Absenteeism time in hours

Introduction

Every work place can't help but have workers who, depending on various reasons, decide to be absent. Why is that? Is there a more recurring reason across various subjects?

According to Forbes, Absenteeism is an employee's intentional or habitual absence from work where excessive absences can equate to decreased productivity and can have a major effect on company finances, morale and other factors. There are different types of reasons that can lead to an absence at work such as Illness, Childcare, etc.

This project will be looking at a dataset from Brazil where records of absenteeism at work were collected from July 2007 to July 2010 at a courier company. The goal is to **find out how the company can increase productivity by decreasing absenteeism at work.**

A few questions to consider along the way:

- Which areas of life affect Abseenteeism(i.e: Work or Family..)?
- Is there an obvious relationship between reason for absence and absenteeism?
- Between regression models and Classification models, is there a better model for our problem?

Data Exploration

This dataset consists of **740 observations** and **21 features** with:

- **8 Categorical features**: Reason for absence,Month of Absence, Day of the week, Seasons, Disciplinary failure,Education,Social drinker and Social smoker
- **13 numerical features**: ID, Transportation Expense , Distance from Residence to Work,Service time, Age , Work load , Hit target , Son, Pet, Weight, Height, Body mass index and Absenteeism time in hours

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).

1. Month of absence
2. Day of the week (Monday (2), Tuesday (3), Wednesday (4), Thursday (5), Friday (6))
3. Seasons
4. Transportation expense
5. Distance from Residence to Work (kilometers)
6. Service time
7. Age
8. Work load Average/day
9. Hit target
10. Disciplinary failure (yes=1; no=0)
11. Education (high school (1), graduate (2), postgraduate (3), master and doctor (4))
12. Son (number of children)
13. Social drinker (yes=1; no=0)
14. Social smoker (yes=1; no=0)
15. Pet (number of pet)
16. Weight
17. Height

18. Body mass index
19. Absenteeism time in hours (target)

In [262]:
```
#Importing libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import seaborn as sns
from sklearn.preprocessing import StandardScaler
%matplotlib inline

#First 10 rows
absent=pd.read_csv('./Absenteeism_at_work - Absenteeism_at_work.csv',sep
=',')
print (type(absent))
absent.head(5)
```

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

Out[262]:

	ID	Reason for absence	Month of absence	Day of the week	Seasons	Transportation expense	Distance from Residence to Work	Service time	Age	Work load Average/day
0	11	26	7	3	1	289	36	13	33	239,554
1	36	0	7	3	1	118	13	18	50	239,554
2	3	23	7	4	1	179	51	18	38	239,554
3	7	7	7	5	1	279	5	14	39	239,554
4	11	23	7	5	1	289	36	13	33	239,554

5 rows × 21 columns

Pre Processing

In order to build a good model, we need to have some understanding of the dataset using the Exploratory Data Analysis Process (EDA). For the first step, we will be looking at any necessary data cleaning.

In [263]:
```
#changing a float 64 to int64
absent['Work load Average/day ']=absent['Work load Average/day '].astype
('str')
absent['Work load Average/day ']=absent['Work load Average/day '].str.re
place(',', '')
absent['Work load Average/day ']=pd.to_numeric(absent['Work load Averag
e/day '], errors='coerce')
#absent['Work load Average/day ']
```

In [264]:
```
absent.isnull().any()
```

Out[264]:
```
ID                                 False
Reason for absence                 False
Month of absence                   False
Day of the week                    False
Seasons                            False
Transportation expense             False
Distance from Residence to Work    False
Service time                       False
Age                                False
Work load Average/day              False
Hit target                         False
Disciplinary failure               False
Education                          False
Son                                False
Social drinker                     False
Social smoker                      False
Pet                                False
Weight                             False
Height                             False
Body mass index                    False
Absenteeism time in hours          False
dtype: bool
```

Thankfully, our dataset does not have any missing values and column **'Work load Average/day'** is tranformed from **float 64 to int 64** in order to be used for analysis.

After that, a general analysis with descriptive statistics is made in order to capture the shape or tendency of each column.

In [592]:
```
absent.describe()
```

Out[592]:

	ID	Reason for absence	Month of absence	Day of the week	Seasons	Transportation expense	Distance from Residence to Work
count	740.000000	740.000000	740.000000	740.000000	740.000000	740.000000	740.000000
mean	18.017568	19.216216	6.324324	3.914865	2.544595	221.329730	29.631081
std	11.021247	8.433406	3.436287	1.421675	1.111831	66.952223	14.836788
min	1.000000	0.000000	0.000000	2.000000	1.000000	118.000000	5.000000
25%	9.000000	13.000000	3.000000	3.000000	2.000000	179.000000	16.000000
50%	18.000000	23.000000	6.000000	4.000000	3.000000	225.000000	26.000000
75%	28.000000	26.000000	9.000000	5.000000	4.000000	260.000000	50.000000
max	36.000000	28.000000	12.000000	6.000000	4.000000	388.000000	52.000000

8 rows × 21 columns

A few Observation Points:

- We can tell that the maximum value of 'Absenteeism time in hours is **120** which is more than **9 standard deviations away from the mean!**
- The minimum values for **Reason for absence and Month of absence are 0's (unexpected).**

A closer look of these 2 columns is taken:

```
In [266]: len(absent[absent['Absenteeism time in hours']==0])
```

```
Out[266]: 44
```

```
In [267]: len(absent[absent['Reason for absence']==0])
```

```
Out[267]: 43
```

```
In [268]: absent[absent['Month of absence']==0]
```

Out[268]:

	ID	Reason for absence	Month of absence	Day of the week	Seasons	Transportation expense	Distance from Residence to Work	Service time	Age	Work lo Average/d
737	4	0	0	3	1	118	14	13	40	2712
738	8	0	0	4	2	231	35	14	39	2712
739	35	0	0	6	3	179	45	14	53	2712

3 rows × 21 columns

- A closer inspection of those two columns reveals that all the 0's in both **Absenteeism time in hours,Month of absence and Reason for absence columns** correpond to **Absenteeism time in hours=0** . If this conclusion were to represent every employee who didn't take any absent days, the Day of the week column shouldn't be populated. I decided to categorize these **44 rows as outliers.**
- Doing so also takes out all the **Disciplinary failure==1** leaving us with only Disciplinary failure==0. This means that this feature is no longer needed for the analysis.

