생체 신호에 따른 흡연 여부 예측과 Decision Tree와 Random Forest 성능 비교

인공지능 - 컴퓨터공학전공 3학년 신현규

Index

Motivation

현재 청소년 흡연 문제 심각

청소년기에 시작된 흡연은
니코틴 중독을 강화

정신적 육체적 피해가 더 높아
금연 실천이 절실히 요구

Related Research

Perbandingan Algoritma Klasifikasi Naïve Bayes, C4.5 Dan Knn Untuk Menentukan Perokok Aktif Dan Perokok Pasif

Muhammad, Isra Muntaha Tanjung (2023) *Perbandingan Algoritma Klasifikasi Naïve Bayes, C4.5 Dan Knn Untuk Menentukan Perokok Aktif Dan Perokok Pasif.* Undergraduate Thesis thesis, Institut Teknologi Telkom Purwokerto.

Text
Cover (1).pdf
Download (1MB)

Text
Abstract (1).pdf
Download (13kB)

K-Nearest Neighbor
Naive Bayes

Abstract

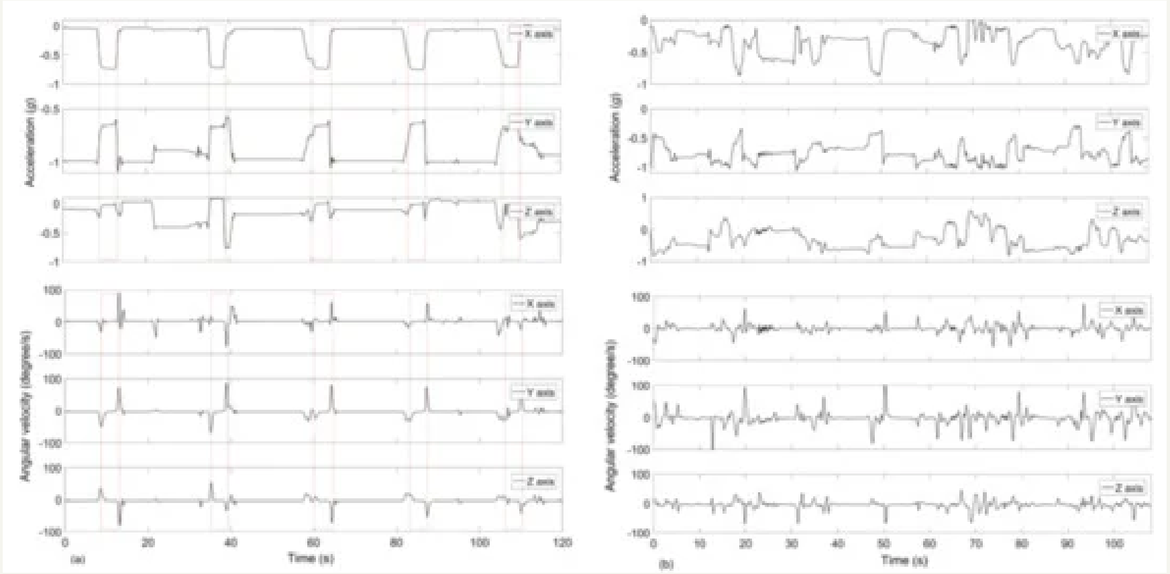
Smoking is a common habit in many countries, apart from developed countries, it has also become a habit in developing countries, especially in Indonesia. According to the smoke inhaled, smokers can be divided into active smokers and passive smokers. Active smokers can be classified based on the number of cigarettessmoked per day. Passive smoking is defined as someone who is exposed to cigarette smoke for more than 15 minutes a day for more than 1 day a week. WHO (World Health Organization), estimates that in 2025 the number of smokers in Indonesia will increase by around 45% of the total population. Data mining is able to help classify whether a person belongs to the category of passive or active smokers, with symptoms in smokers or indications for a smoker. The process of analyzing active and passive smokers is carried out by a classification process and the result isthat the person is an active or passive smoker. This study uses 3 data mining algorithms namely Decision Tree, K-Nearest Neighbor and Naive Bayes. The dataset used is the Body Signal of Smoking from Kaggle. From the results of the study successfully implemented the Decision Tree algorithm (C4.5), K-Nearest Neighbor and Naïve Bayes using the Body Signal of smoking (Kaggle) dataset in predicting active and passive smoking. From the results of this study the accuracy produced by the C4.5 algorithm is 70.78%, the K-NN algorithm is 71.76% and the Naïve Bayes algorithm is 71.19%. From a comparison of the three algorithms it was found that the K-NN algorithm is the algorithm with the highest level of accuracy so that the K-NN algorithm is suitable for use in the classification of determining active and passive smokers. Keywords: Smokers, Decision Tree, Naive Bayes, K-Nearest Neighbor.

Item Type: Thesis (Undergraduate Thesis)
Subjects: T Technology > TA Engineering (General). Civil engineering (General)
Divisions: Faculty of Informatics > Informatics Engineering
Depositing User: pustakawan ittp
Date Deposited: 30 Mar 2023 02:49
Last Modified: 30 Mar 2023 02:49
URI: http://repository.ittelkom-pwt.ac.id/id/eprint/9191

Actions (login required)

View Item

Related Research



THE ACCELEROMETER AND GYROSCOPE SIGNALS FROM A PARTICIPANT. (A) SMOKING EVENT, (B) AN EATING EVENT. DASHED LINES SHOW THE SMOKING-HMGS.

