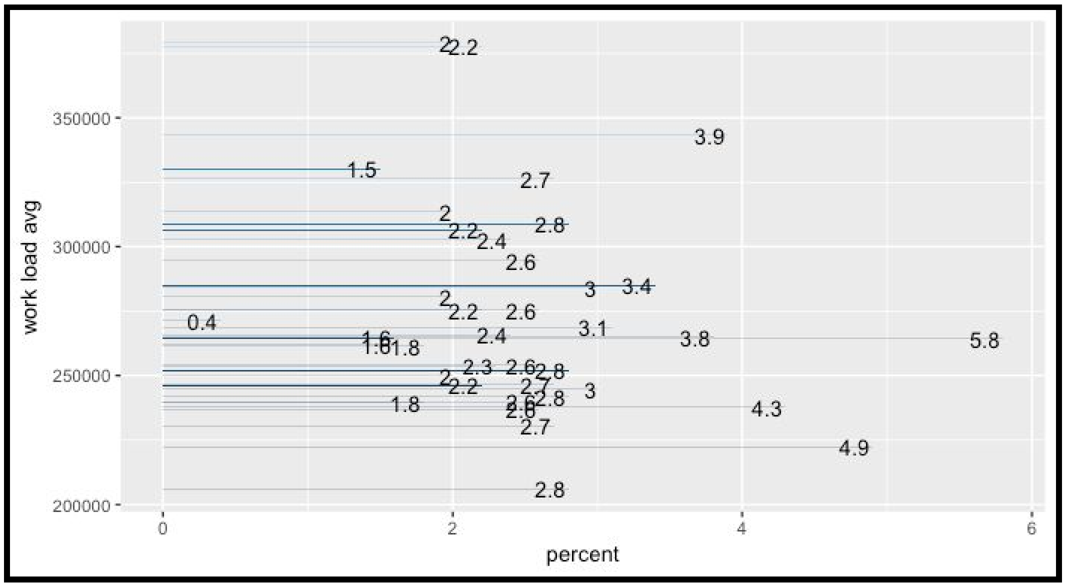
People whose age is between 25 to 40 has more absent hours almost 75% belongs to this age group



*****People with weight above 70 kgs has more absenteeism hours (>50%)



*****People who stay more than 20 km away from the company has higher absent hours



Conclusions drawn from above graphs:

1. 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

2. People who are social drinker have more number of absent hours

3. People who have very low qualification level have more absent hours than others
4. People whose age is between 25 to 40 has more absent hours almost 75% belongs to this age group
5. People with weight above 70 kgs has more absenteeism hours (>50%)

6. People who stay more than 20 km away from the company has higher absent hours

Things to do to reduce absenteeism hours:

1. company should take necessary care to the workers who are suffering with above diseases to reduce number of absenteeism absenteeism.medical consultation ,dental consultation,physiotherapy,diseases of the genitourinary system contribute more than 50% absenteeism hours
2. People who has low level educational qualification are being more absent.so company should talk to the workers regarding what the exact problem is**(could not find appropriate reason for this problem,it ultimately depends on workers).**
3. Age group between 25 and 40 has almost 75% absent hours.Its the age between youth and middle adulthood age.Company should take necessary things(**depending on their interest like fun activities,motivational speeches..**) to make sure that their absent hours should reduce
4. 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(providing healthy food ,having some exercise sessions ,awareness about having more weight will lead to what kind of issues)
5. Company should provide necessary transportation for the people who stay(**>20 km**) far from workplace.

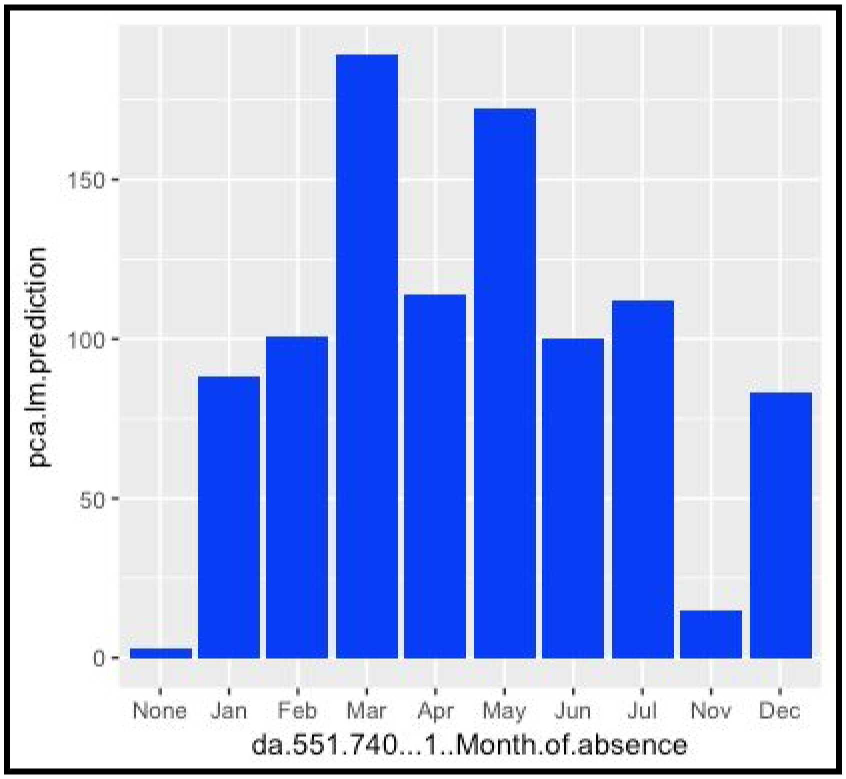
2.How much losses every month can we project in 2011 if same trend of absenteeism continues?

