Predicting Absenteeism

Abhishek Mandal

31/7/2019

Table of Contents

Chapter 1

Introduction

1.1. Problem statement

Absenteeism is an issue faced by the courier industry in particular and logistics industry in general. The client here faces genuine issue of absenteeism. As a consumer facing industry, where workforce is the main resource, absenteeism can seriously effect a company's wellbeing, financial or otherwise.

Here the client has given us a dataset comprising of the employees details. The client wants us to provide insights to the data so as to identify the causes of absenteeism and take corrective measures so as to minimize the workhours lost.

Also the client wants us to predict the future trends of absenteeism so that available resources can be better optimized and planning for the future can be done.

1.2. The Data

The task given has two main objectives:

To gain insights into the given data and identify the factors which is leading to absenteeism.

To create a model which will predict the future absenteeism.

The data for 1 year is given and our task is to predict the absenteeism hours for the next year. It is a seasonal data.

Table 1.1: Absenteeism data (column 1-7)

ID	Reason for absence	Month of absence	Day of the week	Seasons	Transportation expense	Distance from Residence to Work
11	26	7	3	1	289	36
36	0	7	3	1	118	13
3	23	7	4	1	179	51
7	7	7	5	1	279	5
11	23	7	5	1	289	36
3	23	7	6	1	179	51
10	22	7	6	1		52
20	23	7	6	1	260	50
14	19	7	2	1	155	12

Table 1.2: Absenteeism data (column 8-14)

Service time	Age	Work load Average/day	Hit target	Disciplinary failure	Education	Son
13	33	239,554	97	0	1	2
18	50	239,554	97	1	1	1
18	38	239,554	97	0	1	0
14	39	239,554	97	0	1	2
13	33	239,554	97	0	1	2
18	38	239,554	97	0	1	0
3	28	239,554	97	0	1	1
11	36	239,554	97	0	1	4
14	34	239,554	97	0	1	2

Table 1.3: Absenteeism data (column 15-21)

Social drinker	Social smoker	Pet	Weight	Height	Body mass index	Absenteeism time in hours
1	0	1	90	172	30	4
1	0	0	98	178	31	0
1	0	0	89	170	31	2
1	1	0	68	168	24	4
1	0	1	90	172	30	2
1	0	0	89	170	31	
1	0	4	80	172	27	8
1	0	0	65	168	23	4
1	0	0	95	196	25	40

Chapter 2

Methodology

2.1. Pre Processing

Preprocessing is a very important part of data analysis. Raw or unprocessed data can be very messy, contain outliers and missing values, N/A values, can be skewed etc., which can severely affect the model and hence predictions. Also different machine learning algorithms require data in particular formats. Like Naïve Bayes take categorical data as input and Neural Networks numerical data. So the data has to be modified and transformed accordingly.

2.1.1. Missing Value Analysis

The raw data provided by the client can have several missing values which cannot be processed by many algorithms like Logistic Regression etc. So we have to analyze and fill those gaps.

Table 2.1: Missing data

Column	Missing Values
Body mass index	31
Absenteeism time in hours	22
Height	14
Education	10
Work load Average/day	10
Transportation expense	7
Disciplinary failure	6
Hit target	6
Son	6
Social smoker	4
Social drinker	3
Age	3
Service time	3
Distance from Residence to Work	3
Reason for absence	3
Pet	2
Weight	1
Month of absence	1
Seasons	0
Day of the week	0
ID	0

There are a few techniques, which we can use for replacing missing values like replacing with 0, mean, mode etc. Here we have used mode (most frequently occurring term) to replace missing values.

2.1.2. Outlier Analysis

An Outlier is a data point which is very far from the mean. Such a data can affect the distribution and model. It may come from incorrect data reporting or may be an exception. So it is important to remove these data. However some algorithms are not affected by outliers. Also in small datasets removing outliers can affect the data negatively. In this case, due to small size of dataset and its cyclic nature means removing outliers affected the overall dataset negatively. Also the regression algorithms used for predictions are made robust to outliers. Hence we are not removing outliers for this particular dataset.

2.1.3. Feature Selection

Before performing different models we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of regression. Feature selection become more important in this data set, because it have features of time series as well as regression. So we analyzed all the data again with various plots, we are plotting the correlation plot below.
We have 20 numeric variables but 5 of them are with levels.

- Disciplinary failure
- Education
- Son
- Social drinker
- Social smoker
- Pet

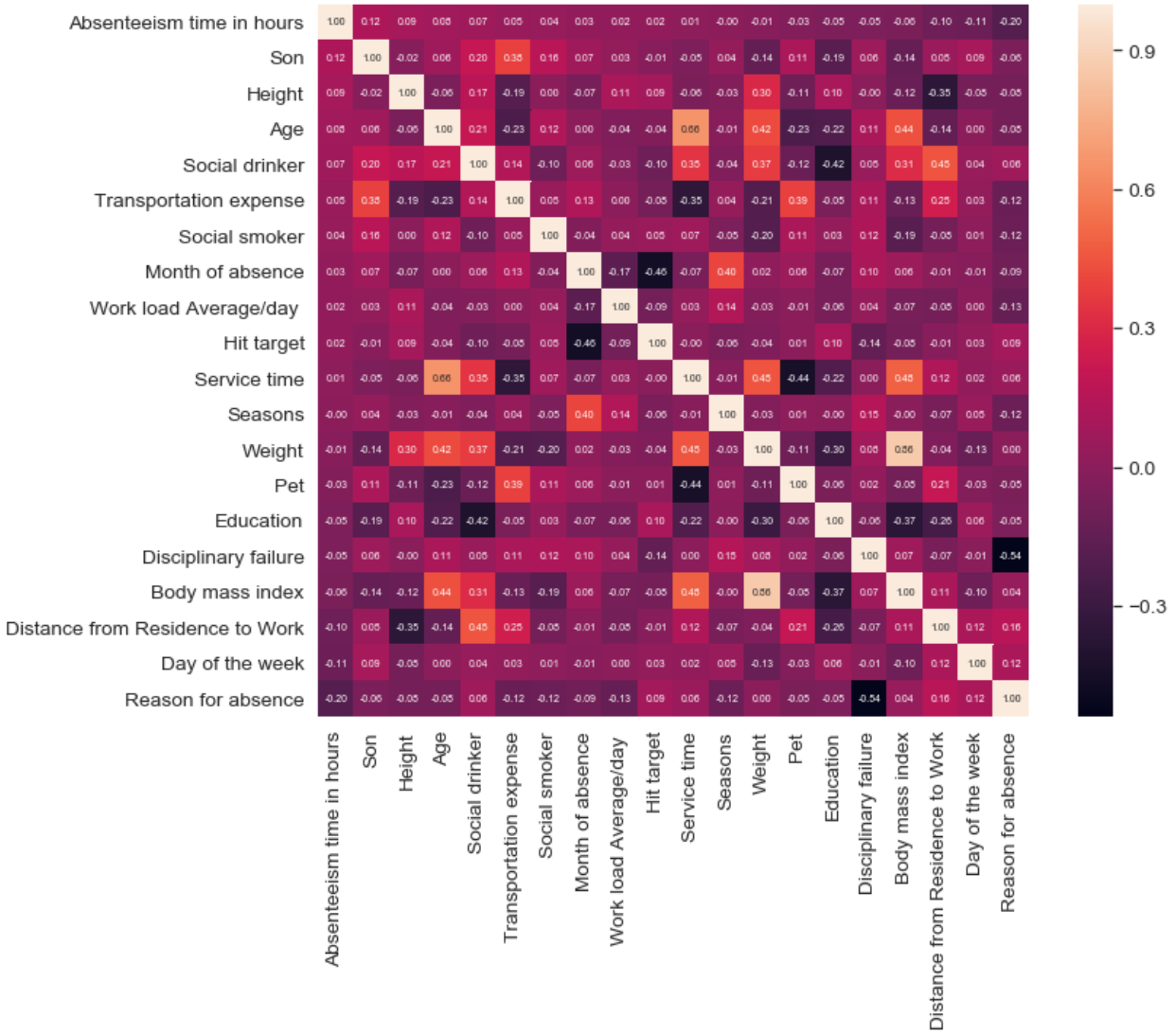


Figure 2.1 Correlation Matrix

The above plot is a correlation plot which gives us an idea how much the variables are correlated to each other. So we will take a look at the more correlated variables and get the correlation factor.