About Dataset

smoking.csv

건강검진정보

국민건강보험의 직장가입자와 40세 이상의 피부양자, 세대주인 지역가입자와 40세 이상의 지역가입자의 일반건강검진 결과와 생애전환기건강진단 수검이력이 있는 각 연도별 수진자 100만 명에 대한 기본정보(성, 연령대, 시도코드 등)와 검진내역(신장, 체중, 총콜레스테롤, 혈색소 등)으로 구성된 개방데이터

- gender : **성별**
- age : **5년 간격**
- height(cm) : **신장(키)**
- weight(kg) : **몸무게**
- waist(cm) : **허리 둘레 길이**
- eyesight(left) : **왼쪽 시력**
- eyesight(right) : **오른쪽 시력**
- hearing(left) : **왼쪽 청력**
- hearing(right) : **오른쪽 청력**
- systolic : **수축기혈압**
- relaxation : **이완기혈압**
- fasting blood sugar : **공복 혈당**
- Cholesterol : **총 콜레스테롤**
- triglyceride : **중성지방 (트리글리세리드)**
- HDL : **고밀도 클레스테롤**
- LDL : **저밀도 클레스테롤**
- hemoglobin : **적혈구 (헤모글로빈)**
- Urine protein : **단백뇨 (소변 단백질)**
- serum creatinine : **혈장 크레아티닌**
- AST : ****아스파라긴산 분해효소 (GOT)****
- ALT : ****알라닌아미노 분해효소 (GOT)****
- Gtp : **감마글루타밀전이효소 (γ-GTP)**
- oral : **구강건강상태**
- dental caries : **충치**
- tartar : **치석 상태**
- smoking : **흡연 여부 (0 or 1)**

Data Exploration

```
#Importing the basic librarires fot analysis

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use("ggplot")  #using style ggplot

%matplotlib inline
import plotly.graph_objects as go
import plotly.express as px
```

Importing the basic libraries for analysis using style ggplot

Data Exploration

```
df =pd.read_csv("../input/body-signal-of-smoking/smoking.csv")

df.head()
```

	ID	gender	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)	...	hemoglobin	Urine protein	serum creatinine	AST	ALT	Gtp	oral	dental caries	tartar	smoking
0	0	F	40	155	60	81.3	1.2	1.0	1.0	1.0	...	12.9	1.0	0.7	18.0	19.0	27.0	Y	0	Y	0
1	1	F	40	160	60	81.0	0.8	0.6	1.0	1.0	...	12.7	1.0	0.6	22.0	19.0	18.0	Y	0	Y	0
2	2	M	55	170	60	80.0	0.8	0.8	1.0	1.0	...	15.8	1.0	1.0	21.0	16.0	22.0	Y	0	N	1
3	3	M	40	165	70	88.0	1.5	1.5	1.0	1.0	...	14.7	1.0	1.0	19.0	26.0	18.0	Y	0	Y	0
4	4	F	40	155	60	86.0	1.0	1.0	1.0	1.0	...	12.5	1.0	0.6	16.0	14.0	22.0	Y	0	N	0

Importing the dataset and look the data set

Data Exploration

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55692 entries, 0 to 55691
Data columns (total 27 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   ID                   55692 non-null  int64  
 1   gender               55692 non-null  object 
 2   age                  55692 non-null  int64  
 3   height(cm)           55692 non-null  int64  
 4   weight(kg)           55692 non-null  int64  
 5   waist(cm)            55692 non-null  float64
 6   eyesight(left)       55692 non-null  float64
 7   eyesight(right)      55692 non-null  float64
 8   hearing(left)        55692 non-null  float64
 9   hearing(right)       55692 non-null  float64
 10  systolic             55692 non-null  float64
 11  relaxation           55692 non-null  float64
 12  fasting blood sugar  55692 non-null  float64
 13  Cholesterol          55692 non-null  float64
 14  triglyceride         55692 non-null  float64
 15  HDL                  55692 non-null  float64
 16  LDL                  55692 non-null  float64
 17  hemoglobin           55692 non-null  float64
 18  Urine protein        55692 non-null  float64
 19  serum creatinine     55692 non-null  float64
 20  AST                  55692 non-null  float64
 21  ALT                  55692 non-null  float64
 22  Gtp                  55692 non-null  float64
 23  oral                 55692 non-null  object 
 24  dental caries        55692 non-null  int64  
 25  tartar               55692 non-null  object 
 26  smoking              55692 non-null  int64  
dtypes: float64(18), int64(6), object(3)
memory usage: 11.5+ MB
```

No any missing value

Checking the dtypes of all the columns

Data Exploration