The rest of the analysis will be looking at the non zero values of Absenteeism with a new dataset called **allabsents**

```
In [516]: #new dataset with no Absenteeism time in hours=0 and Disciplinary failur
          e

          allabsents=absent[absent['Absenteeism time in hours']!=0]
          allabsents=allabsents.drop(['Disciplinary failure'],axis=1)
```

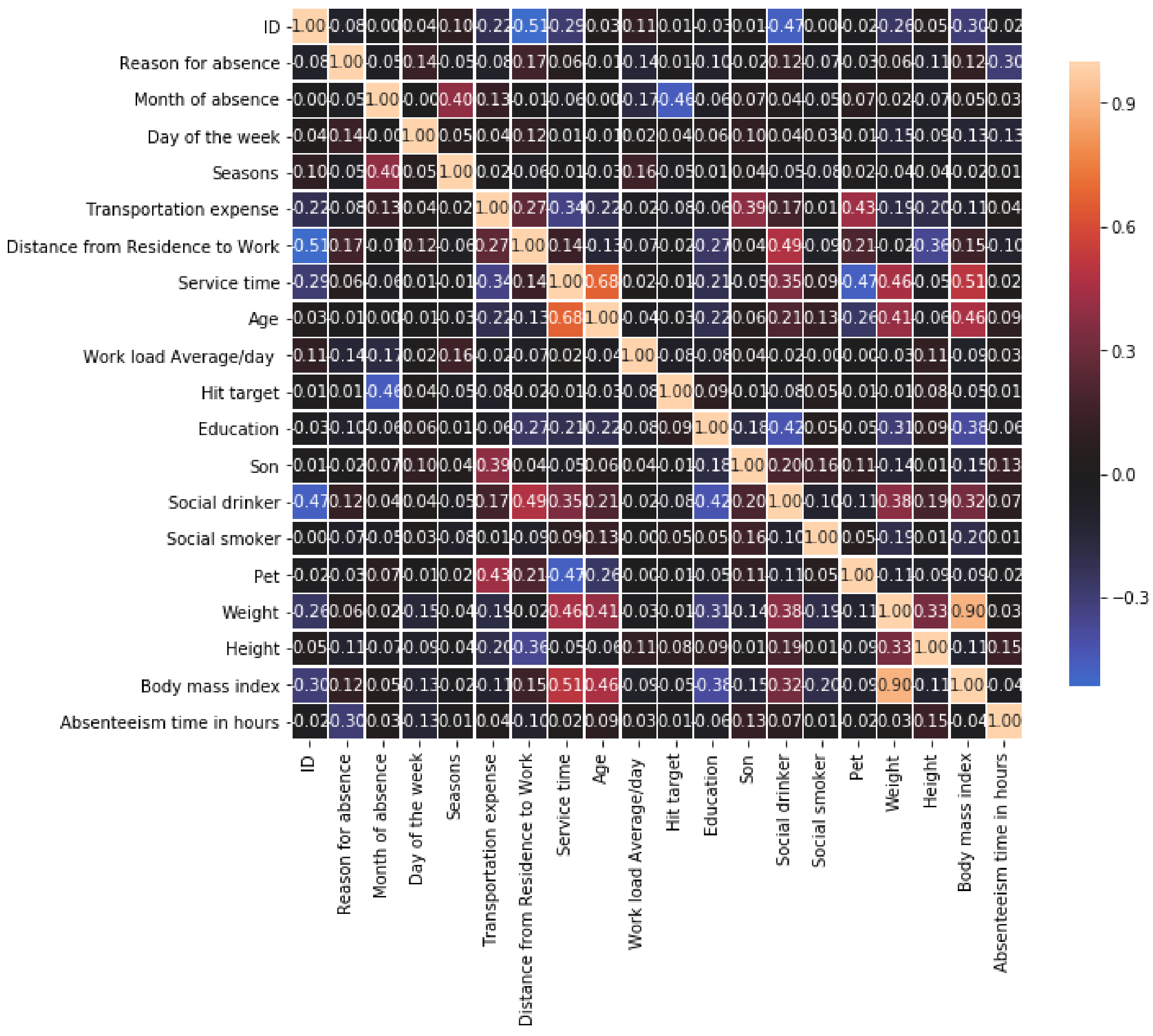
Before looking at specific statistics for each variable, a heatmap gives an overall sense of a correlation between features.

- The most positive correlations are: Service and Body mass Index,Weight and Body mass Index, Age and Service time
- The most negative correlations are: Reason for absence and Disciplinary failure, Month of Absence and Hit Target

In [270]:
```
def correlation_heatmap(li):
    correlations = li.corr()

    fig, ax = plt.subplots(figsize=(10,10))
    sns.heatmap(correlations, vmax=1.0, center=0, fmt='.2f',
                square=True, linewidths=.5, annot=True, cbar_kws={"shrin
k": .70})
    plt.show();

correlation_heatmap(allabsents)
```



After that, a statistical focus looks at the dynamics of employees according to Age, Service time and Absenteeism time in hours

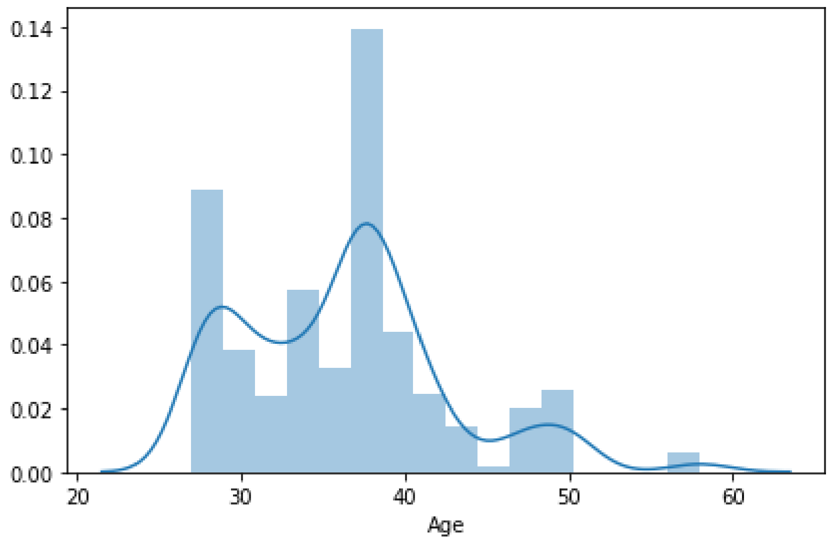
- Most workers are in between their late 20's and early 40's. On top of that, most employees have between 10 to 18 years of service time.

