1 INTRODUCTION

1.1 Overview

Nowadays, technology is very important. Since everyone in the family works in a different industry and has limited free time, people are becoming more and more dependent on junk food, hybrid vegetables, etc. As technology develops more swiftly, people are not eating enough nutritious food, resulting in at least one member of every family suffering health problems. They must therefore contact a doctor and be prepared to take the medication that contains the drugs.

As a dataset for this project, we have some patient characteristics. This dataset's target variable is drugs. The names of the drugs are kept secret. In order to avoid confusion, those names are now DrugX, DrugY, DrugA, DrugB, and DrugC. The approach used to solve this kind of problem, as well as the feature sets and target sets, is the key issue here.

We use classification algorithms like Decision tree, Random forest, KNN, and xgboost in this situation.These algorithms are evaluated and trained on the data. The best model is chosen and stored in pkl format from this set. and will be carrying out IBM deployment and flask integration.

1.2 Purpose

we must pay a doctor's fee and other expenses each time you see a doctor. We can use this web program me to forecast your medicine type to save money and time. The primary goal of the Drug Classification system is to accurately forecast the sort of drug that will work best for patients based on those patients' characteristics.

2 LITERATURE SURVEY

2.1 Existing problem

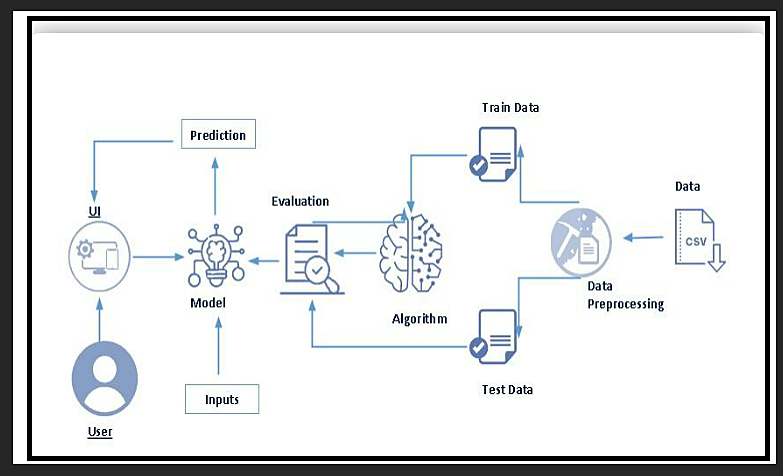
Our way of life has changed in recent years. At least one member of each family owns with health difficulties, too. With hybrid vegetables, junk food, etc., we have completely changed. We are not getting enough nutrients from these meals, which is why we are ill. We are seeking medical advice and using medicines which are drugs to tackle this. We must o

2.2 Proposed solution

we must pay a doctor's fee and other expenses each time you see a doctor. We can use this web program to forecast your medicine type to save money and time. The major goal of the medication classification system is to accurately forecast the patient's preferred medicine or drug type based on their features.