```
df.describe().round(2)
```

	ID	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)	systolic
count	55692.00	55692.00	55692.00	55692.00	55692.00	55692.00	55692.00	55692.00	55692.00	55692.00
mean	27845.50	44.18	164.65	65.86	82.05	1.01	1.01	1.03	1.03	121.49
std	16077.04	12.07	9.19	12.82	9.27	0.49	0.49	0.16	0.16	13.68
min	0.00	20.00	130.00	30.00	51.00	0.10	0.10	1.00	1.00	71.00
25%	13922.75	40.00	160.00	55.00	76.00	0.80	0.80	1.00	1.00	112.00
50%	27845.50	40.00	165.00	65.00	82.00	1.00	1.00	1.00	1.00	120.00
75%	41768.25	55.00	170.00	75.00	88.00	1.20	1.20	1.00	1.00	130.00
max	55691.00	85.00	190.00	135.00	129.00	9.90	9.90	2.00	2.00	240.00

look describe data set

Data Exploration

HDL	LDL	hemoglobin	Urine protein	serum creatinine	AST	ALT	Gtp	dental caries	smoking
55692.00	55692.00	55692.00	55692.00	55692.00	55692.00	55692.00	55692.00	55692.00	55692.00
57.29	114.96	14.62	1.09	0.89	26.18	27.04	39.95	0.21	0.37
14.74	40.93	1.56	0.40	0.22	19.36	30.95	50.29	0.41	0.48
4.00	1.00	4.90	1.00	0.10	6.00	1.00	1.00	0.00	0.00
47.00	92.00	13.60	1.00	0.80	19.00	15.00	17.00	0.00	0.00
55.00	113.00	14.80	1.00	0.90	23.00	21.00	25.00	0.00	0.00
66.00	136.00	15.80	1.00	1.00	28.00	31.00	43.00	0.00	1.00
618.00	1860.00	21.10	6.00	11.60	1311.00	2914.00	999.00	1.00	1.00

look describe data set

Data Exploration

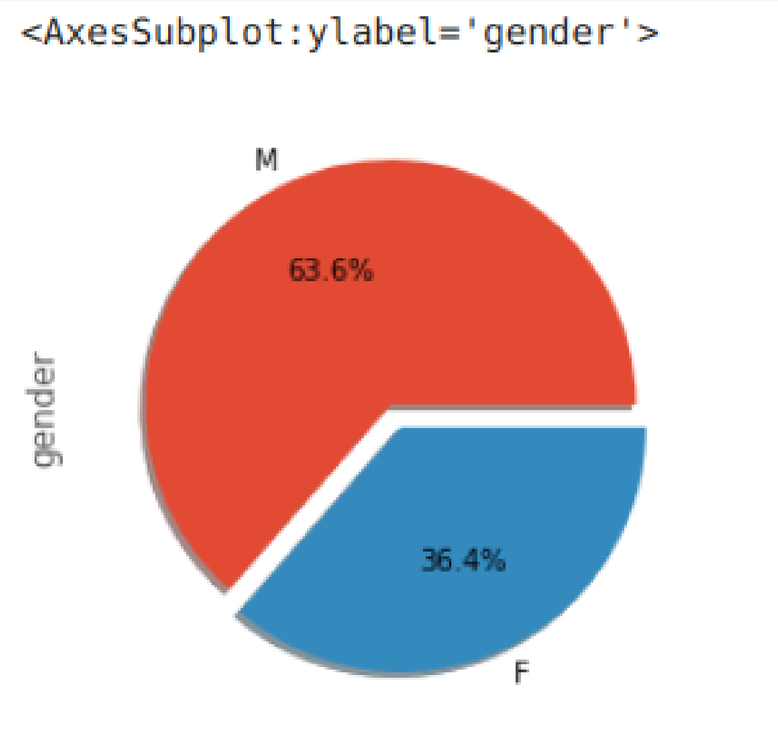
```
df.nunique().sort_values()
```

```
oral                      1
smoking                   2
gender                    2
dental caries             2
hearing(left)             2
hearing(right)            2
tartar                    2
Urine protein             6
height(cm)               13
age                      14
eyesight(right)          17
eyesight(left)           19
weight(kg)               22
serum creatinine         38
relaxation               95
HDL                     126
systolic                130
hemoglobin              145
AST                     219
ALT                     245
fasting blood sugar     276
Cholesterol             286
LDL                     289
triglyceride            390
Gtp                     488
waist(cm)               566
ID                    55692
dtype: int64
```

check unique value

Visualisations

```
df['gender'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',shadow=True)
```



The percentage gender in the dataset:

Female = 36.4%

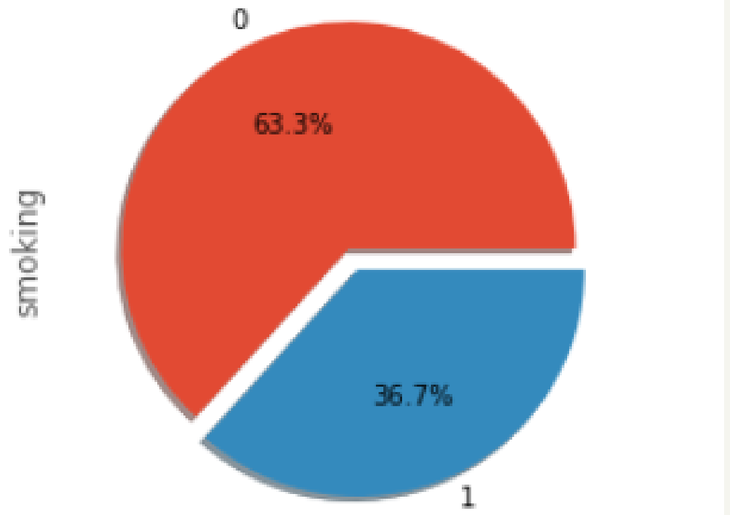
Male = 63.6 %

how much percentage Gender in the dataset

Visualisations

```
df['smoking'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',shadow=True)
```