*****As the company did not provide how much loss will occur if the person is absent by one hour in a month,I am plotting the results of predicted number of absent hours vs every month. *****

It tells the number of absent hours in different months can be be predicted if it follows the same trend

Total of 983(for 190 samples) absent hours predicted if it follows same trend in 2011

I am using MLR predictions as it has r-squared of 1 ~ 100% and very low RMSE ~0



Code:
R

```
library(readxl)
train= read_excel("Absenteeism_at_work_Project.xls")
train <- data.frame(train)
summary(train)
#install.packages('Hmisc')
library(Hmisc)
#missing value imputation
for (i in (1:21)){
  train[i]=impute(train[i],median)
}



#converting every column to numeric
col <- c(1:21)
train[col]=as.numeric(unlist(train[col]))

#Converting categorical data to factor type
for (i in c(1,2,3,4,5,12,13,15,16)) {
  train[,i]=as.factor(unlist(train[i]))
}


summary(train)
#boxplot(train)

#da <- train
#install.packages(c("forcats", "DataExplorer", "ggthemes",
"grid","gridExtra","factoextra","FactoMineR"))
#library(forcats)
#library(dplyr)
#library(ggplot2)
#library(plyr)
#library(DataExplorer)
#library(ggthemes)
install.packages("psych")
```

```
library(psych)
pairs.panels(train)
library(grid)
library(gridExtra)
library(factoextra)
library(FactoMineR)
da=train
library(dplyr)
```

#recoding categorical data

```
da$Reason.for.absence=factor(da$Reason.for.absence,levels =
c(0:19,21:28),labels = c('infectious,parasitic
diseases','Neoplasms','Diseases of the blood','Endocrine and
metabolic diseases','Mental and behavioural disorders',
                             'Diseases of the nervous
system','Diseases of the eye and adnexa','Diseases of the ear
and mastoid process',
                             'Diseases of the circulatory
system','Diseases of the respiratory system','Diseases of the
digestive system',
                             'Diseases of the skin and
subcutaneous tissue','Diseases of the musculoskeletal system and
connective tissue',
                             'Diseases of the genitourinary
system','Pregnancy, childbirth and the puerperium','Certain
conditions originating in the perinatal',
                             'Congenital malformations,
deformations and chromosomal abnormalities','Symptoms, signs and
abnormal clinical  findings',
                             'Injury, poisoning and certain
other consequences of external causes','causes of morbidity and
mortality',
                             'Factors influencing health status
and contact with health services','patient follow-up','medical
consultation','blood donation',
                             'laboratory
examination','unjustified absence','physiotherapy','dental
consultation'))
```

```
da$Month.of.absence=factor(da$Month.of.absence,levels=c(0:12),la
bels=c('None','Jan','Feb','Mar','Apr','May',

'Jun','Jul','Aug','Sep','Oct','Nov','Dec'))
da$Seasons <-
factor(da$Seasons,levels=c(1:4),labels=c('summer','autumn','wint
er','spring'))

da$Education <-
factor(da$Education,levels=c(1:4),labels=c('highschool','graduat
e','postgraduate','master& doctrate'))

da$Disciplinary.failure <- factor(da$Disciplinary.failure,levels
=c(0:1),labels=c('No','Yes'))
da$Social.drinker <- factor(da$Social.drinker,levels
=c(0:1),labels=c('No','Yes'))
da$Social.smoker <- factor(da$Social.smoker,levels
=c(0:1),labels=c('No','Yes'))
da$Day.of.the.week
<-factor(da$Day.of.the.week,levels=c(2:6),labels=c("Monday","Tue
sday","Wednesday","Thursday","Friday"))
```

# Data exploration-

```
p <- ggplot(da, aes(x = Education, fill = Education)) +
geom_bar()
s <- ggplot(da, aes(x =
Reason.for.absence,fill=Reason.for.absence)) + geom_bar()

SS <- ggplot(da, aes(x =  Social.smoker, fill =  Social.smoker))
+ geom_bar()
SD <- ggplot(da, aes(x =  Social.drinker, fill =
Social.drinker)) + geom_bar()
D <- ggplot(da, aes(x =  Disciplinary.failure, fill =
Disciplinary.failure)) + geom_bar()
month<- ggplot(da, aes(x =  Month.of.absence, fill =
Month.of.absence)) + geom_bar()
```

```
Day <- ggplot(da, aes(x = Day.of.the.week, fill =
Day.of.the.week)) + geom_bar()
S <- ggplot(da, aes(x =   Seasons,fill = Seasons)) + geom_bar()
plot(p)
plot(SS)
plot(S)
plot(SD)
plot(D)
plot(Day)
plot(month)
plot(s)

detach("package:plyr", unload=TRUE)
```

#selecting samples which have absenteeism time hours > 0

```
absent <- as.data.frame( da %>% select(everything()) %>%
filter(Absenteeism.time.in.hours > 0))
```