Table 2.2 Correlation coefficients

Variables	Correlation coefficient
Height & Body mass index	-0.1202946
Weight & Body mass index	0.86293245
Age & Service time	0.66102651

As we can see from the above table, Weight and Body mass index and Age and Service time are highly correlated so they pose a problem can ***multicollinearity***. We have to remove one of the variables. So we remove Weight and Service time.

2.1.4 Exploratory data Analysis

In exploratory data analysis, we get insights into the data. In this case, we will use visual methods to establish a relation between the target variable i.e. 'Absenteeism time in hours' and the predictor variables. We will use bar charts.

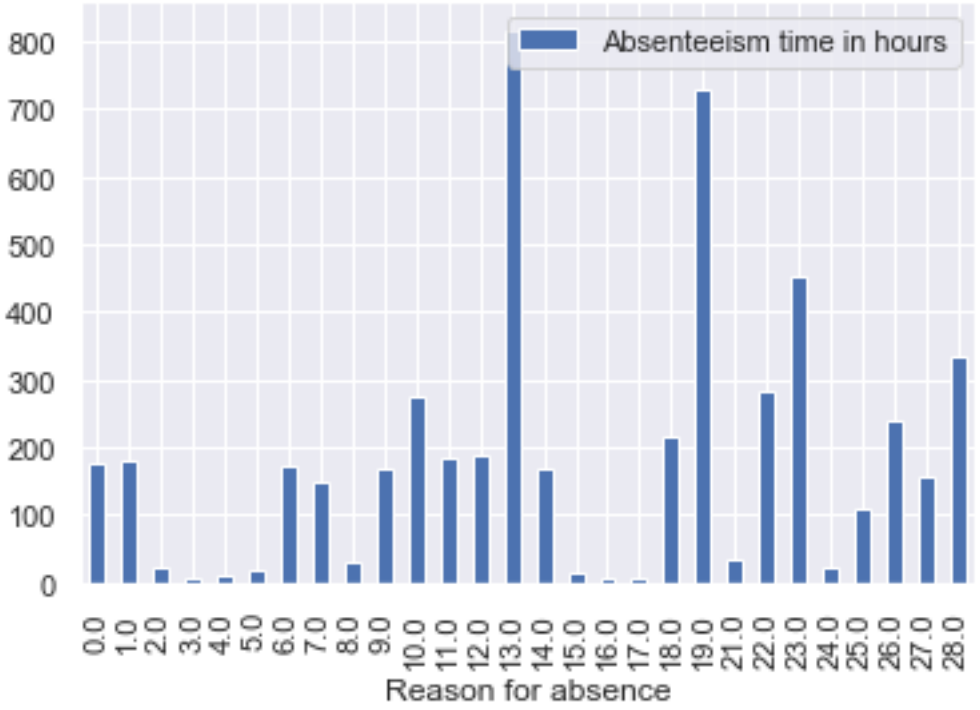


Figure 2.2: Absenteeism time in hours vs Reason of absence

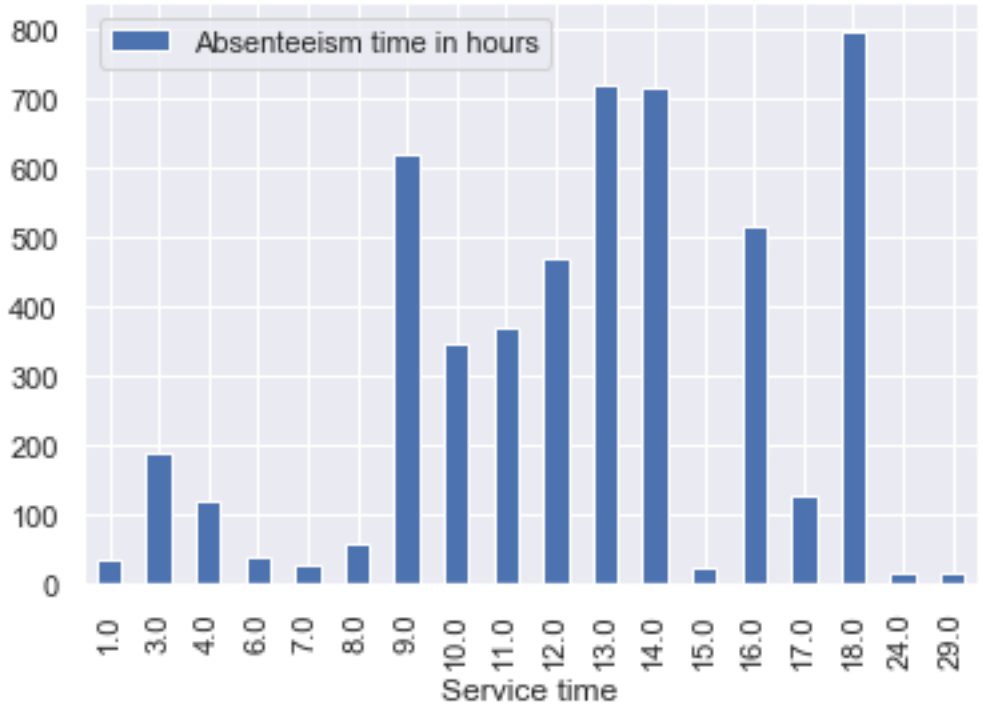


Fig 2.3: Absenteeism time in hours vs Service time

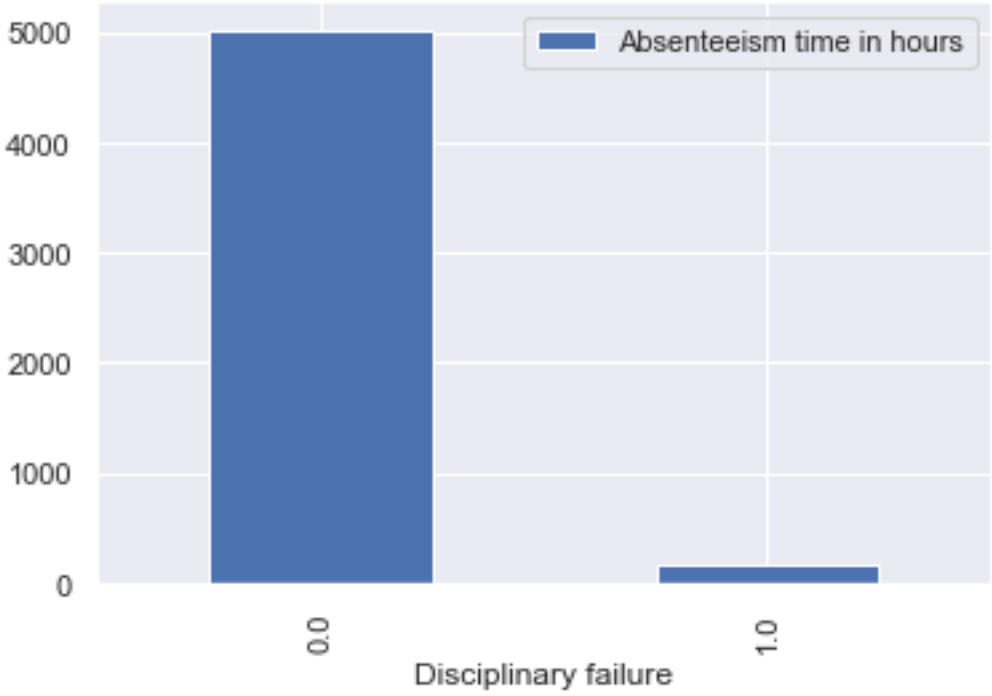


Fig 2.4: Absenteeism time in hours vs Disciplinary failure

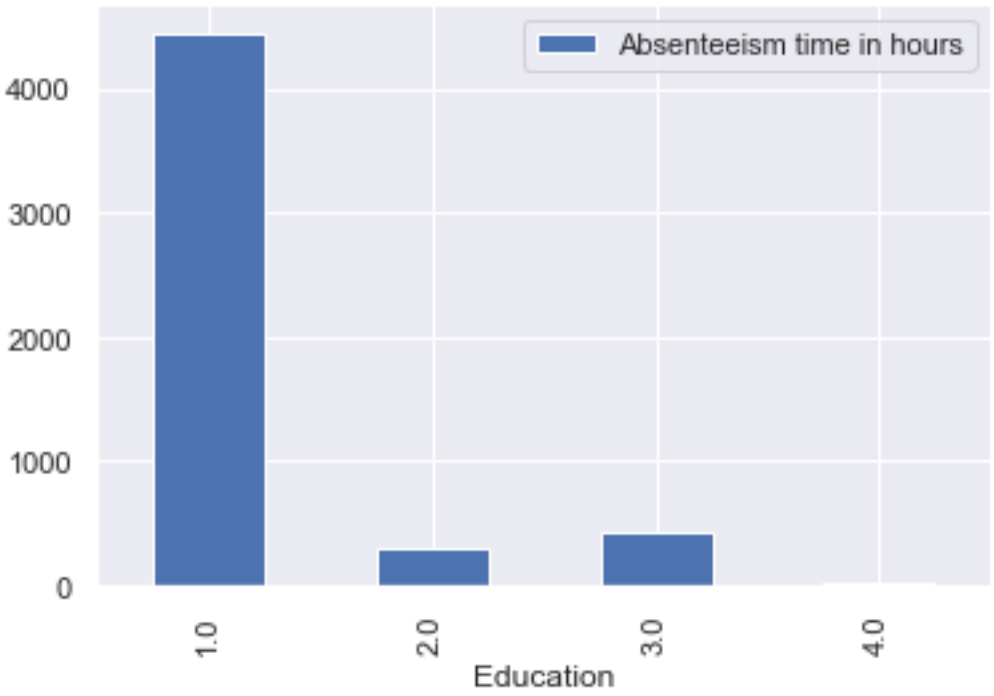


Fig 2.6: Absenteeism time in hours vs Education

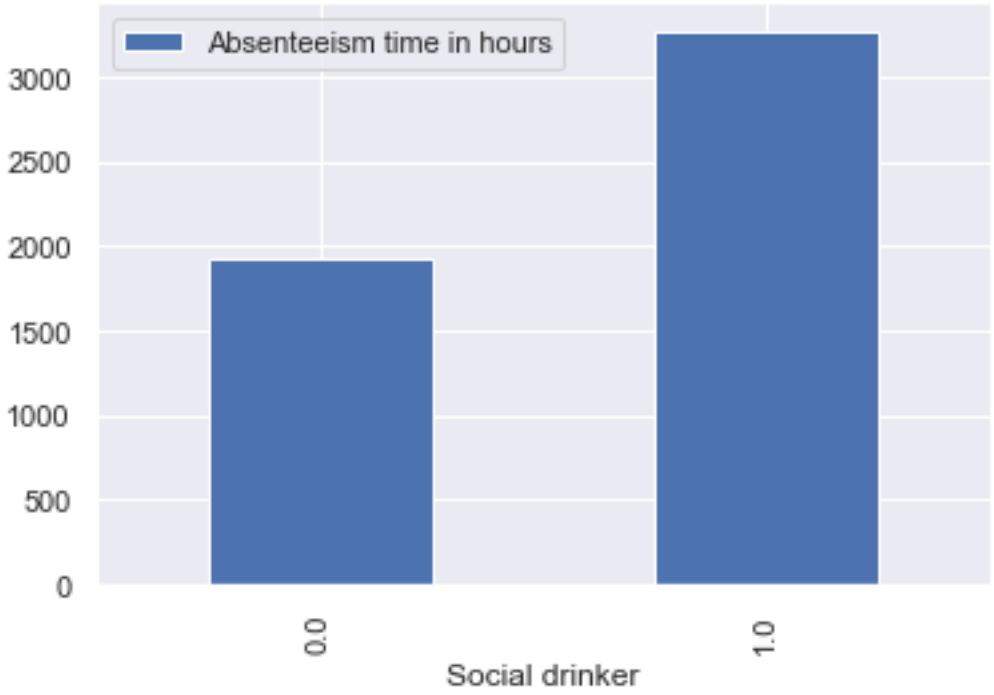


Fig 2.7: Absenteeism time in hours vs Social drinker

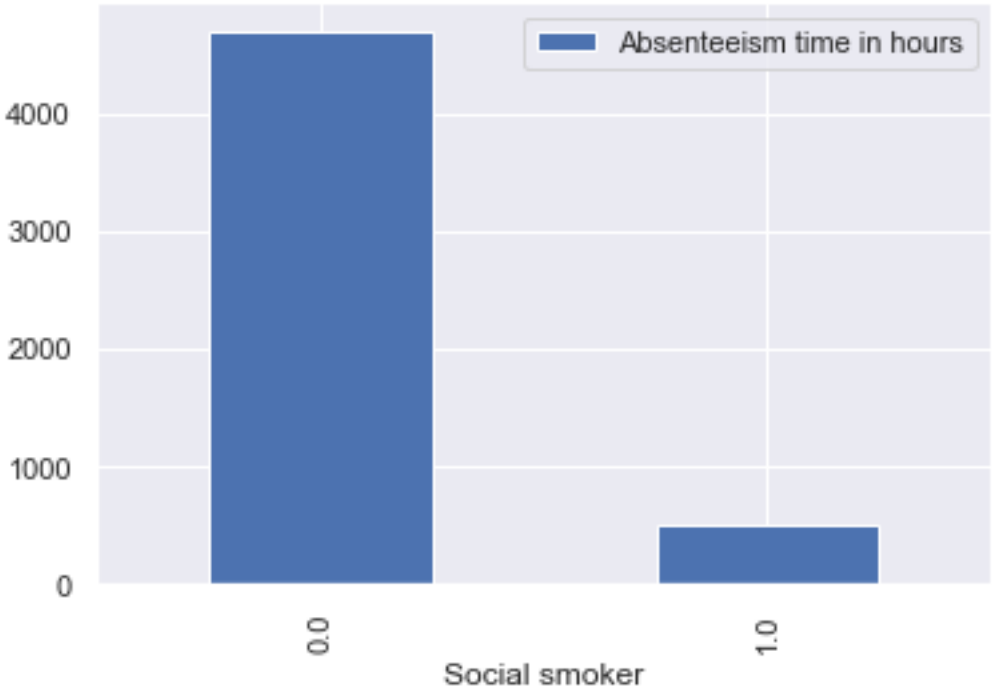


Fig 2.8: Absenteeism time in hours vs Social smoker

2.1.4.1 Insights from diagrams

- From fig2.2 certain diseases like Diseases of the musculoskeletal system and connective tissue and Injury, poisoning and certain other consequences of external causes affect absenteeism time more than others. Injury is one of the factors and can be due to workplace related and/or related to the nature of the industry.
- From 2.4, we can see that employees who had been reprimanded for disciplinary cases tend to be less absent than those who haven't by a huge margin many times over.
