

Intelligent Data Analysis Assignment 1

Tasks

There are two main tasks in there assignment:

- (1) Use **"Reason for absence"** as the label, adopt a machine learning model to predict the label for a given instance.
- (2) Use **"Absenteeism time in hours"** as the target, adopt a machine learning model to predict the target of a given instance.

So there are two main tasks, regression and classification. The assignment contains 3 main steps: analysis the data, pre-process the data and classification/regression.

The next three parts will be each step of this assignment.

Analysis the data

(1) Reading data

The first step is reading data from data files. Here I use api from **pandas** library, as it show below:

```
train_path = "train.csv"
test_path = "val.csv"
```

After reading, the raw is gotten. Using api from pandas library to get more details of the data. Information is like: data types, total numbers, if existing null data, total space and so on.

```
RangeIndex: 540 entries, 0 to 539
Data columns (total 21 columns):
ID                    540 non-null int64
Reason for absence    540 non-null int64
Month of absence      540 non-null int64
Day of the week       540 non-null int64
Seasons              540 non-null int64
Transportation expense 540 non-null int64
Distance from Residence to Work 540 non-null int64
Service time         540 non-null int64
Age                  540 non-null int64
Work load Average/day 540 non-null float64
Hit target           540 non-null int64
Disciplinary failure  540 non-null int64
Education            540 non-null int64
Son                  540 non-null int64
Social drinker       540 non-null int64
Social smoker        540 non-null int64
Pet                  540 non-null int64
Weight               540 non-null int64
Height               540 non-null int64
Body mass index       540 non-null int64
Absenteeism time in hours 540 non-null int64
```

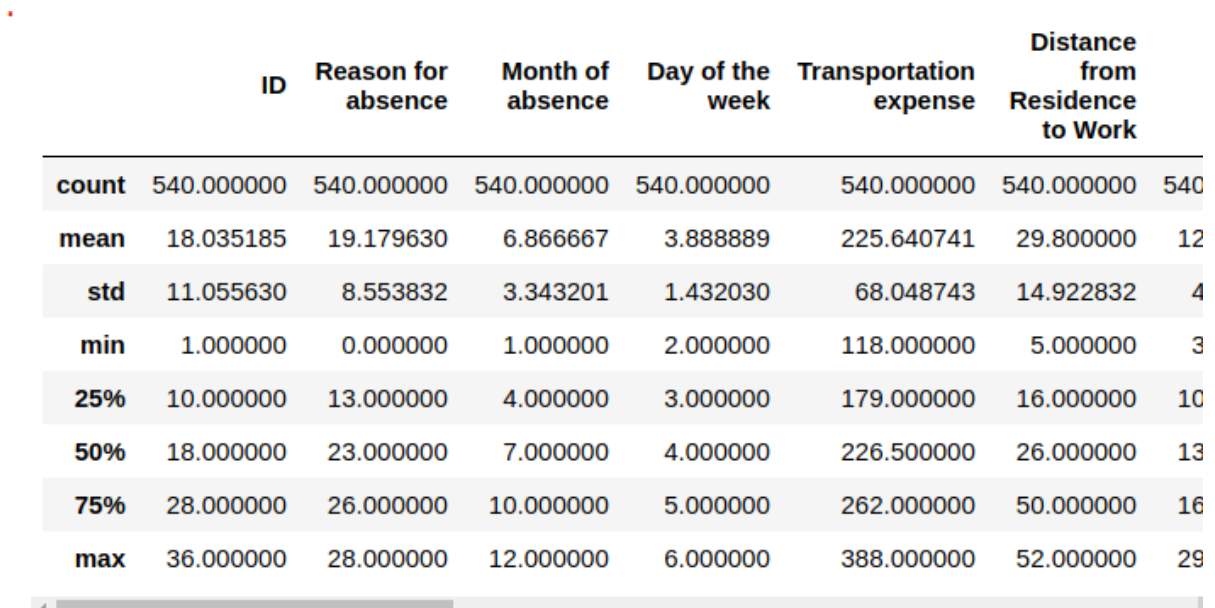
Figure 1: Data information

(2) Analyzing

Combining with the data file, It can be determined that some data belongs to a continuous type, some data belong to a discrete type, and the rests are boolean type. The different types of data determine how the data

is processed later, data with a discrete type may be transformed with the method one-hot encoding for instance.