```
<AxesSubplot:ylabel='smoking'>
```



The percentage gender in the dataset:

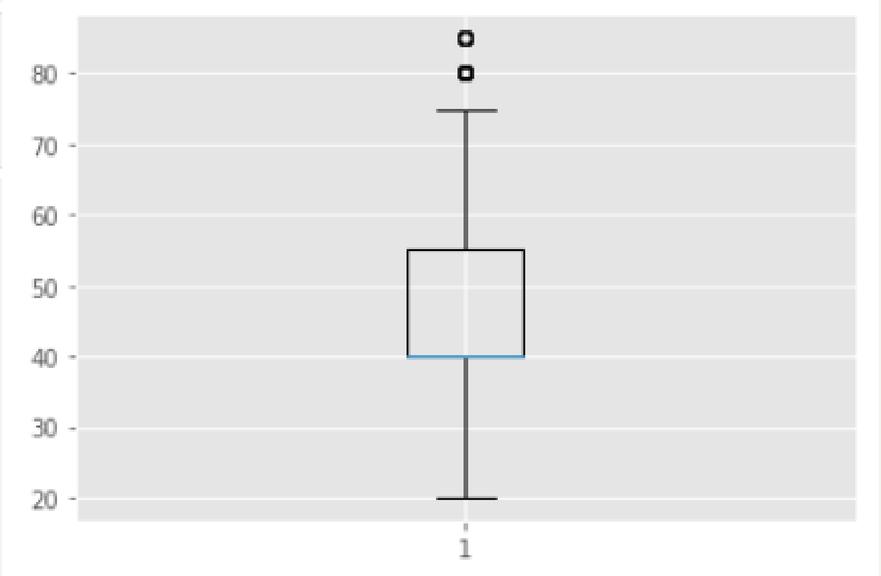
Non-smoking = 63.3%

Smoking = 36.7 %

how much percentage smoking in the dataset

Visualisations

```
plt.boxplot(df["age"])
plt.show()
```



Describe age :

mean = 44

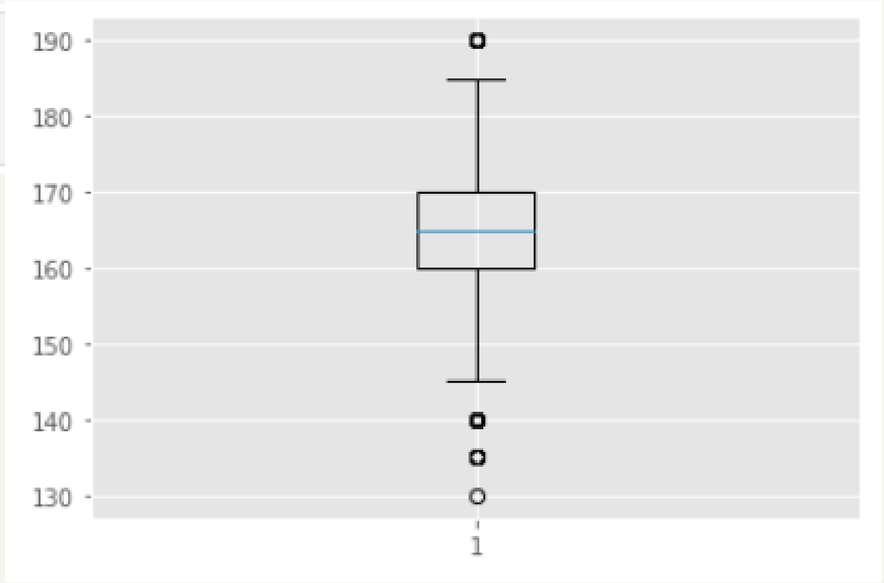
min = 20

max = 85

boxplot for show describe age

Visualisations

```
plt.boxplot(df["height(cm)"])
plt.show()
```



Describe height- cm :

mean= 164

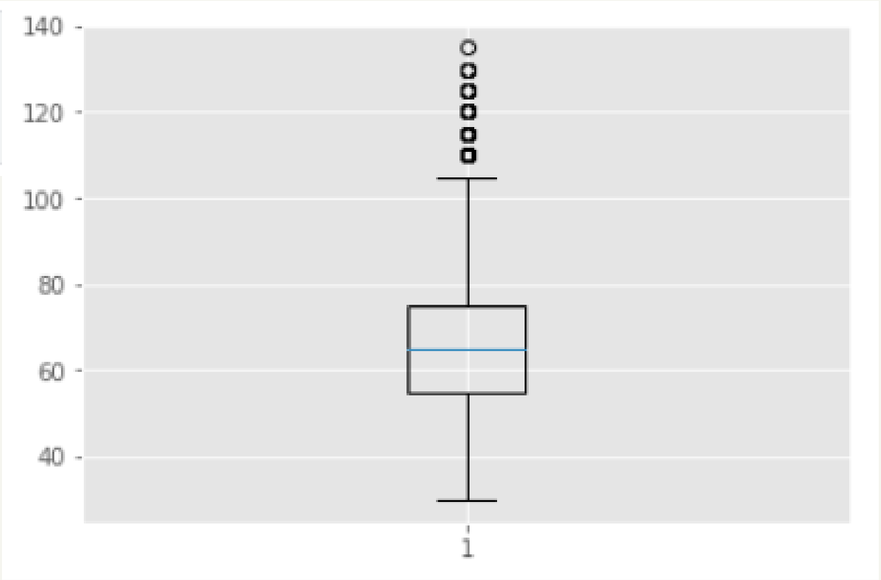
min = 130

max = 190

boxplot for show describe height

Visualisations