- From fig 2.6 it can be inferred that those with only a high school degree tend to be far more absent than those who had further education with not much difference in the categories after that.
- From fig 2.7 we can see that drinkers tend to be absent more.
- Other figures do not show anything conclusive

2.2. Modelling

2.2.1. Model Selection

From preprocessing, we can see that the data is seasonal and there is cyclic trend. So we will approach the problem two ways. First we will use Regression to predict continuous numeric integer values and second will be time series model.

2.2.2. Regression

For regression, we have used 5 models:

- XGBoost
- LightGBM
- LASSO Regression
- Elastic Net Regression
- Ridge Regression

Created a blended model of all the models to get the final data

2.2.3. Time series model

Univariate analysis is the simplest form of analyzing data. “Uni” means “one”, so in other words your data has only one variable. It doesn't deal with causes or relationships (unlike regression) and its major purpose is to describe; it takes data, summarizes that data and finds patterns in the data. We used this method to see the other perspective of data set, before using this we have change the data to make data sets to put in the model, we changed data into two variables dates and data, and dates are in range of years. And after the analysis we got good results in respect of prediction. We also tried different order of model to get the best result. Actually ARIMA model is having 3 inputs for creating model. A seasonal ARIMA model is classified as an ARIMA (p,d,q)x(P,D,Q) model, where P=number of seasonal autoregressive (SAR) terms, D=number of seasonal differences, Q=number of seasonal moving average (SMA)

Chapter 3

Conclusion

3.1. Model Evaluation

Now we have 2 models: Blended Regression Model and Time Series Model. We will use 2 parameters for deciding the model.