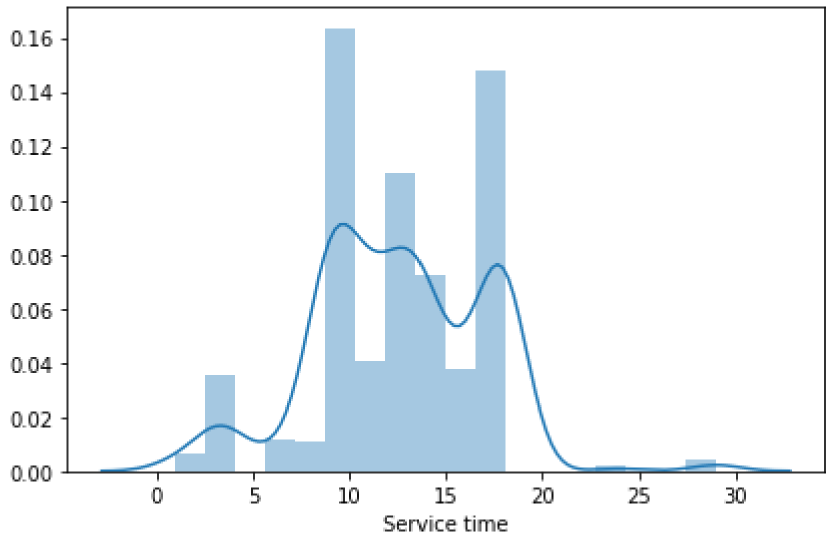
It is worth noticing a few characteristics of the Absenteeism time in hours column:

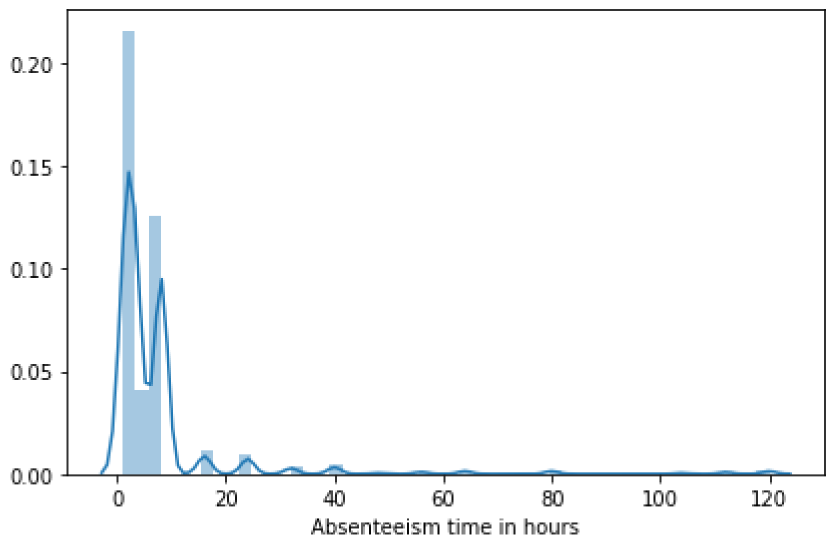
- The plot is more skewed to the right where most values are less than 80 hrs of absence. In fact, **61% of employees have been absent for less than 8hrs** and **29% of them absent for exactly 8hrs**. Hence **91% of Absenteeism time in hours is explained by ony 8 or less hours of absenteeism**.
- There is a trend in Absenteeism time in hours. After 7hrs, there is mostly an **8hr difference (or 1 day)** between each reported absence.

In [524]:
```
col=['Age','Service time','Absenteeism time in hours']
for i in col:
    sns.distplot(allabsents[i])
    plt.tight_layout()
    plt.show()
```

```
/Users/marlynehakizimana/anaconda3/lib/python3.7/site-packages/scipy/st
ats/stats.py:1713: FutureWarning: Using a non-tuple sequence for multid
imensional indexing is deprecated; use `arr[tuple(seq)]` instead of `ar
r[seq]`. In the future this will be interpreted as an array index, `arr
[np.array(seq)]`, which will result either in an error or a different r
esult.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```





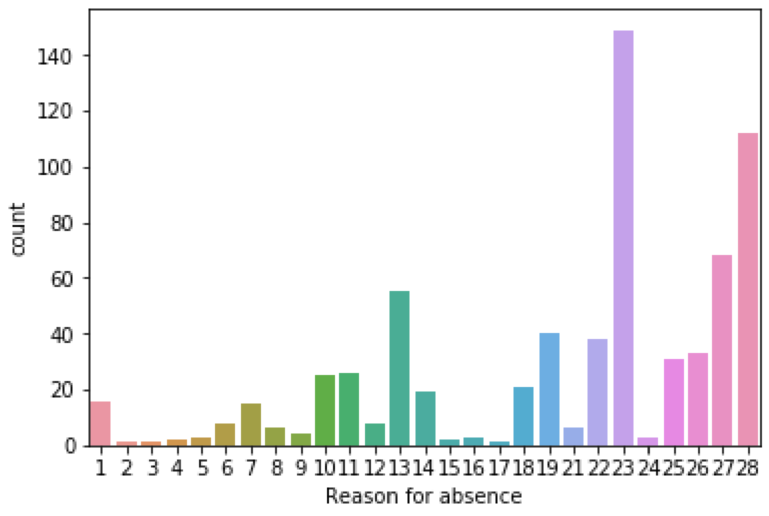


Another interesting column is **Reason for absence** with its 28 levels with each number explaining the reason for absence. The column had originally categorical input that was converted into numerical input. Full explanation of each level is under attribute information.

- Most common reasons were the categories outside of the ICD (International Code of Diseases) meaning that the most reasons were non serious diseases such as :patient follow-up (22), medical consultation (23), blood donation (24), laboratory examination (25), unjustified absence (26), physiotherapy (27), dental consultation (28). These 7 categories were used by 62% of employees.
- The 2 categories within ICD with relatively high numbers are **13:Diseases of the musculoskeletal system and connective tissue and 19:Injury, poisoning and certain other consequences of external causes**. These two seem like they could be related to injuries one gets from this type of industry. Someone who is always doing deliveries for a long period of time is more likely to be affected.
- Surprising how only a few absences are related to birthing or children related.

In [525]:
```
sns.countplot(x='Reason for absence',data=allabsents)
```

Out[525]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1dcde550>
```



With this information, I wanted to analyze how depending on the reason, insights from other features can be found: Is there a specific category of reasons with less absenteeism?