```
plt.boxplot(df["weight(kg)"])
plt.show()
```



Describe weight(kg) :

mean= 65

min = 30

max = 135

boxplot for show describe weight

Visualisations

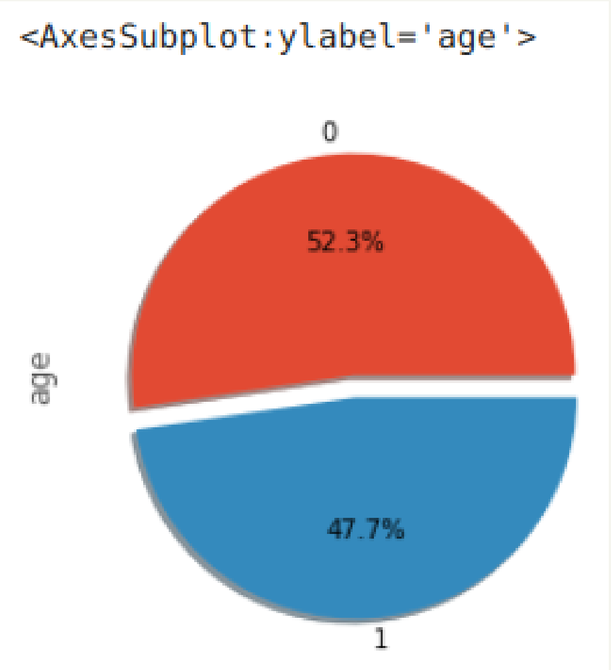
```
ag=df.groupby("smoking")["age"].mean()
ag
```

```
smoking
0    45.677981
1    41.607431
Name: age, dtype: float64
```

make groupby to show the average age smoking

Visualisations

```
ag.plot(kind="pie",explode=[0,0.1],autopct='%1.1f%%',shadow=True)
```



The average age smoking

age 45 = non- somkign

age 41 = somking

graph average age smoking

Visualisations

```
summary=df.groupby(["gender","smoking"])["age","weight(kg)","height(cm)"].mean().round(0)
summary
```

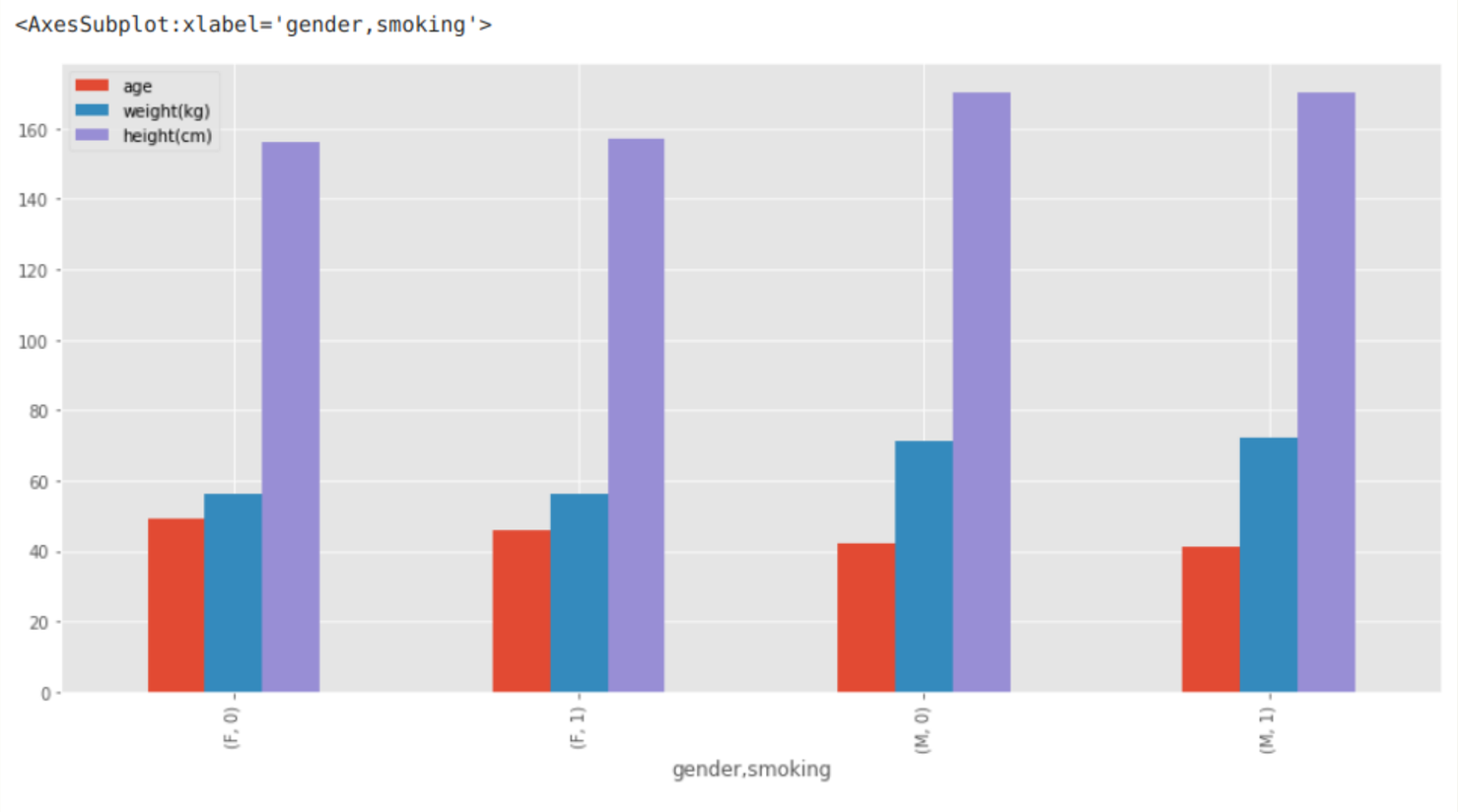
gender	smoking	age	weight(kg)	height(cm)
F	0	49.0	56.0	156.0
	1	46.0	56.0	157.0
M	0	42.0	71.0	170.0
	1	41.0	72.0	170.0

group by for show the average age , weight and height by the gender

Visualisations