- Mean Absolute Percentage Error(MAPE)
- R squared value

The mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting. It usually expresses accuracy as a percentage, and is defined by the formula:

$$\mathrm{M} = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|,$$

where At is the actual value and Ft is the forecast value. The difference between At and Ft is divided by the actual value At again. The absolute value in this calculation is summed for every forecasted point in time and divided by the number of fitted points n. multiplying by 100% makes it a percentage error.

The R squared value gives us the distance at which the actual points lie from the regression line.

Model	**MAPE**	**R-Squared value**
Blended Regression	75.88%	0.326
Time Series ARIMA	23.28%	0.977

Table 3.1: Model Evaluation Summary

The Code for the models and evaluation is given in Appendix

3.2. Model Selection

We can see that the time series model gives better result.

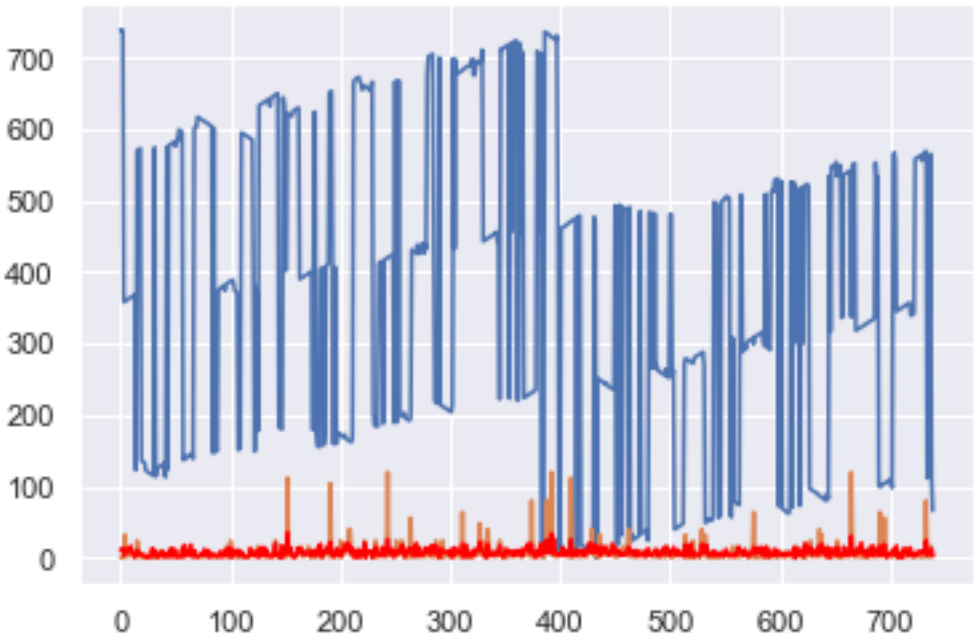