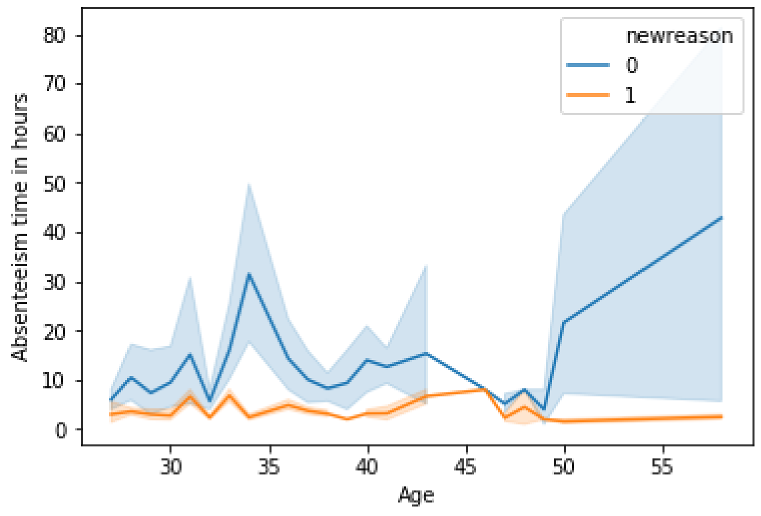
I decided to create two categories of Reason for absence where one class represents ICD and the other non ICD reason.

In [531]:
```
bo=[]
for i in allabsents['Reason for absence']:
    if i<22:
        bo.append(0)
    else:
        bo.append(1)
allabsents['newreason']=bo
```

In [552]:
```
sns.lineplot(x='Age',y='Absenteeism time in hours',hue='newreason',data=
allabsents)
```

```
/Users/marlynehakizimana/anaconda3/lib/python3.7/site-packages/scipy/st
ats/stats.py:1713: FutureWarning: Using a non-tuple sequence for multid
imensional indexing is deprecated; use `arr[tuple(seq)]` instead of `ar
r[seq]`. In the future this will be interpreted as an array index, `arr
[np.array(seq)]`, which will result either in an error or a different r
esult.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

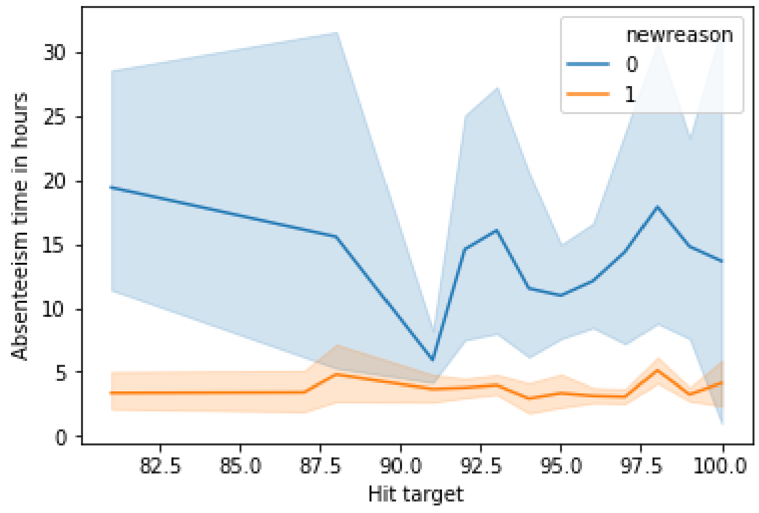
Out[552]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a3659deb8>
```



In [553]:
```
sns.lineplot(x='Hit target',y='Absenteeism time in hours',hue='newreaso
n',data=allabsents)
```

```
/Users/marlynehakizimana/anaconda3/lib/python3.7/site-packages/scipy/st
ats/stats.py:1713: FutureWarning: Using a non-tuple sequence for multid
imensional indexing is deprecated; use `arr[tuple(seq)]` instead of `ar
r[seq]`. In the future this will be interpreted as an array index, `arr
[np.array(seq)]`, which will result either in an error or a different r
esult.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

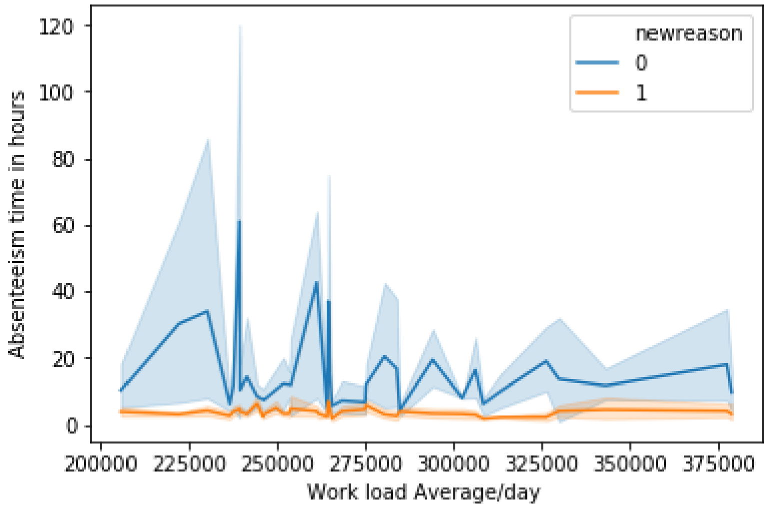
Out[553]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a365261d0>
```



In [544]:
```
sns.lineplot(x='Work load Average/day ',y='Absenteeism time in hours',hu
e='newreason',data=allabsents)
```

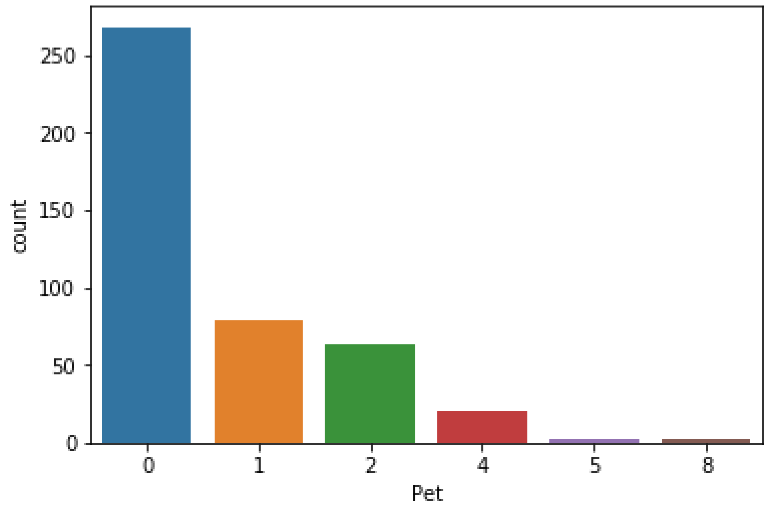
```
/Users/marlynehakizimana/anaconda3/lib/python3.7/site-packages/scipy/st
ats/stats.py:1713: FutureWarning: Using a non-tuple sequence for multid
imensional indexing is deprecated; use `arr[tuple(seq)]` instead of `ar
r[seq]`. In the future this will be interpreted as an array index, `arr
[np.array(seq)]`, which will result either in an error or a different r
esult.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[544]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a34cd03c8>
```