At the same time, with the help of original data file, we could have a rough understanding of the data. Using function `describe()` we can get the **statistical characteristics of the data**, like mean/average, and then make more analysis.



| | ID | Reason for absence | Month of absence | Day of the week | Transportation expense | Distance from Residence to Work | |
|--------------|------------|--------------------|------------------|-----------------|------------------------|---------------------------------|-----|
| count | 540.000000 | 540.000000 | 540.000000 | 540.000000 | 540.000000 | 540.000000 | 540 |
| mean | 18.035185 | 19.179630 | 6.866667 | 3.888889 | 225.640741 | 29.800000 | 12 |
| std | 11.055630 | 8.553832 | 3.343201 | 1.432030 | 68.048743 | 14.922832 | 4 |
| min | 1.000000 | 0.000000 | 1.000000 | 2.000000 | 118.000000 | 5.000000 | 3 |
| 25% | 10.000000 | 13.000000 | 4.000000 | 3.000000 | 179.000000 | 16.000000 | 10 |
| 50% | 18.000000 | 23.000000 | 7.000000 | 4.000000 | 226.500000 | 26.000000 | 13 |
| 75% | 28.000000 | 26.000000 | 10.000000 | 5.000000 | 262.000000 | 50.000000 | 16 |
| max | 36.000000 | 28.000000 | 12.000000 | 6.000000 | 388.000000 | 52.000000 | 29 |

Figure 2: Data statistical information

For example, as we can see, "ID" is not primary, that means some students have been absence more than one time, so "ID" may effect the regression or classification. "Disciplinary failure", "Social drinker", "Social smoker" is the data with boolean type, so the minimal is 0 while the maximal is 1.0. 20%/50%/75% tells us a rough distribution of data, with other methods like drawing scatter plot could help us understand the distribution of data more intuitively.

Using api to calculate the correlation coefficient of each pair of data, and draw heatmap matrix, as follows:

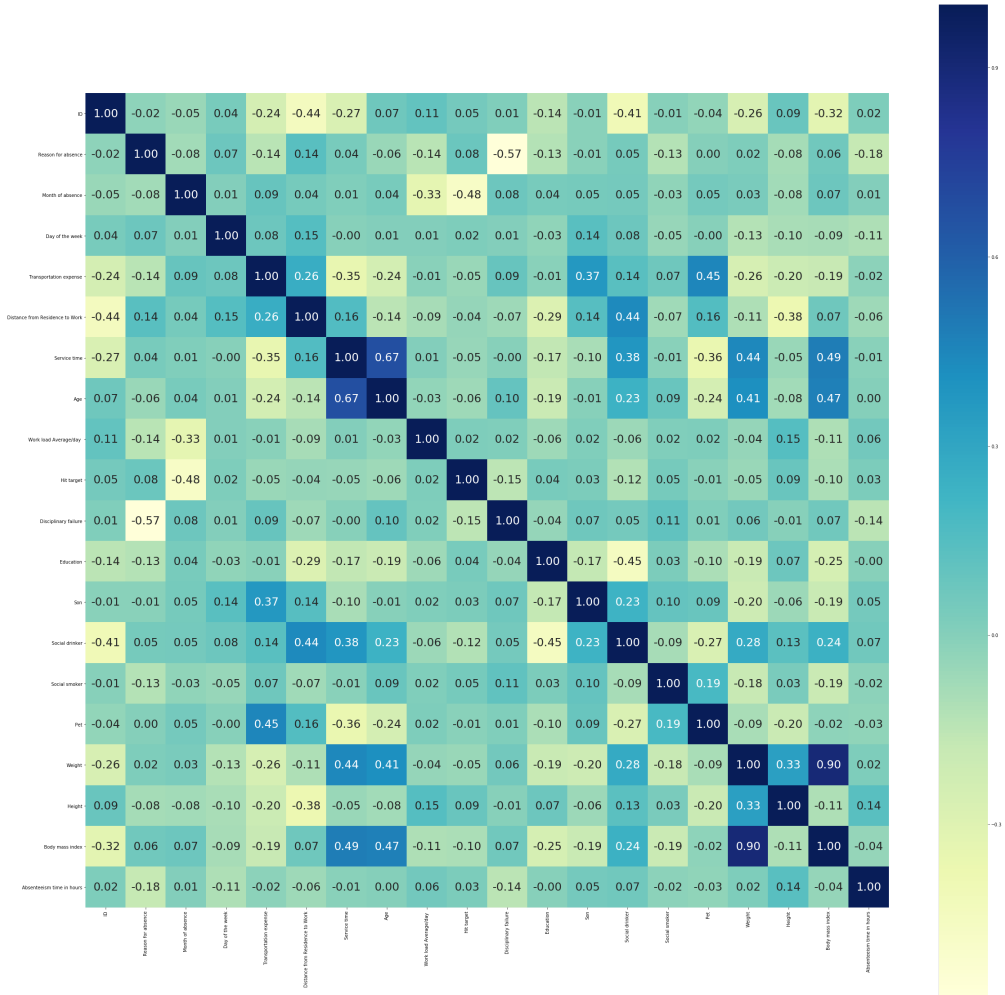


Figure 3: Correlation matrix

It's obviously illustrated that: There is strong positive correlation between "Weight" and "Body mass index". So here using one of them is enough. Also, "Age" and "Service time" are similar. Using correlation coefficient also helps us find the relation between target label and other attributes.