Fig 3.1: Actual vs Predicted-Regression

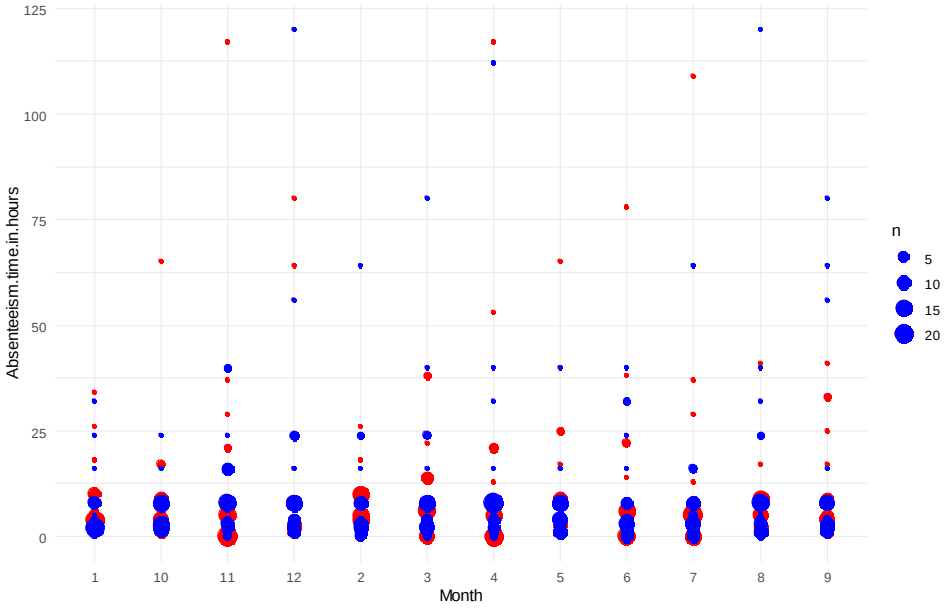


Fig 3.2: Actual vs Predicted- Time Series-Count Plot

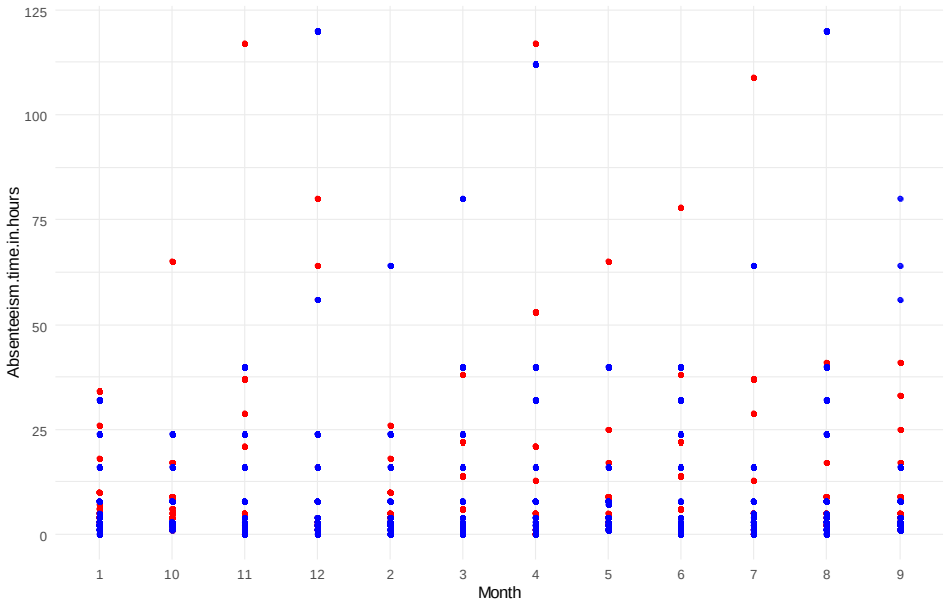


Fig 3.3: Actual vs Predicted-Time Series – Point Plot

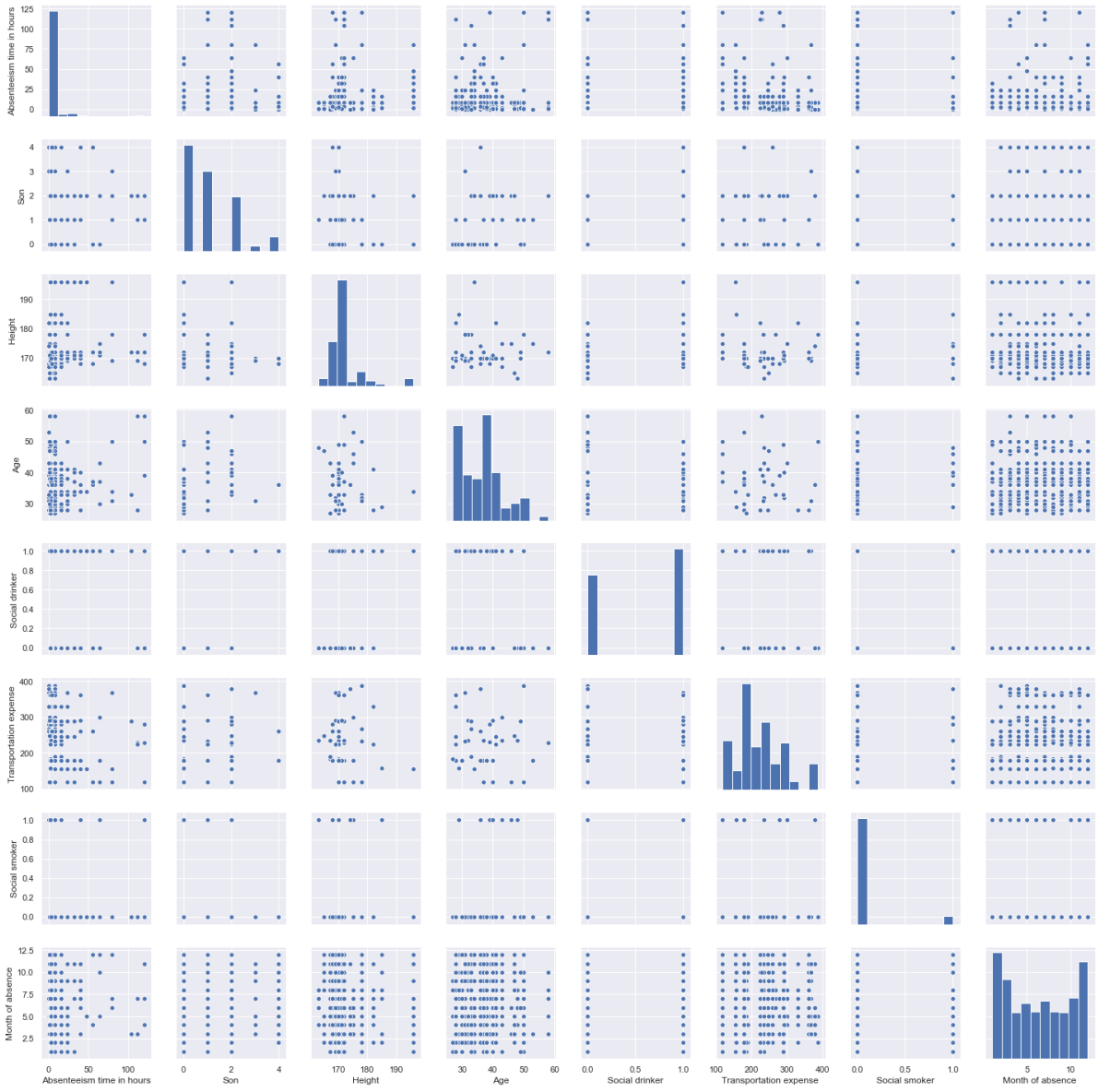
We can see from the graphs that time series model gives the best predictions.

Solutions

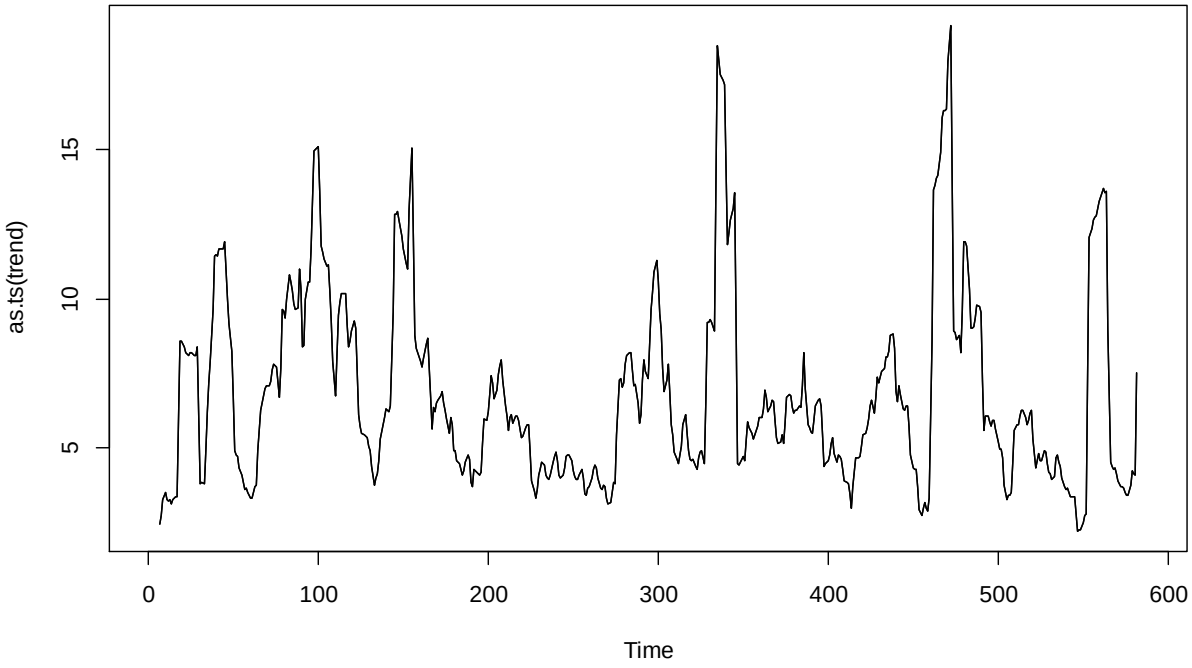
- To reduce absenteeism, the company needs to hire more qualified and healthier workforce.
- Disciplinary action should be taken as needed to reduce absenteeism.
- Company should try to retain more experienced workers as they are less likely to be absent.
- More insights are given in section 2.1.4.1
- Based on the previous year's data, absenteeism can be predicted as a time series model. Predictions are attached separately.

Appendix A - Extra Figures

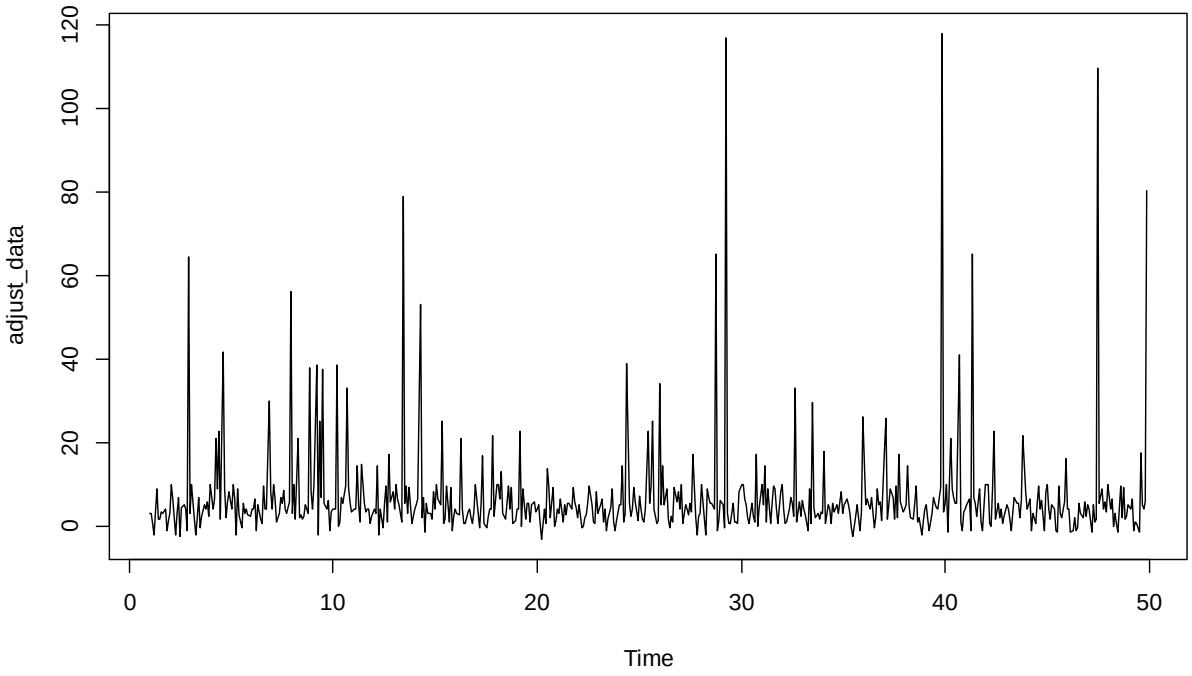
We potting some extra figures for better perspective of data