In [591]:
```
sns.countplot(x='Pet',data=allabsents[allabsents['newreason']==1])
```

Out[591]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a3755acf8>
```

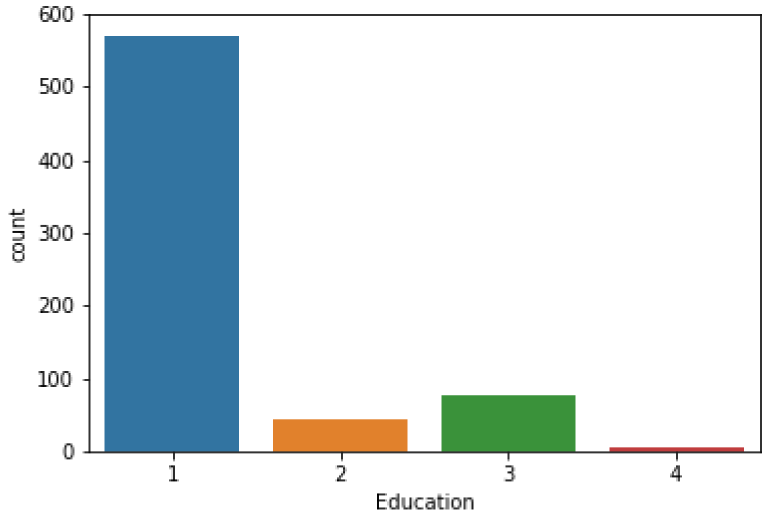


The above plots show how both Age, Hit target and Work load Average/day have high absenteeism time in hours for ICD group and low absenteeism time in hours for non ICD group.

I also explored other categorical columns such as education and social drinker.

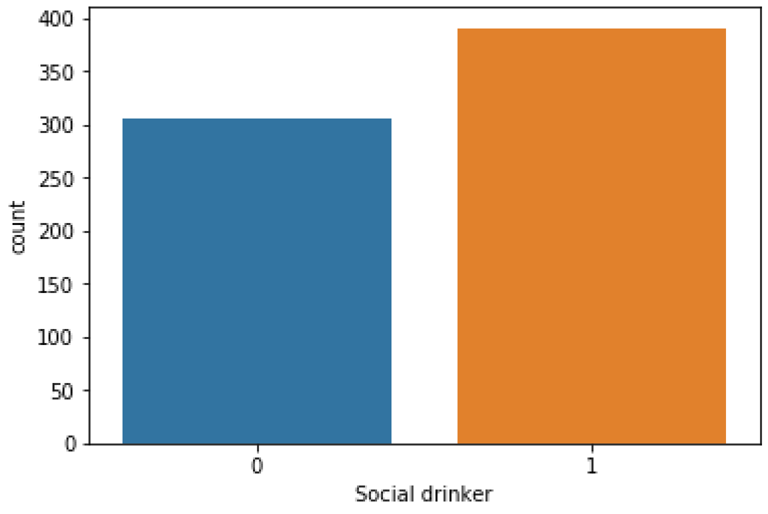
```
In [284]: sns.countplot(x='Education',data=allabsents)

Out[284]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1be64dd8>
```



```
In [285]: sns.countplot(x='Social drinker',data=allabsents)

Out[285]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1be64c88>
```



The next steps is to normalize our data. A few ways were used in this process:

- Log transform and normalization for the skewed depeendent variable Absenteeism time in hours
- One Hot Encoding for independent categorical variables
- StandardScaler() for all independent variables

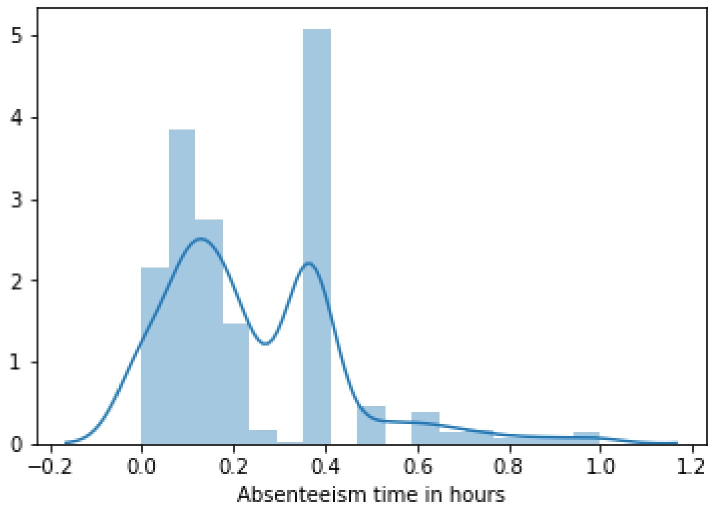
```
In [287]: #normalization
def normalize(column):
    upper = column.max()
    lower = column.min()
    y = (column - lower)/(upper-lower)
    return y
```

In [288]:
```
#Log transform our Y component
absent_time=allabsents['Absenteeism time in hours']
log_absent_time=np.log(absent_time+1)
norm_log_absent_time= normalize(log_absent_time)

sns.distplot(norm_log_absent_time)
#norm_log_absent_time.describe()
```

```
/Users/marlynehakizimana/anaconda3/lib/python3.7/site-packages/scipy/st
ats/stats.py:1713: FutureWarning: Using a non-tuple sequence for multid
imensional indexing is deprecated; use `arr[tuple(seq)]` instead of `ar
r[seq]`. In the future this will be interpreted as an array index, `arr
[np.array(seq)]`, which will result either in an error or a different r
esult.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[288]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1c12d908>
```



In [289]:
```
#OneHotEncoding for categorical features
from sklearn import preprocessing
from sklearn.preprocessing import OneHotEncoder,StandardScaler