```
summary.plot(kind="bar",figsize=(15,7))
```

graph the group by



Analysis Results

Important columns

- Gender , smoking , age , weight(kg) , height(cm)

The shape DataSet

- Rows= 55692 , Columns = 27

The percentage gender in the dataset

- Female = 36.4%
- Male =63.6 %

The percentage gender in the dataset

- Non-smoking = 63.3%
- Smoking =36.7 %

Describe age

- mean = 44
- min = 20
- max = 85

Describe height- cm

- mean= 164
- min = 130
- max = 190

Describe weight(kg)

- mean= 65
- min = 30
- max = 135

The average age smoking

- age 45 = non- somking
- age 41 = somking

Analysis Results

The average data

Female

- somking avg /> age= 46 ,weght =56 kg ,height 157 cm
- non-somking ave / >age= 49 ,weght =56 kg ,height 165 cm

Male

- somking avg /> age= 41 ,weght =72 kg ,height 170 cm
- non-somking ave / >age= 42 ,weght =71 kg ,height 170 cm

Data Preprocessing

```
X_train = pd.read_csv('../input/body-signal-of-smoking/competition_format/x_train.csv')
X_test = pd.read_csv('../input/body-signal-of-smoking/competition_format/x_test.csv')
y_train = pd.read_csv('../input/body-signal-of-smoking/competition_format/y_train.csv')
y_test = pd.read_csv('../input/body-signal-of-smoking/competition_format/y_test.csv')
```

Importing the train dataset and test datest

Data Preprocessing

```
l = [X_train,X_test,y_train,y_test]
for i in l:
    i.info()
    print('=====================================\n\n\n')
```

look test dataset and train dataset

Data Preprocessing

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44553 entries, 0 to 44552
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   44553 non-null  int64
 1   gender               44553 non-null  object
 2   age                  44553 non-null  int64
 3   height(cm)           44553 non-null  int64
 4   weight(kg)           44553 non-null  int64
 5   waist(cm)            44553 non-null  float64
 6   eyesight(left)       44553 non-null  float64
 7   eyesight(right)      44553 non-null  float64
 8   hearing(left)        44553 non-null  float64
 9   hearing(right)       44553 non-null  float64
 10  systolic             44553 non-null  float64
 11  relaxation           44553 non-null  float64
 12  fasting blood sugar  44553 non-null  float64
 13  Cholesterol          44553 non-null  float64
 14  triglyceride         44553 non-null  float64
 15  HDL                  44553 non-null  float64
 16  LDL                  44553 non-null  float64
 17  hemoglobin           44553 non-null  float64
 18  Urine protein        44553 non-null  float64
 19  serum creatinine     44553 non-null  float64
 20  AST                  44553 non-null  float64
 21  ALT                  44553 non-null  float64
 22  Gtp                  44553 non-null  float64
 23  oral                 44553 non-null  object
 24  dental caries        44553 non-null  float64
 25  tartar               44553 non-null  object
dtypes: float64(19), int64(4), object(3)
memory usage: 8.8+ MB
====================================
```

X_train

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11139 entries, 0 to 11138
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   11139 non-null  int64
 1   gender               11139 non-null  object
 2   age                  11139 non-null  int64
 3   height(cm)           11139 non-null  int64
 4   weight(kg)           11139 non-null  int64
 5   waist(cm)            11139 non-null  float64
 6   eyesight(left)       11139 non-null  float64
 7   eyesight(right)      11139 non-null  float64
 8   hearing(left)        11139 non-null  float64
 9   hearing(right)       11139 non-null  float64
 10  systolic             11139 non-null  float64
 11  relaxation           11139 non-null  float64
 12  fasting blood sugar  11139 non-null  float64
 13  Cholesterol          11139 non-null  float64
 14  triglyceride         11139 non-null  float64
 15  HDL                  11139 non-null  float64
 16  LDL                  11139 non-null  float64
 17  hemoglobin           11139 non-null  float64
 18  Urine protein        11139 non-null  float64
 19  serum creatinine     11139 non-null  float64
 20  AST                  11139 non-null  float64
 21  ALT                  11139 non-null  float64
 22  Gtp                  11139 non-null  float64
 23  oral                 11139 non-null  object
 24  dental caries        11139 non-null  float64
 25  tartar               11139 non-null  object
dtypes: float64(19), int64(4), object(3)
memory usage: 2.2+ MB
```

X_test

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44553 entries, 0 to 44552
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   ID       44553 non-null  int64
 1   smoking  44553 non-null  int64
dtypes: int64(2)
memory usage: 696.3 KB
====================================
```

