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

from ipywidgets import interact_manual
import plotly.express as px

ts=pd.read_csv("/content/test.csv")
tr=pd.read_csv("/content/train.csv")

tr.columns

ts.head()

print(ts.shape)

tr.isnull().sum()

ts.isnull().sum()

percent_missing=tr.isnull().sum()*100/len(tr)
percent_missing

train=tr.dropna(axis=0,how='any')
test=ts.dropna(axis=0,how='any')

train.shape

@interact_manual

def viz(x=list(train.select_dtypes('number').columns)):
  sns.countplot(train[x])

px.bar(tr,x='gender',y='stroke')

train.groupby(['gender'])['stroke'].value_counts()

train['smoking_status'].value_counts()

train.groupby(['gender'])['smoking_status'].value_counts()

sns.countplot(x=train['gender'],hue=train["smoking_status"])
plt.title("Gender vs type of smokers",fontsize=15)
plt.show()

str_train=train.select_dtypes(include=['object'])
str_test=test.select_dtypes(include=['object'])

int_train=train.select_dtypes(include=['int','float'])
```

```
int_test=test.select_dtypes(include=['int','float'])

from sklearn.preprocessing import LabelEncoder
label=LabelEncoder()
train_data=str_train.apply(label.fit_transform)
train_data=train_data.join(int_train)
train_data.head()

# prompt: Using dataframe train_data: age

train_data['age'].describe()


test_data=str_test.apply(label.fit_transform)
test_data=test_data.join(int_test)
test_data.head()

xtrain=train_data.drop(["stroke"],axis=1)
xtrain.shape

ytrain=train_data ["stroke"]
ytrain.shape

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(xtrain,ytrain,test_size=0.5)

print(x_test.shape)
print(y_test.shape)

x_train.head()

y_train.head()


x_test.head()

y_test.head()

from sklearn.tree import DecisionTreeClassifier
df_model=DecisionTreeClassifier()
df_model.fit(x_train,y_train)

y_predict=df_model.predict(x_test)
print(y_predict)

dtscore=df_model.score(x_test,y_test)
print("Decision Tree",dtscore)

from sklearn.ensemble import RandomForestClassifier
```

```
rf=RandomForestClassifier(n_estimators=100)

rf.fit(x_train,y_train)
y_pred_rfc=rf.predict(x_test)
rfscore=rf.score(x_test,y_test)
print("Random Forest",rfscore)

print('Score:', rf.score(x_test, y_test))

from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()

lr.fit(x_train,y_train)

y_pred=lr.predict(x_test)

lrmodel=lr.score(x_test,y_test)
print(lrmodel)
```