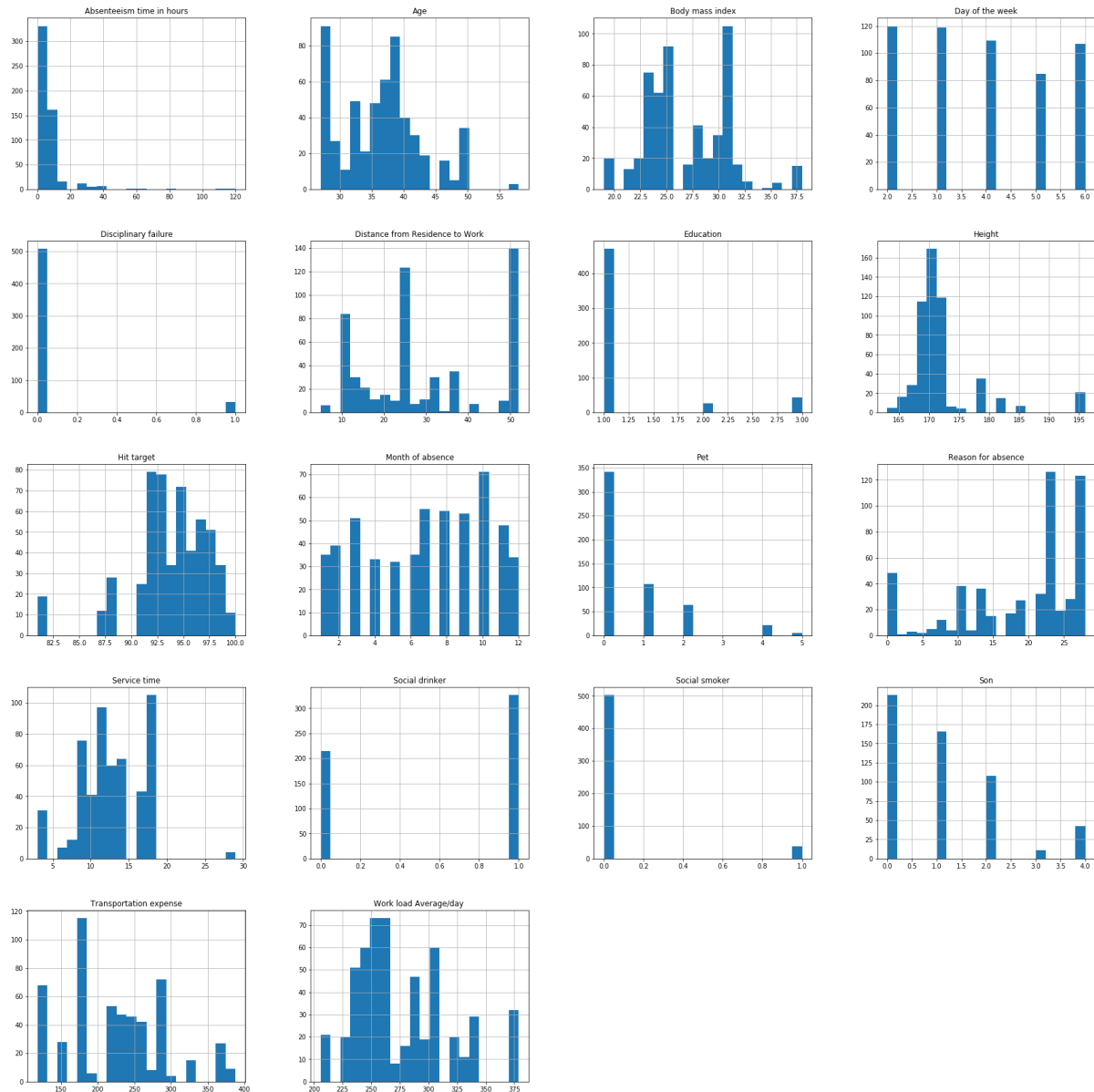


Figure 4: Histogram

A normal distribution is an arrangement of a data set in which most values cluster in the middle of the range and the rest taper off symmetrically toward either extreme, which is the best distribution for the machine learning methods. Using data distribution histogram to check if data is biased or "skewed".

And we could find attributes "Age", "Absenteeism time in hours", "Height", "Hit target", "Reason for absence" are more or less bias. It is necessary to solve the bias of these attributes.

Moreover, some data are quite different from other data. Such data may be the noise while sampling, they may effect the training result. So they should also be processed.

Moreover, some data are quite different from other data. Such data may be the noise while sampling, they may effect the training result. So they should also be processed.

After analyzing, the coming step is pre-processing.

Pre-processing

In previous steps, data with different types have been splited, here some methods will be used to deal with each type of data.

(1) Biased data

I deal with biased data first. Using log function to transform distributions. However, checking if the data is "skewed" enough before transforming.

```

for s in featureCon:
    if(abs(train_raw_data[s].skew()) > skew_line):
        train_raw_data[s] = train_raw_data[s].apply(lambda x : np.log(x +
1))
        test_raw_data[s] = test_raw_data[s].apply(lambda x : np.log(x + 1))

```

(2) Abnormal data

In these assignment, I calculate quartile(四分位数) for each attributes, Q1 splits off the lowest 25% of data from the highest 75%, Q2 is the median, Q3 splits off the highest 25% of data from the lowest 75%. Defining IQR as the distance between Q1 and Q3, setting the upper bound $Q3 + a * IQR$ and the lower bound $Q1 - a * IQR$, a is the parameter. For attributes, delete those data which Outside the upper and lower bounds. The code is as follows:

```

index = set(train_raw_data.index)
for s in featureAbn:
    percentile = np.percentile(train_raw_data[s], [0, 25, 50, 75, 100])
    IQR = percentile[3] - percentile[1]
    upLimit = percentile[3] + IQR * IQR_limit
    downLimit = percentile[1] - IQR * IQR_limit
    for idx in index:
        if(train_raw_data.loc[idx][s] > upLimit or train_raw_data.loc[idx]
[s] < downLimit):
            train_raw_data.drop([idx], inplace=True)
            index = index - set([idx])

```

(3) Decide which attribute to use

Calculating correlation coefficient again and selecting those attributes that are sufficiently large for the coefficient value associated with the target attribute. However, scikit-learn also provides apis to achieve such job, like **GenericUnivariateSelect** to select attributes conveniently. Here is the code of attribute selecting:

```

using_attrs = []
for attr in rest_attrs:
    if(abs(corr_matrix["Absenteeism time in hours"][attr]) > corr_line):
        using_attrs.append(attr)

```

(4) Standardization and Splitting

The last step of data pre-processing. For continues data, using StandardScaler to standardize. For discrete data, using one-hot encoding.

```

if(X_vec_con.shape[1] > 0):
    scaler=preprocessing.StandardScaler().fit(X_vec_con)
    X_vec_con_ed=scaler.transform(X_vec_con)
else:
    X_vec_con_ed = X_vec_con