#percentage absenteeism hours(>0) vs seasons

```
season1 <- as.data.frame(absent %>% group_by(Seasons) %>%
summarise(count= n(), percent =
round(count*100/nrow(absent),1))%>% arrange(desc(count)))
ggplot(season1,aes(x= reorder(Seasons,percent), y= percent, fill
= Seasons)) + geom_bar(stat='identity') + coord_flip() +
  geom_text(aes(label = percent), vjust = 1.1, hjust = 1.2) +
xlab('Seasons')
```

#percentage absenteeism hours vs disciplinary failure

```
disciplinary <-as.data.frame(absent %>%
group_by(Disciplinary.failure) %>% summarise(count= n(), percent
= round(count*100/nrow(absent),1))%>% arrange(desc(count)))
ggplot(disciplinary,aes(x=
reorder(Disciplinary.failure,percent), y= percent, fill =
Disciplinary.failure)) + geom_bar(stat='identity') +
coord_flip() +
  geom_text(aes(label = percent), vjust = 1.1, hjust = 1.2) +
xlab('Disciplinary failure')
```

#percentage absenteeism hours vs Reason for absence

```
Reason <-  as.data.frame(absent %>% group_by(Reason.for.absence)
%>% summarise(count= n(), percent =
round(count*100/nrow(absent),1))%>% arrange(desc(count)))
ggplot(Reason,aes(x = reorder(Reason.for.absence,percent), y=
percent, fill= Reason.for.absence)) + geom_bar(stat =
'identity') + coord_flip() + theme(legend.position='none') +
  geom_text(aes(label = percent), vjust = 0.5, hjust = 1.1) +
xlab('Reason for absence')
```

##percentage absenteeism hours vs day of the week

```
dayofweek <-  as.data.frame(absent %>% group_by(Day.of.the.week)
%>% summarise(count= n(), percent =
round(count*100/nrow(absent),1))%>% arrange(desc(count)))
ggplot(dayofweek,aes(x = reorder(Day.of.the.week,percent), y=
percent, fill= Day.of.the.week)) + geom_bar(stat = 'identity') +
coord_flip() + theme(legend.position='none') +
  geom_text(aes(label = percent), vjust = 0.5, hjust = 1.1) +
xlab('Day.of.the.week')
```

##percentage absenteeism hours vs social smoker

```
socialsmoker <-  as.data.frame(absent %>%
group_by(Social.smoker) %>% summarise(count= n(), percent =
round(count*100/nrow(absent),1))%>% arrange(desc(count)))
ggplot(socialsmoker,aes(x = Social.smoker, y= percent, fill=
Social.smoker)) + geom_bar(stat = 'identity') + coord_flip() +
theme(legend.position='none') +
  geom_text(aes(label = percent), vjust = 0.5, hjust = 1.1) +
xlab('Social.smoker')
```

##percentage absenteeism hours vs social drinker

```
socialdrinker <-  as.data.frame(absent %>%
group_by(Social.drinker) %>% summarise(count= n(), percent =
round(count*100/nrow(absent),1))%>% arrange(desc(count)))
ggplot(socialdrinker,aes(x = Social.drinker, y= percent, fill=
Social.drinker)) + geom_bar(stat = 'identity') + coord_flip() +
theme(legend.position='none') +
  geom_text(aes(label = percent), vjust = 0.5, hjust = 1.1) +
xlab('Social.drinker')

#percentage of absent people vs education
education=as.data.frame(absent %>% group_by(Education) %>%
summarise(count= n(), percent =
round(count*100/nrow(absent),1))%>% arrange(desc(count)))
ggplot(education,aes(x = Education, y= percent, fill=
Education)) + geom_bar(stat = 'identity') + coord_flip() +
theme(legend.position='none') +
  geom_text(aes(label = percent), vjust = 0.5, hjust = 1.1) +
xlab('Education')

#percentage of absent people vs people having pet
pet=as.data.frame(absent %>% group_by(Pet) %>% summarise(count=
n(), percent = round(count*100/nrow(absent),1))%>%
arrange(desc(count)))
ggplot(pet,aes(x = Pet, y= percent, fill= Pet)) + geom_bar(stat
= 'identity') + coord_flip() + theme(legend.position='none')   +
  geom_text(aes(label = percent), vjust = 0.5, hjust = 1.1) +
xlab('pet')
#percentage of absent people vs age
age=as.data.frame(absent %>% group_by(Age) %>% summarise(count=
n(), percent = round(count*100/nrow(absent),1))%>%
arrange(desc(count)))
ggplot(age,aes(x = Age, y= percent, fill= Age)) + geom_bar(stat
= 'identity') + coord_flip() + theme(legend.position='none') +
  geom_text(aes(label = percent), vjust = 0.5, hjust = 1.1) +
xlab('age')

#percentage of absent people vs weight
```

```
weight=as.data.frame(absent %>% group_by(Weight) %>%
summarise(count= n(), percent =
round(count*100/nrow(absent),1))%>% arrange(desc(count)))
ggplot(weight,aes(x = Weight, y= percent, fill= Weight)) +
geom_bar(stat = 'identity') + coord_flip() +
theme(legend.position='none') +
  geom_text(aes(label = percent), vjust = 0.5, hjust = 1.1) +
xlab('weight')
```

#percentage of absent people vs dist

```
dist=as.data.frame(absent %>%
group_by(Distance.from.Residence.to.Work) %>% summarise(count=
n(), percent = round(count*100/nrow(absent),1))%>%
arrange(desc(count)))
ggplot(dist,aes(x = Distance.from.Residence.to.Work, y= percent,
fill= Distance.from.Residence.to.Work)) + geom_bar(stat =
'identity') + coord_flip() + theme(legend.position='none') +
  geom_text(aes(label = percent), vjust = 0.5, hjust = 1.1) +
xlab('dist from residance')
```