feat=['Reason for absence','Work load Average/day ','Day of the week',
'Age', 'Social drinker', 'Weight', 'Son','Body mass index','Height','Hit
target']
#fe=coolio[['Son', 'Age','Weight', 'Body mass index','Service time','Hei
ght','Work load Average/day ','Hit target']]

encoder = OneHotEncoder(categorical_features=[0,2,4])
X_features_values = encoder.fit_transform(allabsents[feat]).toarray()
```

Model Evaluations

After looking at the multiple ways of how Absenteeism time is influenced by features like Absenteeism time in hours and Work load average per day, I wanted to test and see how supervised models ,both regression and Classification, could pick up these tendencies and find the most prominent features.

For Regression models, I explored Lasso Regression, Ridge Regressions, Linear Regression, Decision Trees and Random Forest.RMSE and R^2 are my evaluation metrics in order to see model performance.

Splitting Data

In [290]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,classification_report,r2_
score,roc_curve
from math import sqrt
from sklearn.tree import DecisionTreeRegressor,DecisionTreeClassifier
from sklearn.linear_model import LinearRegression,LogisticRegression
from sklearn.ensemble import RandomForestClassifier,RandomForestRegresso
r
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
```

In [291]:
```
X=StandardScaler().fit_transform(X_features_values)
Y=norm_log_absent_time

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,
random_state=324)
print (X_train.shape, Y_train.shape)
print (X_test.shape, Y_test.shape)
```

```
(487, 41) (487,)
(209, 41) (209,)
```

Cross Validation

In order to optimize our results, I used GridSearchCV, a method where parametersare optimized by cross-validation.

Linear,Lasso,Ridge,Random Forest Regression

In [292]:
```
#CROSS VALIDATION WITH LASSO REGRESSION
from sklearn.linear_model import Lasso, Ridge
from sklearn.model_selection import GridSearchCV
lasso = Lasso()
parameters={'alpha':[1e-4,1e-3,1e-2,1,5]}
lasso_regressor=GridSearchCV(lasso,parameters,scoring='neg_mean_squared_
error',cv=324)
lasso_regressor.fit(X,Y)
print(lasso_regressor.best_params_)
#print(lasso_regressor.best_score_)
```

```
/Users/marlynehakizimana/anaconda3/lib/python3.7/site-packages/sklearn/
linear_model/coordinate_descent.py:491: ConvergenceWarning: Objective d
id not converge. You might want to increase the number of iterations. F
itting data with very small alpha may cause precision problems.
  ConvergenceWarning)
/Users/marlynehakizimana/anaconda3/lib/python3.7/site-packages/sklearn/
linear_model/coordinate_descent.py:491: ConvergenceWarning: Objective d
id not converge. You might want to increase the number of iterations. F
itting data with very small alpha may cause precision problems.
  ConvergenceWarning)

{'alpha': 0.001}
```

In [249]:
```
#CROSS VALIDATION WITH RIDGE REGRESSION
ridge = Ridge()
parameters={'alpha':[1e-8,1e-4,1e-3,1e-2,1,5,10,20,30]}
ridge_regressor=GridSearchCV(ridge,parameters,scoring='neg_mean_squared_
error',cv=5)
ridge_regressor.fit(X,Y)
print(ridge_regressor.best_params_)
print(ridge_regressor.best_score_)
```

```
{'alpha': 30}
-0.024701747983802404
```

In [250]:
```
#CROSSVALIDATION WITH RANDOM FOREST
random=RandomForestRegressor()
estimators={'n_estimators':[100,150,300,500,800]}
random_regressor=GridSearchCV(random,estimators,scoring='neg_mean_square
d_error',cv=5)
random_regressor.fit(X,Y)
print(random_regressor.best_params_)
print(random_regressor.best_score_)
```

```
{'n_estimators': 800}
-0.022912241684939362
```

In [28]:
```
#CROSSVALIDATION WITH DECISION TREE
decision=DecisionTreeRegressor()
estimators={'max_depth':[2,3,5,10,100]}
decision_regressor=GridSearchCV(decision,estimators,scoring='neg_mean_sq
uared_error',cv=5)
decision_regressor.fit(X,Y)
print(decision_regressor.best_params_)
print(decision_regressor.best_score_)
```

```
{'max_depth': 5}
-0.01789593617429371
```

Models for Regression

After that, I created a function including all models while giving R^2 and RMSE values for each. As seen below, Random forest has the best score. It is interesting to see that Lasso and Ridge Regression have a similar score. Unfortunately, all scores do not tell us much about our data.

In [293]:
```
#Putting all models together
def modeleval(xtrain,ytrain,x,y):
    #Ridge
    u=Ridge(alpha=30).fit(xtrain,ytrain)
    y_ridge=u.predict(x)
    RMSE_ridge=sqrt(mean_squared_error(y_true=y,y_pred=y_ridge))
    #lasso
    u=Lasso(alpha=0.001).fit(xtrain,ytrain)
    y_lasso=u.predict(x)
    RMSE_lasso=sqrt(mean_squared_error(y_true=y,y_pred=y_lasso))
    #linear
    lm = LinearRegression().fit(xtrain, ytrain)
    Y_linear = lm.predict(x)
    RMSE_linear=sqrt(mean_squared_error(y_true=y,y_pred=Y_linear))
    mse=mean_squared_error(y_true=y,y_pred=Y_linear)
    #decision tree
    regressor=DecisionTreeRegressor(max_depth=5).fit(xtrain, ytrain)
    Y_regre=regressor.predict(x)
    RMSE_regre=sqrt(mean_squared_error(y_true=y,y_pred=Y_regre))

    #RandomForest
    random=RandomForestRegressor(n_estimators=800,random_state=42).fit(x
train, ytrain)
    Y_random=random.predict(x)
    #RMSE random
    RMSE_random=sqrt(mean_squared_error(y_true=y,y_pred=Y_random))
    print('Ridge Regression score:',u.score(x,y),', RMSE:',RMSE_ridge)
    print('Lasso Regression score:',u.score(x,y),', RMSE:',RMSE_lasso)
    print('Linear Regression score:',lm.score(x,y),',  RMSE:',RMSE_linea
r)
    print('Decision tree score:',regressor.score(x,y),',  RMSE:',RMSE_re
gre)
    print('Random Forest Regressor score:',random.score(x, y),'RMSE:',RM
SE_random)
    #print('Logistic Regression score:',',  RMSE:',RMSE_logi)
```