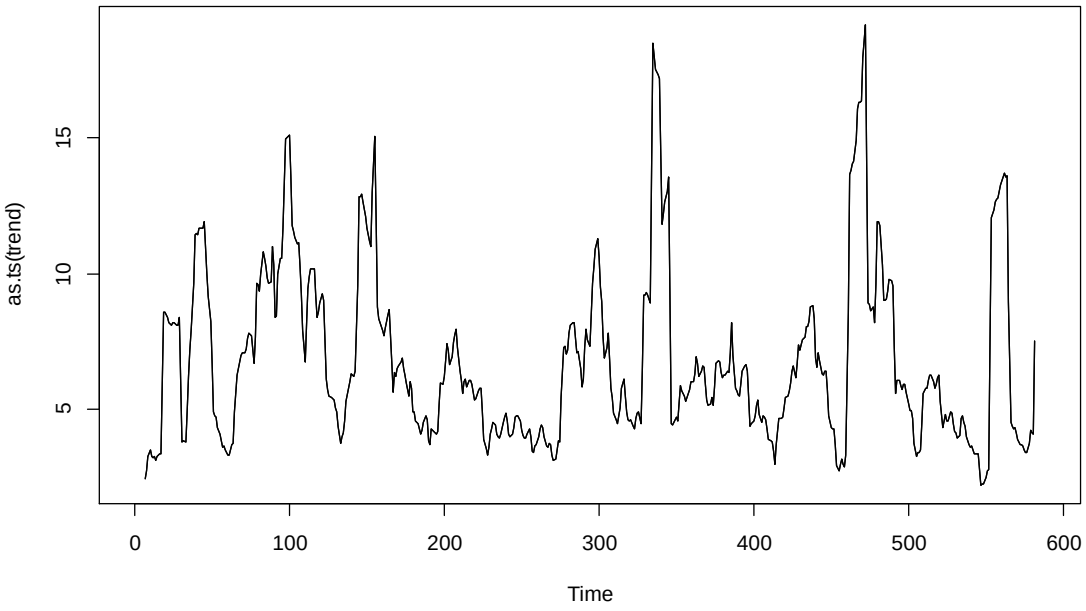
Pair plot for all variables



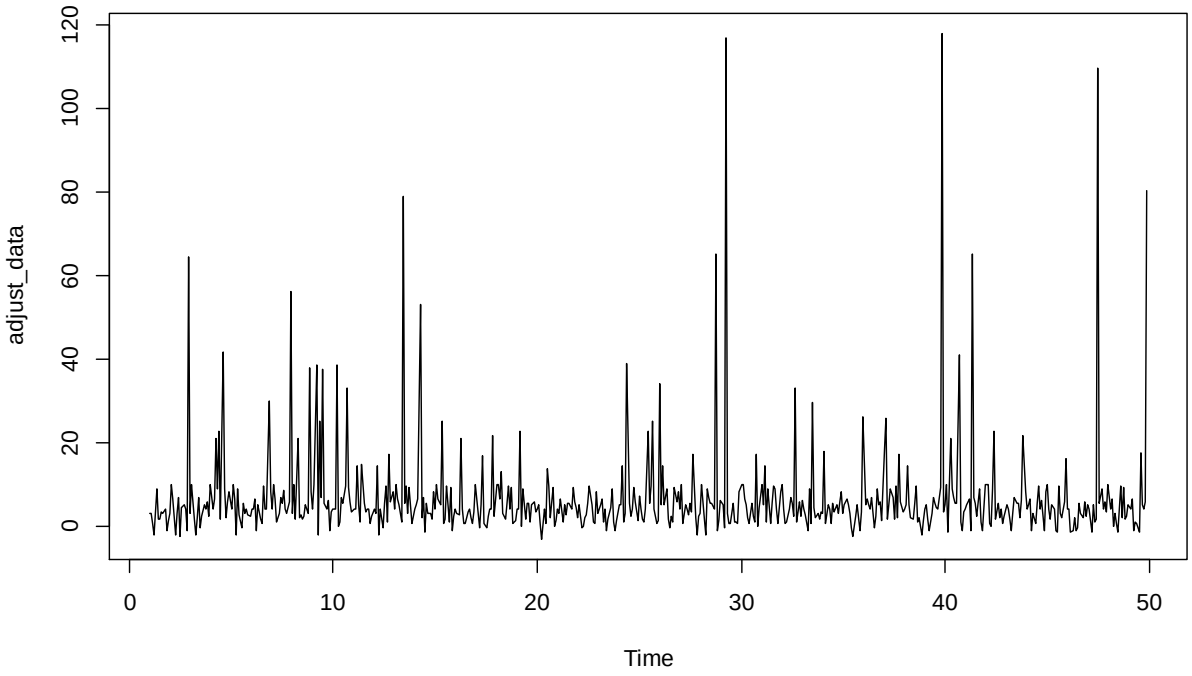
Data converted to time series



Time series data after removing seasonality



Forecasted time series data



Forecasted time series data adjusted

Appendix B - Python Code

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jul 25 14:58:05 2019

@author: Abhishek Mandal
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from IPython import get_ipython
get_ipython().run_line_magic('matplotlib', 'inline')
import os
import seaborn as sns


#Importing data
os.getcwd()
os.chdir('D:\\Data Science\\Datasets\\EmployeeAbsentism_Edwisor')
data = pd.read_excel("Absenteeism_at_work_Project.xls")


#Data wrangling
data['Month of absence'].head()
data.dtypes

data['Month of absence'].unique()



#Sorting by month as its a cyclic data
data = data.sort_values('Month of absence')

##Analysing the data
#Missing value analysis

data.isnull().sum().sort_values(ascending=False)
```

```
#Replacing 0 with modefor Month of absence column
month_mode = data.loc[:,"Month of absence"].mode()
month_mode = pd.to_numeric(month_mode, errors ='ignore')
data=data.replace({'Month of absence': {0: 3.0}})


#Replace all missing values with mode
cols = list(data.columns)
print(cols)
for col in cols:
   data[col] = data[col].fillna(data[col].mode().iloc[0])
   print(col)

#Dropping ID and saving it to variable
data_id = data['ID']
data = data.drop('ID', axis=1)


#Outlier analysis
#from scipy import stats
#data = data[(np.abs(stats.zscore(data)) < 3).all(axis=1)] #Remove all rows which have
outlier in atleast 1 column

##Plotting diagrams
corrmat = data.corr(method='pearson')
#Absenteesm time in hours correlation matrix
corr_num = 20 #number of variables for heatmap
cols_corr = corrmat.nlargest(corr_num, 'Absenteeism time in hours')['Absenteeism
time in hours'].index
corr_mat_sales = np.corrcoef(data[cols_corr].values.T)
sns.set(font_scale=1.25)
f, ax = plt.subplots(figsize=(12, 9))
hm = sns.heatmap(corr_mat_sales, cbar=True, annot=True, square=True, fmt='.2f',
annot_kws={'size': 7}, yticklabels=cols_corr.values, xticklabels=cols_corr.values)
plt.show()

# Pairplot for the most intresting parameters
# pair plots for variables with largest correlation
var_num = 8
vars = cols_corr[0:var_num]

sns.set()
sns.pairplot(data[vars], size = 2.5)
plt.show();
```

```
#Bar chart
cols.remove('Absenteeism time in hours')
for col in cols:
   data.groupby(col).sum().plot(y='Absenteeism time in hours', kind='bar')

#Removing overfit
# Removes colums where the threshold of zero's is (> 99.95), means has only zero
values
X = data
overfit = []
for i in X.columns:
   counts = X[i].value_counts()
   zeros = counts.iloc[0]
   if zeros / len(X) * 100 > 99.95:
      overfit.append(i)

overfit = list(overfit)
print(overfit)        #Overfit is empty i.e. no overfitting

##Feature Engineering
#Calculating correlation coeff for highly correlated values
np.corrcoef(data['Height'], data['Body mass index'])  #-0.1202946
np.corrcoef(data['Weight'], data['Body mass index'])  #0.86293245
np.corrcoef(data['Age'], data['Service time'])  #0.66102651