#percentage of absent people vs workload

```
workload=as.data.frame(absent %>%
group_by(Work.load.Average.day) %>% summarise(count= n(),
percent = round(count*100/nrow(absent),1))%>%
arrange(desc(count)))
ggplot(workload,aes(x =Work.load.Average.day, y= percent, fill=
Work.load.Average.day)) + geom_bar(stat = 'identity') +
coord_flip() + theme(legend.position='none') +
  geom_text(aes(label = percent), vjust = 0.5, hjust = 1.1) +
xlab('work load avg')
```

#histogram of numerical features

```
hist(train$Transportation.expense)
hist(train$Distance.from.Residence.to.Work)
hist(train$Service.time)
hist(train$Age)
hist(train$Work.load.Average.day)
```

```r
hist(train$Hit.target)
hist(train$Pet)
hist(train$Weight)
hist(train$Height)
hist(train$Son)
hist(train$Body.mass.index)
hist(train$Absenteeism.time.in.hours)
#boxplots of numerical features
boxplot(train$Transportation.expense)
boxplot(train$Distance.from.Residence.to.Work)
boxplot(train$Service.time)
boxplot(train$Age)
boxplot(train$Work.load.Average.day)
boxplot( train$Hit.target)
boxplot(train$Weight)
boxplot(train$Body.mass.index)
boxplot(train$Absenteeism.time.in.hours)
library(RcmdrMisc)



#anova by month of abscence
AnovaModel.1 <- (lm(Absenteeism.time.in.hours ~ Seasons, data =
train))
Anova(AnovaModel.1)

# Absence Rate By Pet
install.packages('RcmdrMisc')
library(RcmdrMisc)
AnovaModel.2 <- (lm(da$Absenteeism.time.in.hours ~
Reason.for.absence, data = da))
Anova(AnovaModel.2)

# By Season

AnovaModel.3 <- (lm(da$Absenteeism.time.in.hours ~
Month.of.absence, data = da))
Anova(AnovaModel.3)
```

```
# By Social smoker

AnovaModel.4 <- (lm(da$Absenteeism.time.in.hours ~
Social.smoker, data = da))
Anova(AnovaModel.4)

# By Social drinker
AnovaModel.5 <- (lm(da$Absenteeism.time.in.hours ~
Social.drinker, data = da))
Anova(AnovaModel.5)

# By Education

AnovaModel.7 <- (lm(da$Absenteeism.time.in.hours ~ Education,
data = da))
Anova(AnovaModel.7)

# By Disciplanary failure
AnovaModel.8 <- (lm(da$Absenteeism.time.in.hours ~
Disciplinary.failure, data = da))
Anova(AnovaModel.8)

#Min max scaling of numerical data(as they have different
ranges)
#note: not scaling target variable
for (i in c(1:13,15:16,18:20)){
  if (class(train[,i])=="numeric"){
    for (j in c(1:740)){

train[j,i]=(train[j,i]-min(train[i]))/(max(train[i])-min(train[i
]))
    }
  }
}
#replacing outliers
#replacing values greater than 0.95 quantile with values present
in 0.95 quantile
#replacing values less than 0.05 quantile with values present in
0.05 quantile
```

```
qn = quantile(train$Absenteeism.time.in.hours, c(0.05, 0.95),
na.rm = TRUE)
print(qn)
train$Absenteeism.time.in.hours[train$Absenteeism.time.in.hours<
qn[1]]=qn[1]
train$Absenteeism.time.in.hours[train$Absenteeism.time.in.hours>
qn[2]]=qn[2]

#one hot encoding of categorical variables
library(dummies)
hi=data.frame(dummy(train$Reason.for.absence))
train$Reason.for.absence=NULL
hi1=data.frame(dummy(train$Month.of.absence))
train$Month.of.absence=NULL
hi2=data.frame(dummy(train$Day.of.the.week))
train$Day.of.the.week=NULL
hi3=data.frame(dummy(train$Education))
train$Education=NULL
hi4=data.frame(dummy(train$Social.smoker))
train$Social.smoker=NULL
hi5=data.frame(dummy(train$Social.drinker))
train$Social.drinker=NULL
hi6=data.frame(dummy(train$Disciplinary.failure))
train$Disciplinary.failure=NULL
hi7=data.frame(dummy(train$Seasons))
train$Seasons=NULL

#combining binary features with original data

train=cbind(train,hi,hi1,hi2,hi3,hi4)
train=cbind(train,hi5,hi6,hi7)

#correlation plot
library(caret)
library(corrplot)
target=train["Absenteeism.time.in.hours"]
corm=train[-1]
```

```
#coorelation matrix
matrix=cor(corm)
#correlation plot
corrplot(matrix, method="pie")
#removing features with value greater than 0.95
hc = findCorrelation(matrix, cutoff=0.95) #  we can putt any
value as a "cutoff"
#sorting out the columns to be removes
hc = sort(hc)
#removing highly correlated columns
reduced_Data = corm[,-c(hc)]
#combining the clean data with the factor variables of original
data
new_train=cbind(train[factor_cols],reduced_Data)
test=new_train[550:740,-1]
install.packages('DAAG')
library(DAAG)
#feature selection using boruta package
library(Boruta)
new_train$Absenteeism.time.in.hours=log(new_train$Absenteeism.ti
me.in.hours)
finail.boruta=Boruta(Absenteeism.time.in.hours~., data =
new_train[,-1], doTrace = 2)
selected_features=getSelectedAttributes(finail.boruta,
withTentative = F)
set.seed(123)
#creating formula from boruta selected features
formula=as.formula(paste("Absenteeism.time.in.hours~",paste(sele
cted_features,collapse = "+")))


#predictive models

# linear model
train_control <- trainControl(method = "repeatedcv",
                        number = 10)
options(warn=-1)
lm_model<- train(formula,data=new_train[1:500,-1],
      metric="RMSE", method="lm",trControl=train_control)
```

```
pred=predict(lm_model,test)
# estimate variable importance
lm_importance <- varImp(lm_model, scale=FALSE)
# summarize importance
print(lm_model)
# plot importance
plot(lm_importance)



#random forest model
rf_model<- train(formula,data=new_train[1:550,-1],
              metric="RMSE",
method="rf",trControl=train_control)

pred=predict(ridge_model,test)


# summarize model
print(rf_model)
# plot model
plot(rf_model)


#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)


# summarize model
print(dt_model)
# plot importance
plot(dt_model)
```

```
###################
#
#
##removing certain features from the  model as heir p_value is
very high not statistically significant
#
#
#
#
#
###################

new=c('Reason.for.absence.1','Reason.for.absence.9',