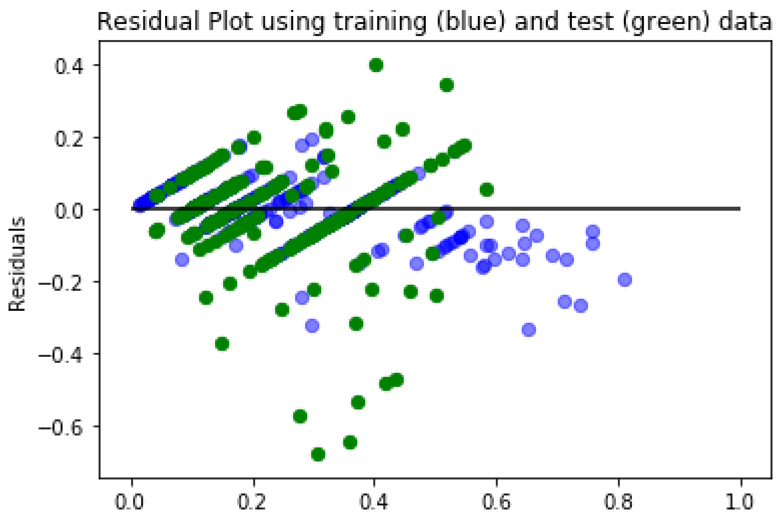
In [294]:
```
modeleval(X_train,Y_train,X_test,Y_test)
```

```
Ridge Regression score: 0.42818485185302224 , RMSE: 0.14828891852048146
Lasso Regression score: 0.42818485185302224 , RMSE: 0.14773396815320505
Linear Regression score: -1.2733441437365961e+26 ,  RMSE: 220457781493
5.491
Decision tree score: 0.3627625566321938 ,  RMSE: 0.15595639163395886
Random Forest Regressor score: 0.44741858785503097 RMSE: 0.145228105172
18124
```

Looking at our residuals and scatter points, we can see that a classifiction model is probably going to perform better. In fact, we can see that our X values on the scatter plot displays more specific values than continuous ones.

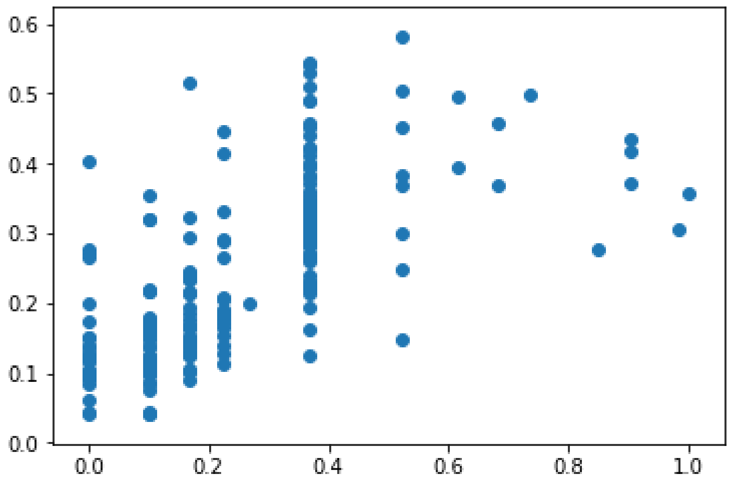
In [295]:
```
#residual
lm1 =RandomForestRegressor(n_estimators=100,random_state=0).fit(X_train,
Y_train)
plt.scatter(lm1.predict(X_train),lm1.predict(X_train)-Y_train,c='b',s=40
,alpha=0.5)
plt.scatter(lm1.predict(X_test),lm1.predict(X_test)-Y_test,c='g',s=40)
plt.hlines(y=0,xmin=0,xmax=1)
plt.title('Residual Plot using training (blue) and test (green) data')
plt.ylabel('Residuals')
```

Out[295]:
```
Text(0,0.5,'Residuals')
```



In [296]:
```
plt.scatter(Y_test,lm1.predict(X_test))
```

Out[296]:
```
<matplotlib.collections.PathCollection at 0x1a1cabaeb8>
```



Classification

For Classification models, I went ahead and categorized my **Absenteeism time in hours** column in two classes.

- h<8: for all employees who missed less than 8hrs of work
- h>=8: for all employees who missed more than 8hrs of work. I decided to combine 8 hours or more since h>8 accounted for less than 10% of the data.

Note: A few variations made to the models:

- Categorized Reason for absence into 2 classes, where the first one is ICD reasons and the second one were non ICD ones
- Categorized Reason for absence into putting similar categories together such as respiratory and digestive.

However, both cases decreased the performance of the model instead of increasing it.

In [554]:
```
aba=[]
for i in allabsents['Absenteeism time in hours']:
    if i<8:
        aba.append(0)
    else:
        aba.append(1)
allabsents['newabsent']=aba
```

pd.get_dummies is used for creating dummy variable for categorical variables. Similar approach to OneHotEncoder.

In [582]:
```
de=pd.get_dummies(allabsents['Reason for absence'],prefix='Reason')
da=pd.get_dummies(allabsents['Month of absence'],prefix='Month')
pp=pd.get_dummies(allabsents['Social drinker'],prefix='soc')
pj=pd.get_dummies(allabsents['Social smoker'],prefix='smo')
pa=pd.get_dummies(allabsents['Day of the week'],prefix='day')
px=pd.get_dummies(allabsents['Seasons'],prefix='smo')
rr=pd.get_dummies(allabsents['Education'],prefix='edu')

fe=allabsents[['Son','Age','Weight', 'Body mass index','Distance from Re
sidence to Work','Service time','Height','Work load Average/day ','Hit t
arget']]

x_features=pd.concat([fe,de,da,pp,pj,pa,rr],axis=1)
```

In [583]:
```
XX=x_features
YY=allabsents['newabsent']

XX_train, XX_test, YY_train, YY_test = train_test_split(XX, YY, test_siz
e=0.3,random_state=324)
print (XX_train.shape, YY_train.shape)
print (XX_test.shape, YY_test.shape)
```

```
(487, 61) (487,)
(209, 61) (209,)
```

Models for Classification

Decision trees, RandomForest and Logistic Regression with RMSE, R^2 and AUC score as evaluation metrics.