#As we can see Weight/bmi and Age/Service time are highly correlated, so we can
drop one of the values
data = data.drop('Service time', axis=1)
data = data.drop('Weight', axis=1)

##Model creation
#Creating test and train data
X_train = data.drop('Absenteeism time in hours', axis = 1)
Y_train = data['Absenteeism time in hours']
X_test = X_train


from datetime import datetime
from sklearn.preprocessing import RobustScaler
from sklearn.model_selection import KFold, cross_val_score
from sklearn.metrics import mean_squared_error, make_scorer, accuracy_score
from sklearn.linear_model import ElasticNetCV, LassoCV, RidgeCV
from sklearn.pipeline import make_pipeline
```

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import GradientBoostingRegressor
from mlxtend.regressor import StackingCVRegressor
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor


#Defining folds and score functions

kfolds = KFold(n_splits=10, shuffle=True, random_state=42)

# model scoring and validation function
def cv_rmse(model, X_train=X_train):
   rmse = np.sqrt(-cross_val_score(model, X_train,
Y_train,scoring="neg_mean_squared_error",cv=kfolds))
   return (rmse)

# rmsle scoring function
def rmsle(y, y_pred):
   return np.sqrt(mean_squared_error(y, y_pred))

#Defining models
lightgbm = LGBMRegressor(objective='regression',
                         num_leaves=4, #was 3
                         learning_rate=0.01,
                         n_estimators=10000,
                         max_bin=200,
                         bagging_fraction=0.75,
                         bagging_freq=5,
                         bagging_seed=7,
                         feature_fraction=0.2, # 'was 0.2'
                         feature_fraction_seed=7,
                         verbose=-1,
                         )

xgboost = XGBRegressor(learning_rate=0.01,n_estimators=3460,
                       max_depth=3, min_child_weight=0,
                       gamma=0, subsample=0.7,
                       colsample_bytree=0.7,
                       objective='reg:linear', nthread=-1,
                       scale_pos_weight=1, seed=27,
                       reg_alpha=0.00006)
```

```
# setup models hyperparameters using a pipline
# The purpose of the pipeline is to assemble several steps that can be cross-validated
together, while setting different parameters.
# This is a range of values that the model considers each time in runs a CV
e_alphas = [0.0001, 0.0002, 0.0003, 0.0004, 0.0005, 0.0006, 0.0007]
e_l1ratio = [0.8, 0.85, 0.9, 0.95, 0.99, 1]
alphas_alt = [14.5, 14.6, 14.7, 14.8, 14.9, 15, 15.1, 15.2, 15.3, 15.4, 15.5]
alphas2 = [5e-05, 0.0001, 0.0002, 0.0003, 0.0004, 0.0005, 0.0006, 0.0007, 0.0008]



# Kernel Ridge Regression : made robust to outliers
ridge = make_pipeline(RobustScaler(), RidgeCV(alphas=alphas_alt, cv=kfolds))

# LASSO Regression : made robust to outliers
lasso = make_pipeline(RobustScaler(), LassoCV(max_iter=1e7,
            alphas=alphas2,random_state=42, cv=kfolds))

# Elastic Net Regression : made robust to outliers
elasticnet = make_pipeline(RobustScaler(), ElasticNetCV(max_iter=1e7,
                alphas=e_alphas, cv=kfolds, l1_ratio=e_l1ratio))


stack_gen = StackingCVRegressor(regressors=(ridge, elasticnet, lightgbm),
                    meta_regressor=elasticnet,
                    use_features_in_secondary=True)

# store models, scores and prediction values
models = {'Ridge': ridge,
      'Lasso': lasso,
      'ElasticNet': elasticnet,
      'lightgbm': lightgbm,
      'xgboost': xgboost}
predictions = {}
scores = {}

#Training the models
for name, model in models.items():

   model.fit(X_train, Y_train)
   predictions[name] = np.expm1(model.predict(X_test))
   score = cv_rmse(model, X_train=X_train)
```

```
    scores[name] = (score.mean(), score.std())

#Validating and training each model
# get the performance of each model on training data(validation set)
print('---- Score with CV_RMSLE-----')
score = cv_rmse(ridge)
print("Ridge score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))

score = cv_rmse(lasso)
print("Lasso score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))

score = cv_rmse(elasticnet)
print("ElasticNet score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))

score = cv_rmse(lightgbm)
print("lightgbm score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))

score = cv_rmse(xgboost)
print("xgboost score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))


#Fit the training data X, y
print('----START Fit----',datetime.now())
print('Elasticnet')
elastic_model = elasticnet.fit(X_train, Y_train)
print('Lasso')
lasso_model = lasso.fit(X_train, Y_train)
print('Ridge')
ridge_model = ridge.fit(X_train, Y_train)
print('lightgbm')
lgb_model_full_data = lightgbm.fit(X_train, Y_train)

print('xgboost')
xgb_model_full_data = xgboost.fit(X_train, Y_train)


print('stack_gen')
stack_gen_model = stack_gen.fit(np.array(X_train), np.array(Y_train))

#Blend model prediction
def blend_models_predict(X_test):
    return ((0.2  * elastic_model.predict(X_test)) + \
            (0.25 * lasso_model.predict(X_test)) + \
            (0.2 * ridge_model.predict(X_test)) + \
```

```
        (0.15 * lgb_model_full_data.predict(X_test)) + \
         (0.1 * xgb_model_full_data.predict(X_test)) + \
        (0.2 * stack_gen_model.predict(np.array(X_test))))
pred =  blend_models_predict(X_test)

print('RMSLE score on train data:')
print(rmsle(Y_train,pred))


error = mean_squared_error(Y_train, pred)
print('Test MSE: %.3f' % error)



from sklearn.metrics import r2_score
r2score = r2_score(Y_train, pred,sample_weight=None,
multioutput='uniform_average')
print('R Squared value: %.3f' % r2score)

#MAPE
num = abs(pred-Y_train).sum()
den = abs(Y_train).sum()

mape = (num/den)*100
print('MAPE: %3f' %mape)


# plotting test vs prediction
Y_train = Y_train.to_frame().reset_index()
plt.plot(Y_train)
plt.plot(pred, color='red')
plt.show()
```