'Reason.for.absence.13','Reason.for.absence.19','Reason.for.abse
nce.23',

'Reason.for.absence.25','Reason.for.absence.27','Reason.for.abse
nce.28','Work.load.Average.day' ,'Disciplinary.failure.1' )
#creating formula from new features
new_formula=as.formula(paste("Absenteeism.time.in.hours~",paste(
new,collapse = "+")))
# linear model
train_control <- trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 6)
new_lm_model<- train(new_formula,data=new_train,
              metric="RMSE",
method="lm",trControl=train_control)
print(new_lm_model)
print(summary(new_lm_model))
pred=predict(new_lm_model,test)
# estimate variable importance
new_lm_importance <- varImp(new_lm_model, scale=FALSE)
# summarize importance
print(new_lm_model)
#summary of the model
print(summary(new_lm_model))
```

```
# plot importance
plot(new_lm_importance)


#ridge model
new_ridge_model<- train(new_formula,data=new_train,
              metric="RMSE",
method="ridge",trControl=train_control)
#print model
print(new_ridge_model)
#summary of the model
print(summary(new_ridge_model))
new_ridge_pred=predict(new_ridge_model,test)
# estimate variable importance
new_ridge_importance <- varImp(new_ridge_model, scale=FALSE)
# summarize model
print(new_ridge_importance)
# plot importance
plot(new_ridge_importance)


#random forest model
new_rf_model<- train(new_formula,data=new_train[1:550,-1],
                 metric="RMSE",
method="rf",trControl=train_control)

pred=predict(new_rf_model,test)


# summarize model
print(new_rf_model)
# plot model
plot(new_rf_model)


#decision tree model
new_dt_model<- train(new_formula,data=new_train[1:550,-1],
                 metric="RMSE",
method="rpart",trControl=train_control)
```

```
pred=predict(new_dt_model,test)


# summarize importance
print(new_dt_model)
# plot importance
plot(new_dt_model)


############################
# from the above models even after removing statistically
#insignificant variables
#the RMSE and R-squared value is very low
############################







####################
#
#
#Principal component analysis
#
#
####################



#divide the new data
 pca.train <- train[1:550,-1]
 pca.test <- train[551:740,-1]
 #principal component analysis
 prin_comp <- prcomp(pca.train)
 #outputs the mean of variables
 prin_comp$center
 #outputs the standard deviation of variables
 prin_comp$scale
 dim(prin_comp$x)
 biplot(prin_comp, scale = 0)

 #compute standard deviation of each principal component
 std_dev <- prin_comp$sdev

 #compute variance
  pr_var <- std_dev^2
 #proportion of variance explained
 prop_varex <- pr_var/sum(pr_var)
 #scree plot
 plot(prop_varex, xlab = "Principal Component",
        ylab = "Proportion of Variance Explained",
        type = "b")

 #cumulative scree plot
 plot(cumsum(prop_varex), xlab = "Principal Component",
        ylab = "Cumulative Proportion of Variance Explained",
        type = "b")
 #add a training set with principal components
  train.data <- data.frame(Absenteeism.time.in.hours =
pca.train$Absenteeism.time.in.hours, prin_comp$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(prin_comp, newdata = pca.test)
 test.data= as.data.frame(test.data)

 #select the first 30 components
 test.data=test.data[,1:40]
 #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

pca.lm.trend=data.frame(da[551:740,-1]$Month.of.absence,pca.lm.p
rediction)
 ggplot(pca.lm.trend, aes(x = da.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)
)




 #Ridge regression
 set.seed(123)
 train_control = trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 6)
 pca_ridge_model=
train(Absenteeism.time.in.hours~.,data=train.data,
                      metric="RMSE",
method="ridge",trControl=train_control)
```

```
 print(pca_ridge_model)
 print(summary(pca_ridge_model))
 #make prediction on test data
 pca.ridge.prediction = predict(pca_ridge_model, test.data)
 # absent rate in every month 2011 if same trend of absenteeism
continues


pca.ridge.trend=data.frame(da[551:740,-1]$Month.of.absence,pca.r
idge.prediction)
 ggplot(pca.ridge.trend, aes(x =
da.551.740...1..Month.of.absence , y = pca.ridge.prediction) ) +
geom_bar(stat = 'identity')

 #finding RMSE on test data

print(RMSE(pca.ridge.prediction,pca.test$Absenteeism.time.in.hou
rs))


 #Random forest
 set.seed(123)
 train_control = trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 6)
 pca_rf_model=train(Absenteeism.time.in.hours~.,data=train.data,
                        metric="RMSE",
method="rf",trControl=train_control)
 print(pca_rf_model)
 print(summary(pca_rf_model))

 #make prediction on test data
 pca.rf.prediction = predict(pca_rf_model, test.data)
  # absent rate in every month 2011 if same trend of absenteeism
continues


pca.rf.trend=data.frame(da[551:740,-1]$Month.of.absence,pca.rf.p
rediction)
```

```
 ggplot(pca.rf.trend, aes(x = da.551.740...1..Month.of.absence ,
y = pca.rf.prediction) ) + geom_bar(stat = 'identity')

 #finding RMSE on test data

print(RMSE(pca.rf.prediction,pca.test$Absenteeism.time.in.hours)
)


 #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(da[551:740,-1]$Month.of.absence,pca.dt.p
rediction)
 ggplot(pca.dt.trend, aes(x = da.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)
)


 #lasso regression
 set.seed(123)
 train_control = trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 6)

pca_lasso_model=train(Absenteeism.time.in.hours~.,data=train.dat
a,
                    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(da[551:740,-1]$Month.of.absence,pca.l
asso.prediction)
 ggplot(pca.lasso.trend, aes(x =
da.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.hou
rs))
```