```

```
if(X_vec_cat.shape[1] > 0):
    enc=preprocessing.OneHotEncoder(categories='auto')
    enc.fit(X_vec_cat)
    X_vec_cat_ed=enc.transform(X_vec_cat).toarray()
else:
    X_vec_cat_ed = X_vec_cat
```

Combine each part of data, then split them into training set and testing set.

```
x_train = X_vec[0 : n_train]
y_train = Y_vec[0 : n_train]
x_test = X_vec[n_train : n_test+n_train]
y_test = Y_vec[n_train : n_test+n_train]
```

Machine Learning Models

As for model, cause calculating the linear correlation before, here I choose linear **regression** and **SVR** for regression and SVC for classification. Linear regression is easy to achieve but may not suitable to complicate situation, SVR or SVC is more flexible cause they could use different kernel. However here I just use the linear kernel. With the help of sklearn, it's convenient to achieve machine learning methods.

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
linreg = LinearRegression().fit(x_train, y_train)
y_pre = linreg.predict(x_test)
mse = mean_squared_error(y_test, y_pre)
rmse = np.sqrt(mse)
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
linreg = LinearRegression().fit(x_train, y_train)
y_pre = linreg.predict(x_test)
mse = mean_squared_error(y_test, y_pre)
rmse = np.sqrt(mse)
```

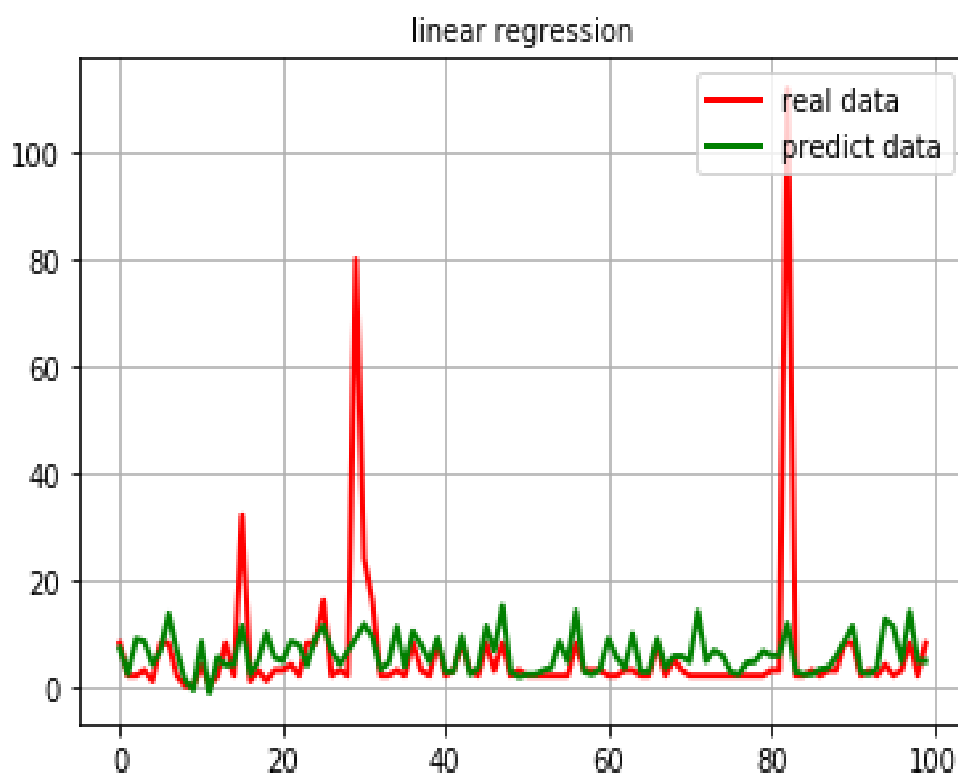


Figure 5: Regression

Results

Maybe linear model is not fit to such data set. The mes and the accuracy of classification is not well enough.

```
MSE = 165.75696334262597
RMSE = 12.87466362056213
R2 = 0.11612210098398501
R2 = 0.1312364209040322
```

Figure 6: Result of regression

```
Accuracy = 0.41
Recall = 0.41
F1 = 0.41
```

Figure 7: Result of classification

Note: For some reason I am not use Anaconda to manage python packages, just using pip.

How to use:

```
python exp.py train_path test_path
```