R Code

Note: All preprocessing has been done in r so only feature engineering (from the insights from python) has been done here

```
rm(list=ls())
setwd("D:/Data
Science/Datasets/EmployeeAbsentism_Edwisor")
getwd()


install.packages(c("dplyr","DMwR","ggplot2"))
library("dplyr") #used for manipulationg select arrange
library("DMwR") #used for msining with R
library("ggplot2") #used for plotting the graphs
install.packages("DMwR")
library("VIM")

#reading file df
df = read.csv("Absent.csv", header = TRUE)

#replacing missing values with data
for(i in 1:ncol(df)){
  df[is.na(df[,i]), i] <- mean(df[,i], na.rm = TRUE)
}

#Dropping highly correlated values

df$Weight<- NULL
df$Age<- NULL

#Transforming data to time series data

df1 = df[df$Month.of.absence == 1, ]
df2 = df[df$Month.of.absence == 2, ]
df3 = df[df$Month.of.absence == 3, ]
df4 = df[df$Month.of.absence == 4, ]
```

```
df5 = df[df$Month.of.absence == 5, ]
df6 = df[df$Month.of.absence == 6, ]
df7 = df[df$Month.of.absence == 7, ]
df8 = df[df$Month.of.absence == 8, ]
df9 = df[df$Month.of.absence == 9, ]
df10 = df[df$Month.of.absence == 10, ]
df11 = df[df$Month.of.absence == 11, ]
df12 = df[df$Month.of.absence == 12, ]

#Create new dataframe for time series analysis

new = data.frame()

for (i in 0:49){
  new <- rbind(new, df1[i, ])
  new <- rbind(new, df2[i, ])
  new <- rbind(new, df3[i, ])
  new <- rbind(new, df4[i, ])
  new <- rbind(new, df5[i, ])
  new <- rbind(new, df6[i, ])
  new <- rbind(new, df7[i, ])
  new <- rbind(new, df8[i, ])
  new <- rbind(new, df9[i, ])
  new <- rbind(new, df10[i, ])
  new <- rbind(new, df11[i, ])
  new <- rbind(new, df12[i, ])

}

#Save id and drop it

id = data.frame()
id = new$ID
new$ID<- NULL

#Plot to check moving average
install.packages("forecast")
```

```
library(forecast)
trend = ma(new$Absenteeism.time.in.hours, order = 12,centre
= T)
plot(as.ts(trend))
lines(trend)
#Decompose seasonality
install.packages("fpp")
library(fpp)
ts_data = ts(new$Absenteeism.time.in.hours, frequency = 12,
start = 1)
decompose_data = decompose(ts_data, "additive")
adjust_data = ts_data-decompose_data$seasonal
plot(adjust_data)


##Creating the ARIMA model
#Generating values for ARIMA
#Getting the auto correlation factor
p = diff(log(adjust_data)) #Differentiating
l = log(adjust_data)
is.na(p) <- sapply(p, is.infinite)        #Removing Inf values
is.na(l) <- sapply(l, is.infinite)

sum(is.na(l))

p<- na.omit(p)        #Removing NA values
l<- na.omit(l)
acf(p)    #Autocorrelation factor = 0

pacf(p)    #pacf factor = 5

length(l)


fit<- arima(l, c(5,1,0), seasonal = list(order = c(5,1,0), period =
12))
pred<- predict(fit, n.ahead = 49*12)
```

```
#Plot to check moving average
install.packages("forecast")
library(forecast)
trend = ma(pred, order = 12,centre = T)
plot(as.ts(trend))
lines(trend)
#Decompose seasonality
install.packages("fpp")
library(fpp)
ts_data = ts(pred, frequency = 12, start = 1)
decompose_data = decompose(ts_data, "additive")
adjust_data = ts_data-decompose_data$seasonal
plot(adjust_data)

#Creating final dataframe for storing predicted values

final = data.frame()
final = as.data.frame(adjust_data)
final$id <- paste(final$x, id)

final$pred<- NULL

final$x<- as.integer(final$x)
final$month <- paste(new$Month.of.absence)

final$x[final$x < 0] <- 0 # Converting any -ve values to 0

#Calculating errors : Mean Average Percent Error and r squared
mape <- (sum(abs(final$x-
new$Absenteeism.time.in.hours))/sum(abs(new$Absenteeism.t
ime.in.hours)))*100  #23.28% error
rsq <- cor(final$x, new$Absenteeism.time.in.hours)^2 #0.977 =
97.7% accuracy

#Plotting
```

```
library(ggplot2)
theme_set(theme_minimal())
ggplot() +
  geom_point(data = final, aes(x = month, y = x), color = "red") +
  geom_point(data = new, aes(x = Month.of.absence, y =
Absenteeism.time.in.hours), color = "blue") +
  ylab('Absenteeism.time.in.hours') +
  xlab('Month')+
  geom_path()

ggplot() +
  geom_count(data = final, aes(x = month, y = x), color = "red") +
  geom_count(data = new, aes(x = Month.of.absence, y =
Absenteeism.time.in.hours), color = "blue") +
  ylab('Absenteeism.time.in.hours') +
  xlab('Month')+
  geom_path()



write.csv(final, "predictions_timeseries.csv", row.names = F)
```

Appendix C

More details on the algorithms used for modelling

Regression

- **XGboost** XGBoost stands for e**X**treme **G**radient **B**oosting. It is an implementation of gradient boosting machines created by Tianqi Chen, now with contributions from many developers. It belongs to a broader collection of tools under the umbrella of the Distributed Machine Learning Community or DMLC who are also the creators of the popular mxnet deep learning library. The library is laser focused on computational speed and model performance, as such there are few frills. Nevertheless, it does offer a number of advanced features. The implementation of the model supports the features of the scikit-learn and R implementations, with new additions like regularization. Three main forms of gradient boosting are supported:
  Gradient Boosting algorithm also called gradient boosting machine including the learning rate.
  Stochastic Gradient Boosting with sub-sampling at the row, column and column per split levels.

  Regularized Gradient Boosting with both L1 and L2 regularization.
  The implementation of the algorithm was engineered for efficiency of compute time and memory resources. A design goal was to make the best use of available resources to train the model. Some key algorithm implementation features include:

  Sparse Aware implementation with automatic handling of missing data values.
  Block Structure to support the parallelization of tree construction.
  Continued Training so that you can further boost an already fitted model on new data.

  ***Source**:* https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

- **LightGBM** Light GBM is a gradient boosting framework that uses tree based learning algorithm. **Light GBM grows tree vertically** while other algorithm grows trees horizontally meaning that Light GBM grows tree **leaf-wise** while other algorithm grows level-wise. It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm.

Source: https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc

- **LASSO (least absolute shrinkage and selection operator)** In statistics and machine learning, lasso (least absolute shrinkage and selection operator; also Lasso or LASSO) is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the statistical model it produces. It was originally introduced in geophysics literature in 1986 and later independently rediscovered and popularized in 1996 by Robert Tibshirani who coined the term and provided further insights into the observed performance.

  Lasso was originally formulated for least squares models and this simple case reveals a substantial amount about the behavior of the estimator, including its relationship to ridge regression and best subset selection and the connections between lasso coefficient estimates and so-called soft thresholding. It also reveals that (like standard linear regression) the coefficient estimates do not need to be unique if covariates are collinear.

  Though originally defined for least squares, lasso regularization is easily extended to a wide variety of statistical models including generalized linear models, generalized estimating equations, proportional hazards models, and M-estimators, in a straightforward fashion. Lasso's ability to perform subset selection relies on the form of the constraint and has a variety of interpretations including in terms of geometry, Bayesian statistics, and convex analysis.
  Source: https://en.wikipedia.org/wiki/Lasso_(statistics)

- **Elastic net** In statistics and, in particular, in the fitting of linear or logistic regression models, the elastic net is a regularized regression method that linearly combines the L1 and L2 penalties of the lasso and ridge methods.
  The elastic net method overcomes the limitations of the LASSO (least absolute shrinkage and selection operator) method which uses a penalty function based on

  $$\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|.$$

  Use of this penalty function has several limitations.[1] For example, in the "large p, small n" case (high-dimensional data with few examples), the LASSO selects at most n variables before it saturates. Also if there is a group of highly correlated variables, then the LASSO tends to select one variable from a group and ignore the others. To overcome these limitations, the elastic net adds a quadratic part to the penalty β^2 which when used alone is ridge regression (known also as Tikhonov regularization). The estimates from the elastic net method are defined by

  $$\hat{\beta} \equiv \underset{\beta}{\operatorname{argmin}}(\|y - X\beta\|^2 + \lambda_2\|\beta\|^2 + \lambda_1\|\beta\|_1).$$

Source: https://en.wikipedia.org/wiki/Elastic_net_regularization

Time series

- **ARIMA (autoregressive integrated moving average)** model is a generalization of an autoregressive moving average (ARMA) model. Both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting). ARIMA models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity.

  The AR part of ARIMA indicates that the evolving variable of interest is regressed on its own lagged (i.e., prior) values. The MA part indicates that the regression error is actually a linear combination of error terms whose values occurred contemporaneously and at various times in the past. The **I** (for "integrated") indicates that the data values have been replaced with the difference between their values and the previous values (and this differencing process may have been performed more than once). The purpose of each of these features is to make the model fit the data as well as possible.

  Non-seasonal ARIMA models are generally denoted ARIMA (p, d, q) where parameters p, d, and q are non-negative integers, p is the order (number of time lags) of the autoregressive model, d is the degree of differencing (the number of times the data have had past values subtracted), and q is the order of the moving-average model. Seasonal ARIMA models are usually denoted ARIMA(p,d,q)(P,D,Q)m, where m refers to the number of periods in each season, and the uppercase P,D,Q refer to the autoregressive, differencing, and moving average terms for the seasonal part of the ARIMA model.

  When two out of the three terms are zeros, the model may be referred to based on the non-zero parameter, dropping "AR", "I" or "MA" from the acronym describing the model. For example, ARIMA (1,0,0) is AR(1), ARIMA(0,1,0) is I(1), and ARIMA(0,0,1) is MA(1).

  ARIMA models can be estimated following the Box–Jenkins approach.

References

https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/

https://lightgbm.readthedocs.io/en/latest/

https://xgboost.readthedocs.io/en/latest/