Y_train

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11139 entries, 0 to 11138
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   ID       11139 non-null  int64
 1   smoking  11139 non-null  int64
dtypes: int64(2)
memory usage: 174.2 KB
====================================
```

Y_test

Data Preprocessing

```
del X_train["oral"]
del X_test["oral"]
```

OneHotEncoding and OrdinalEncoding

```
l3 = [X_train,X_test]
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import OrdinalEncoder
ONE = OneHotEncoder(handle_unknown='ignore')

def oneHot(df,a):
    cat_encoder = OneHotEncoder()
    ec_cat=cat_encoder.fit_transform(df[[a]])
    return ec_cat.toarray()
X_train['gender'] = oneHot(X_train,'gender')
X_test['gender'] = oneHot(X_test,'gender')
```

```
from sklearn.preprocessing import OrdinalEncoder
ordinal_encoder = OrdinalEncoder(categories = [['N','Y']])
X_train["tartar"] = ordinal_encoder.fit_transform(X_train[["tartar"]])
X_test["tartar"] = ordinal_encoder.fit_transform(X_test[["tartar"]])
```

Data Preprocessing

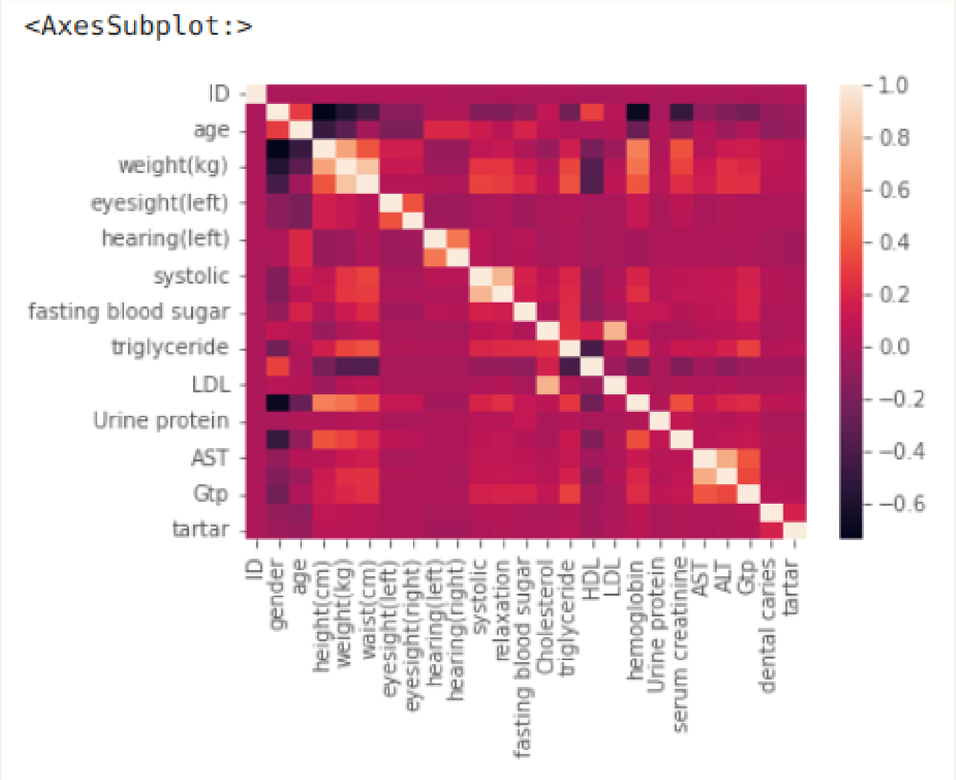
```
from sklearn.metrics import accuracy_score,r2_score


from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import  KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import  MLPClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier


from sklearn.preprocessing import LabelEncoder
from sklearn.inspection import permutation_importance
```

Importing the basic librarires for building model - classification

Data Preprocessing

```
sns.heatmap(X_train.corr())
```

Correlation heatmap



DecisionTreeClassifier

```
DTC = DecisionTreeClassifier()
DTC.fit(X_train, y_train)

y_pred = DTC.predict(X_test)