In [584]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score, precision_score,f1_score,roc_a
uc_score, roc_curve
```

In [585]:
```
#Putting all models together
def modeleval(xtrain,ytrain,x,y):
    #decision tree
    regressor=DecisionTreeRegressor(max_depth=3).fit(xtrain, ytrain)
    Y_regre=regressor.predict(x)
    RMSE_regre=sqrt(mean_squared_error(y_true=y,y_pred=Y_regre))
    #RandomForest
    ra=RandomForestClassifier(n_jobs=-1,n_estimators=300,random_state=42
)
    random=ra.fit(xtrain, ytrain)
    Y_random=random.predict(x)
    y_hat = random.predict_proba(x)[::,1]
    fpr, tpr,_ = roc_curve(y,  y_hat)
    auc = roc_auc_score(y, y_hat)
    #RMSE random
    RMSE_random=sqrt(mean_squared_error(y_true=y,y_pred=Y_random))
    #LogisticRegression
    logi=LogisticRegression(random_state=0,solver='lbfgs',multi_class='m
ultinomial').fit(xtrain, ytrain)
    Y_logi=logi.predict(x)
    y_hat1 = logi.predict_proba(x)[::,1]
    fpr, tpr,_ = roc_curve(y,  y_hat1)
    auc1 = roc_auc_score(y, y_hat1)
    #Logi RMSE
    RMSE_logi=sqrt(mean_squared_error(y_true=y,y_pred=Y_logi))
    print('Decision tree score:',regressor.score(x,y),',  RMSE:',RMSE_re
gre)
    print('RanForest accuracy score:',random.score(x, y),',AUC score:',a
uc,' RMSE:',RMSE_random)
    print('LogiReg accuracy score:',logi.score(x,y),', AUC score:',auc1,
'  RMSE:',RMSE_logi)
```

```
In [586]: modeleval(XX_train,YY_train,XX_test,YY_test)
```

```
Decision tree score: 0.5188414477973493 ,  RMSE: 0.3386932997803416
RanForest accuracy score: 0.8325358851674641 ,AUC score: 0.934367198002
6887  RMSE: 0.4092237955355674
LogiReg accuracy score: 0.6076555023923444 , AUC score: 0.4631265603994
623   RMSE: 0.6263740875927544
```

I decided to pick Random Forest as my main model since it has better scores. Let us have a look into which features have the most importance.

In [587]:
```
random=RandomForestClassifier(n_estimators=300,random_state=42,n_jobs=-1
)
fity=random.fit(XX_train, YY_train)
Y_random=random.predict(XX_test)
y_hat = random.predict_proba(XX_test)[::,1]
pd.DataFrame(random.feature_importances_,
                                   index = XX.columns,columns=['importan
ce']).sort_values('importance', ascending=False)
```

Out[587]:

	importance
Work load Average/day	0.074908
Reason_23	0.071364
Reason_28	0.057815
Hit target	0.053479
Service time	0.044107
Age	0.041386
Distance from Residence to Work	0.040247
Weight	0.039577
Body mass index	0.038135
Height	0.035834
Reason_27	0.031185
Son	0.030902
Reason_19	0.030147
day_2	0.025129
Reason_10	0.020205
Reason_22	0.018848
Reason_13	0.018719
day_4	0.018392
day_3	0.017337
day_5	0.015849
Reason_25	0.015405
Reason_18	0.015057
Month_3	0.014724
Reason_26	0.013439
day_6	0.012994
Month_7	0.012675
soc_0	0.012353
Reason_1	0.010780
Month_4	0.010737
soc_1	0.010427
...	...
Month_8	0.009786
Month_10	0.009243
Month_2	0.009047

	importance
Month_1	0.008718
Month_6	0.008342
edu_1	0.008156
edu_2	0.008040
Month_11	0.007899
Reason_14	0.007214
Month_12	0.006546
Month_5	0.006313
Reason_7	0.006232
edu_3	0.005993
smo_1	0.004900
smo_0	0.004619
Reason_6	0.004062
Reason_12	0.003702
Reason_8	0.003404
Month_9	0.003225
Reason_5	0.002336
Reason_9	0.002101
Reason_16	0.001725
Reason_21	0.001598
Reason_24	0.001401
edu_4	0.001170
Reason_4	0.000908
Reason_2	0.000612
Reason_17	0.000398
Reason_15	0.000339
Reason_3	0.000000

61 rows × 1 columns

With this new information, I ran my Random Forest model one more time with only the most importantant features. A better score was obtained.

In [581]:
```
fp=['Reason_28','Reason_23','Age','Service time', 'Hit target','Height',
'Work load Average/day ','Weight','Son','Reason_27','day_2','Body mass i
ndex','Reason_19','Reason_22','Reason_10','Distance from Residence to Wo
rk','Reason_13','day_4','day_3','day_5','Reason_25']
XX=x_features[fp]
YY=allabsents['newabsent']

XX_train, XX_test, YY_train, YY_test = train_test_split(XX, YY, test_siz
e=0.3,random_state=42)

random=RandomForestClassifier(n_estimators=300,random_state=42,n_jobs=-1
)
fity=random.fit(XX_train, YY_train)
Y_random=random.predict(XX_test)
y_hat = random.predict_proba(XX_test)[::,1]
auc = roc_auc_score(YY_test, y_hat)
print('R^2:',random.score(XX_test,YY_test))
print('AUC score:',auc)
```

```
R^2: 0.861244019138756
AUC score: 0.9362380952380953
```

Conclusion

In both regression and classification models, **Random Forest** is the best model for this problem. Even if there is no easy way to see how a Randowm Forest works, due to its use of a lot of decision trees, it is able to calculate the best path for us while still accounting outliers or variance We can finally answer questions asked from the beginning:

Which areas of life affect Abseenteeism(i.e: Work or Family..)?

Work related reasons such as Work load and Hit target are the main reasons.

Is there an obvious relationship between reason for absence and absenteeism?

The nature of the reason for absence is also a key indication on the number of absenteeism time in hours.

Between regression models and Classification models, is there a better model for our problem?

From our analysis, Classification models performed better than regression models.

Even though more people are taking less than a day of absenteeism, when numbers add up, the company still ends up loosing revenue.

Here are a few suggestions that could help reduce absenteeism:

- **Flexible schedule.** Since medical consultations are high, giving an option for a flexible schedule where the few hours lost can be compensated by either coming in early or leaving late, depending on the employee's role.
- **Employee Wellness program.** For example, the amount of Work load can lead to absenteeism due to stress. Relaxing activities during lunchbreaks targeting specific muscles for a delivery employee not only reduce stress, potential health problems but also increase **employee morale**.

```
In [ ]:
```