Python code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Jun  7 00:02:37 2018

@author: vikramreddy
"""

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
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


df = pd.read_excel("Absenteeism_at_work_Project.xls")
df.info()
df.describe()
#missing value imputation


columns={'Disciplinary failure','Reason for absence','Month of
absence','Day of the week',
         'Seasons','Transportation expense','Distance from
Residence to Work','Service time',
         'Age','Hit target','Education','Son','Social
drinker','Work load Average/day ',
         'Social smoker','Pet','Weight','Height','Body mass
index','Absenteeism time in hours'}
for column in columns:
    dat[column].fillna(dat[column].median(),inplace=True)

dat=df.copy()

#recoding categorical variables
rule={0:'infectious,parasitic
diseases',1:'Neoplasms',2:'Diseases of the blood',3:'Endocrine
and metabolic diseases',4:'Mental and behavioural disorders',
      5:'Diseases of the nervous system',6:'Diseases of the eye
and adnexa',7:'Diseases of the ear and mastoid process',
      8:'Diseases of the circulatory system',9:'Diseases of the
respiratory system',10:'Diseases of the digestive system',
      11:'Diseases of the skin and subcutaneous
tissue',12:'Diseases of the musculoskeletal system and
connective tissue',
      13:'Diseases of the genitourinary system',14:'Pregnancy,
childbirth and the puerperium',15:'Certain conditions
originating in the perinatal',
      16:'Congenital malformations, deformations and chromosomal
abnormalities',17:'Symptoms, signs and abnormal clinical
findings',
      18:'Injury, poisoning and certain other consequences of
external causes',19:'causes of morbidity and mortality',
      21:'Factors influencing health status and contact with
health services',22:'patient follow-up',23:'medical
consultation',24:'blood donation',
      25:'laboratory examination',26:'unjustified
absence',27:'physiotherapy',28:'dental consultation'}
rule2={0:'None',1:'Jan',2:'Feb',3:'Mar',4:'Apr',5:'May',

6:'Jun',7:'Jul',8:'Aug',9:'Sep',10:'Oct',11:'Nov',12:'Dec'}
rule3={1:'summer',2:'autumn',3:'winter',4:'spring'}
rule4={1:'highschool',2:'graduate',3:'postgraduate',4:'master&
doctrate'}
rule5={0:'No',1:'Yes'}
rule6={2:"Monday",3:"Tuesday",4:"Wednesday",5:"Thursday",6:"Frid
ay"}
df["Reason for absence"]=df["Reason for absence"].replace(rule)
df["Month of absence"]=df["Month of absence"].replace(rule2)
df['Day of the week']=df['Day of the week'].replace(rule6)
df['Education']=df['Education'].replace(rule4)
df["Seasons"]=df["Seasons"].replace(rule3)
df["Social drinker"]=df["Social drinker"].replace(rule5)
df["Social smoker"]=df["Social smoker"].replace(rule5)
df["Disciplinary failure"]=df["Disciplinary
failure"].replace(rule5)
df.isnull().any().any()
```

```
dat.isnull().any().any()
dat.info()




#heat map of numerical features
#no features are highly correlated
colormap = plt.cm.RdBu
plt.figure(figsize=(15,15))
plt.title('Pearson Correlation of Features', y=1.0, size=10)
sns.heatmap(dat[['Transportation expense','Distance from
Residence to Work','Service time','Work load Average/day ',
         'Age','Hit target','Pet','Weight','Height','Body mass
index','Absenteeism time in
hours']].corr(),linewidths=0.2,vmax=1.0,
            square=True, cmap=colormap, linecolor='white',
annot=True)