print("Score the X-train with Y-train is : ", DTC.score(X_train,y_train))
print("Score the X-test  with Y-test  is : ", DTC.score(X_test,y_test))
print("Accuracy Score :",accuracy_score(y_test,y_pred)*100)
```

Score the X-train with Y-train is : 1.0

Score the X-test with Y-test is : 0.9383248047401024

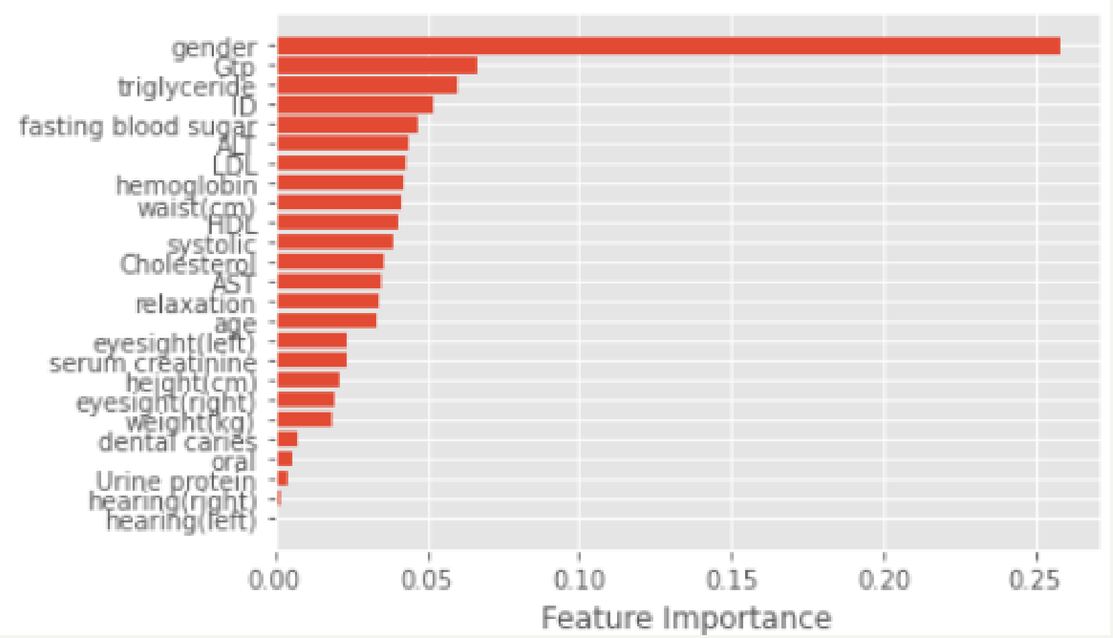
Accuracy Score : 93.83248047401024

DecisionTreeClassifier

```
sort = DTC.feature_importances_.argsort()
plt.barh(df.columns[sort], DTC.feature_importances_[sort])
plt.xlabel("Feature Importance")
```

Feature importances

Text(0.5, 0, 'Feature Importance')



RandomForestClassifier

```
rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train, y_train)

ypred = rf.predict(X_test)

print("Score the X-train with Y-train is : ", rf.score(X_train,y_train))
print("Score the X-test  with Y-test  is : ", rf.score(X_test,y_test))
print("Accuracy Score :",accuracy_score(y_test,ypred)*100)
```

Score the X-train with Y-train is : 1.0

Score the X-test with Y-test is : 0.9927282520872 61

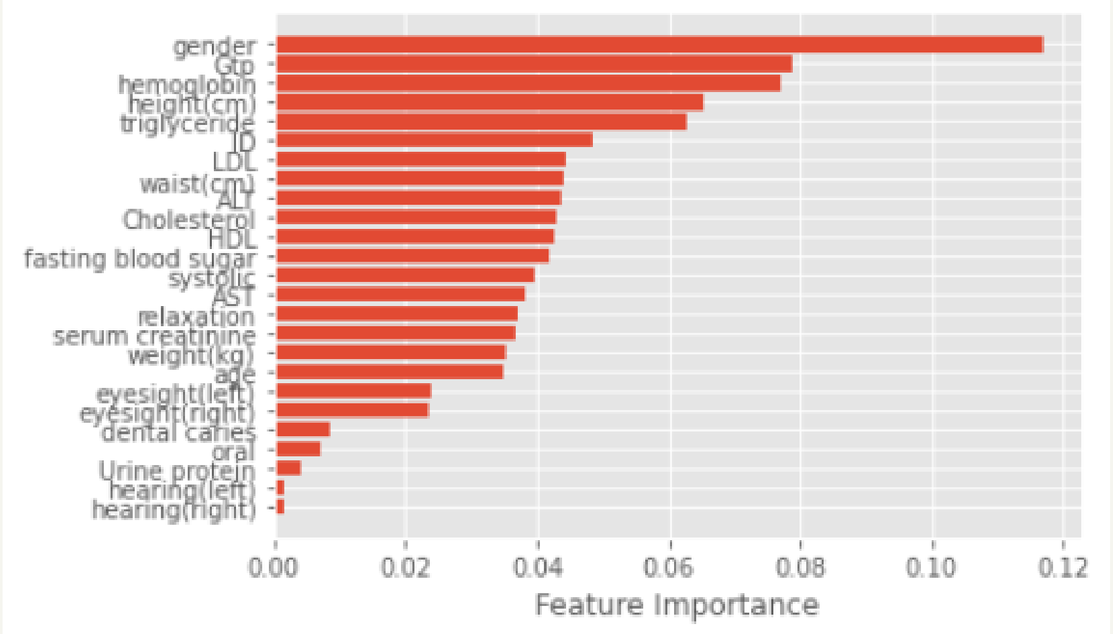
Accuracy Score : 99.27282520872 61

RandomForestClassifier

```
#feature_importances

sort = rf.feature_importances_.argsort()
plt.barh(df.columns[sort], rf.feature_importances_[sort])
plt.xlabel("Feature Importance")
```

Feature importances

Text(0.5, 0, 'Feature Importance')



Model Selection Results

Decision Tree Classifier = 93.6 %

Random Forest Classifier = 99.2 %

Conclusion

모델에 따른 성능 차이 확인

머신러닝에 대한 관심

캐글 문제 풀이

Conclusion

감사합니다.