#outliers plot
#there are outliers in target variable


plt.boxplot(dat['Transportation expense'])
plt.boxplot(dat['Distance from Residence to Work'])
plt.boxplot(dat['Service time'])
plt.boxplot(dat['Age'])
plt.boxplot(dat['Pet'])
plt.boxplot(dat['Weight'])
plt.boxplot(dat['Height'])
plt.boxplot(dat['Body mass index'])
plt.boxplot(dat['Absenteeism time in hours'])#ouliers spotted
#exploratory data analysis of factor variables  with
visualisation
```

```
#boxplots of factor variables vs target variable
sns.boxplot(x='Seasons', y='Absenteeism time in hours', data=df)
sns.boxplot(x='Reason for absence', y='Absenteeism time in
hours', data=df)
sns.boxplot(x='Education', y='Absenteeism time in hours',
data=df)
sns.boxplot(x='Social smoker', y='Absenteeism time in hours',
data=df)
sns.boxplot(x='Social drinker', y='Absenteeism time in hours',
data=df)
sns.boxplot(x='Disciplinary failure', y='Absenteeism time in
hours', data=df)
#histogram of factor variables
sns.countplot(df['Reason for absence'], color='blue')
sns.countplot(df['Month of absence'], color='blue')
sns.countplot(df['Social smoker'], color='blue')
sns.countplot(df['Education'], color='blue')
sns.countplot(df['Day of the week'], color='blue')
sns.countplot(df['Social drinker'], color='blue')
sns.countplot(df['Disciplinary failure'], color='blue')




#intuition on the interactions between continuous variables
sns.pairplot(dat, vars=['Transportation expense','Distance from
Residence to Work','Service time',
         'Age','Hit target','Pet','Weight','Height','Body mass
index','Absenteeism time in hours'],
                 kind='reg')
##intuition on the interactions between categorical variables
sns.pairplot(dat, vars=['Seasons','Reason for
absence','Education','Social smoker','Social
drinker','Disciplinary failure'],
                 kind='reg', hue='SEX')


#anova test
```

```
#as i am getting error dues to names i am changing the column
names
hi=dat.copy()
hi.columns=hi.columns.str.replace(' ','_')
list(hi)
hi.isnull().sum()
np.warnings.filterwarnings('ignore')
# effect of factor variables with target variable
model_1=ols('Absenteeism_time_in_hours ~
C(Education)',data=hi).fit()
sm.stats.anova_lm(model_1, typ=1)#no significant difference in
means
model_2=ols('Absenteeism_time_in_hours ~
C(Reason_for_absence)',data=hi).fit()
sm.stats.anova_lm(model_2, typ=1)#significant difference in
means
model_3=ols('Absenteeism_time_in_hours ~
C(Day_of_the_week)',data=hi).fit()
sm.stats.anova_lm(model_3, typ=1)#no significant difference in
means
model_4=ols('Absenteeism_time_in_hours ~
C(Seasons)',data=hi).fit()
sm.stats.anova_lm(model_4, typ=1)#no significant difference in
means
model_5=ols('Absenteeism_time_in_hours ~
C(Disciplinary_failure)',data=hi).fit()
sm.stats.anova_lm(model_5, typ=1)#no significant difference in
means
model_6=ols('Absenteeism_time_in_hours ~
C(Social_drinker)',data=hi).fit()
sm.stats.anova_lm(model_6, typ=1)##no significant difference in
means
model_7=ols('Absenteeism_time_in_hours ~
C(Social_smoker)',data=hi).fit()
sm.stats.anova_lm(model_7, typ=1)


#scaling numerical features
```

```
numeric_features={'Transportation expense','Distance from
Residence to Work','Service time',
         'Age','Hit target','Pet','Weight','Height','Body mass
index','Work load Average/day '}
scaler= MinMaxScaler()
for column in numeric_features:

dat[column]=scaler.fit_transform(dat[column].values.reshape(-1,1
))

#one_hot_encoding for factor variables

reason=pd.get_dummies(dat['Reason for absence'])
month=pd.get_dummies(dat['Month of absence'])
education=pd.get_dummies(dat['Education'])
season=pd.get_dummies(dat['Seasons'])
day=pd.get_dummies(dat['Day of the week'])
discip=pd.get_dummies(dat['Disciplinary failure'])
soc_smok=pd.get_dummies(dat['Social smoker'])
soc_drink=pd.get_dummies(dat['Social drinker'])

#combining all the binary coded variables to original d data
encod=pd.concat([reason,month,education,season,day,discip,soc_sm
ok,soc_drink],axis=1)




#removing all the factor variables which are used in ine hot
encoding
dat=dat.drop(['Reason for absence','Month of
absence','Education','Seasons',
             'Day of the week','Disciplinary failure','Social
smoker','Social drinker'],axis=1)


#   joining encoded data to the original data
dat = dat.join(encod)
```

```
#######     creating copy of the processed data to use in PCA
model    ####
pca_data=dat.copy()
```

```
###############################################################
#
#
#
######################      Feature engineering
###################
#
#
################################################################
#
```

```
target=dat['Absenteeism time in hours']
train=dat.drop(['Absenteeism time in hours'],axis=1)
```

```
#selecting top 15 features
X_new = SelectKBest(f_regression,
k=15).fit_transform(train,target)
X_train, X_test, y_train, y_test =
train_test_split(X_new,target, test_size=0.2,random_state=42)
```

```
##########################
#
#
#  Removing outliers from the data
#
#
###########################



 #replacing outliers less than first quantile value with 0.05
quantile value
 #replacing outliers greater than 0.95 quantile value with 0.95
quantile value


data_without_outliers=dat.copy()
down=dat['Absenteeism time in hours'].quantile(0.05)  ###value=1

up=dat['Absenteeism time in hours'].quantile(0.95)    ##value
=24

data_without_outliers['Absenteeism time in
hours']=data_without_outliers['Absenteeism time in
hours'].mask(data_without_outliers['Absenteeism time in hours']
< 1,1)
data_without_outliers['Absenteeism time in
hours']=data_without_outliers['Absenteeism time in
hours'].mask(data_without_outliers['Absenteeism time in hours']
>24,24)

target_data=data_without_outliers['Absenteeism time in hours']
train_data=data_without_outliers.drop(['Absenteeism time in
hours'],axis=1)


 #selecting top 15 features
```

```
X_new_clean = SelectKBest(f_regression,
k=15).fit_transform(train_data,target_data)

X_train_clean, X_test_clean, y_train_clean, y_test_clean =
train_test_split(X_new_clean,target_data,
test_size=0.2,random_state=42)

##################################
#
#
# creating a function that displays the results of the model
#
#
##################################



def model_results(model):


    ####################################
    #
    #
    #raw data --------   *****   with outliers    ******
    #
    #
    ####################################

    #fitting the raw data(with outliers tothe model)
    model.fit(X_train,y_train)

    #test predictions
    test_predictions=model.predict(X_test)

    RMSE_raw=np.sqrt(mean_squared_error(y_test,
test_predictions))

    print("test-RMSE of data with outliers")
    print(RMSE_raw)
    print('coefficient of determination R^2 of the prediction')

    print(model.score(X_test, y_test))


    #####################################
    #
    #
    #clean data -------    *******    with out outliers  ******
    #
    #
    #####################################


   #fitting outlier removed data to the model
    model.fit(X_train_clean,y_train_clean)

    #test predictions
    test_pred_clean=model.predict(X_test_clean)


   #rescaling  predictions on test data as we have scaled in the
beginning
    RMSE_clean=np.sqrt(mean_squared_error(y_test_clean,
test_pred_clean))

    print("test-RMSE of outlier removed data(clean data) ")
    print(RMSE_clean)

    # Returns the coefficient of determination R^2 of the
prediction.
    print('coefficient of determination R^2 of the prediction')
    print(model.score(X_test_clean, y_test_clean))

    return ""
```

```
############################################
#
#
#
########   Principal component analysis     ##########
#
#
#
############################################



#data preparations

pca_data['Absenteeism time in hours']=pca_data['Absenteeism time
in hours'].mask(pca_data['Absenteeism time in hours'] < 1,1)
pca_data['Absenteeism time in hours']=pca_data['Absenteeism time
in hours'].mask(pca_data['Absenteeism time in hours'] >24,24)
#convert it to numpy arrays
#dropping employee ID column
data=pca_data.drop(['ID'],axis=1)
X=data.values

#target variable
target_pca=data['Absenteeism time in hours']

#passing the total number of components to the PCA
pca = PCA(n_components=72)

#fitting the values to PCA
pca.fit(X)

#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)


#############################
## 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=40)

#now fitting the selected components to the data
pca.fit(X)

#PCA selected features
X1=pca.fit_transform(X)
#splitting train and test data
X_train_pca, X_test_pca, y_train_pca, y_test_pca =
train_test_split(X1,target_pca, test_size=0.2,random_state=42)
```

```
################################
#
#
#creating a function that displays PCA results of their models
#
#
################################

def pca_model_results(model):



    #fitting training data to the model
    model.fit(X_train_pca,y_train_pca)
    #train predictions
    train_pred_pca=model.predict(X_train_pca)

    #RMSE of Train predictions and train data
    train_RMSE=np.sqrt(mean_squared_error(y_train_pca,
train_pred_pca))
    print("RMSE on trained data of PCA model")
    print(train_RMSE)



    #test predictions
    test_pred_pca=model.predict(X_test_pca)

    #RMSE of test predictions and test data
    RMSE=np.sqrt(mean_squared_error(y_test_pca, test_pred_pca))
    print("test-RMSE PCA model ")
    print(RMSE)


    # Returns the coefficient of determination R^2 of the
prediction.
    print('coefficient of determination R^2 of the prediction')
    print(model.score(X_test_pca, y_test_pca))
    return ""
```

```
#######################################
#
#
#  Building predictive models and calling the results function
that i have created above
#
#
######################################



#ridge regression
Ridge_model=Ridge()#***
model_results(Ridge_model)
pca_model_results(Ridge_model)

#random forest
rf_model=RandomForestRegressor(random_state=2)
model_results(rf_model)
pca_model_results(rf_model)


#lasso regression
lasso_model=Lasso()
model_results(lasso_model)
pca_model_results(lasso_model)

#Decision tree regressor
DT_model=DecisionTreeRegressor(random_state=3)
model_results(DT_model)
pca_model_results(DT_model)



#linear regression
lin_reg_model=LinearRegression()#****
model_results(lin_reg_model)
pca_model_results(lin_reg_model)
```

```
#lars regressor
lars_model=Lars()#****
model_results(lars_model)
pca_model_results(lars_model)
```

References

1.Towards data science
2.Machine learning by bretty lantz
3.Augmented startup videos
4.Analytics vidhya