3 THEORITICAL ANALYSIS

3.1 Block diagram



3.2 Software Specifications :

Python packages for the Ananconda navigator

Watson Studio by IBM

Machine Learning from IBM

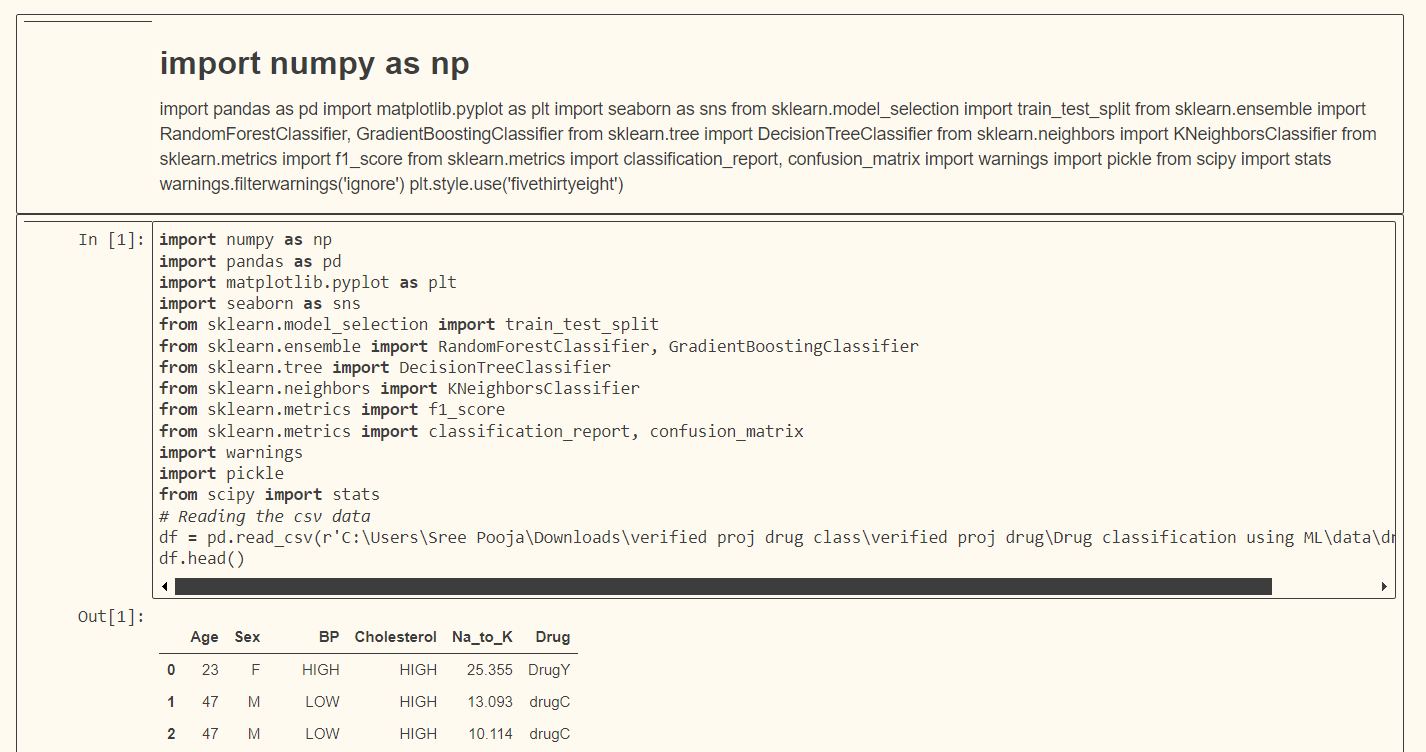
Cloud Object Storage from IBM

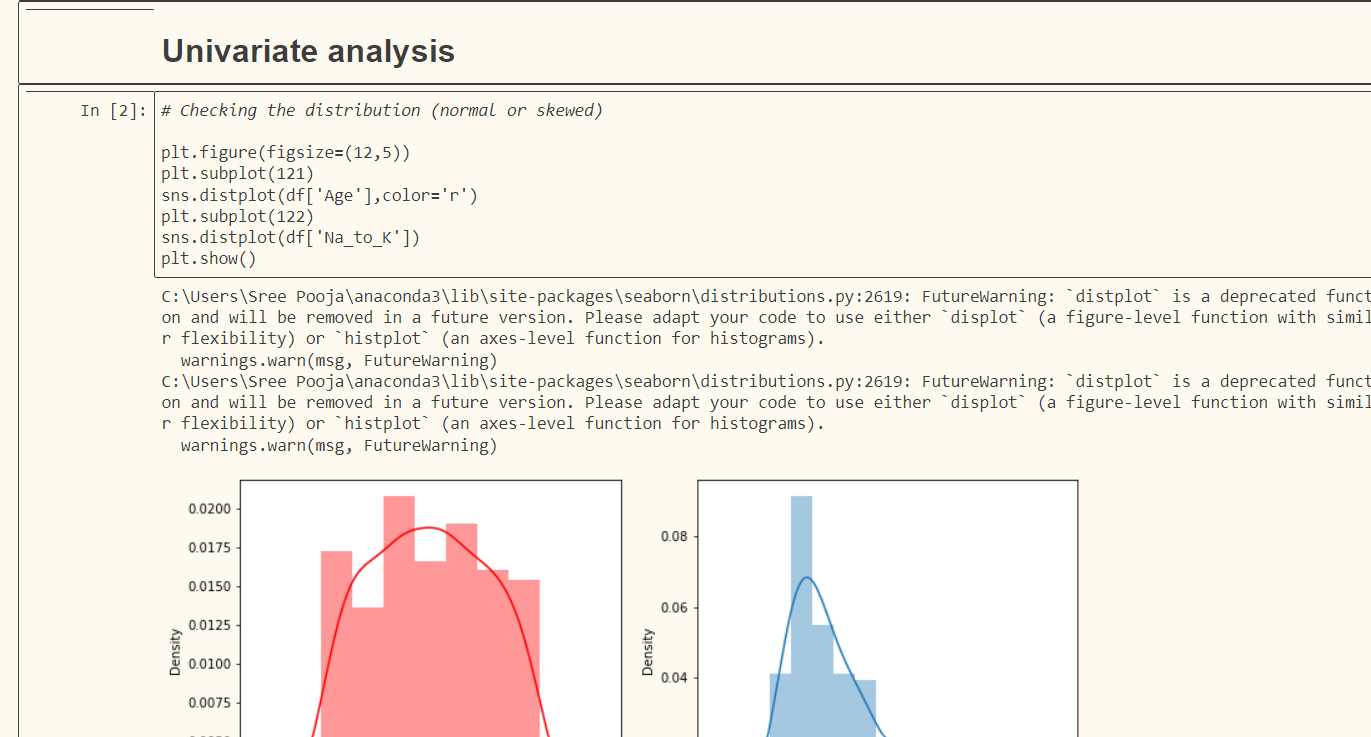
4 EXPERIMENTAL INVESTIGATIONS

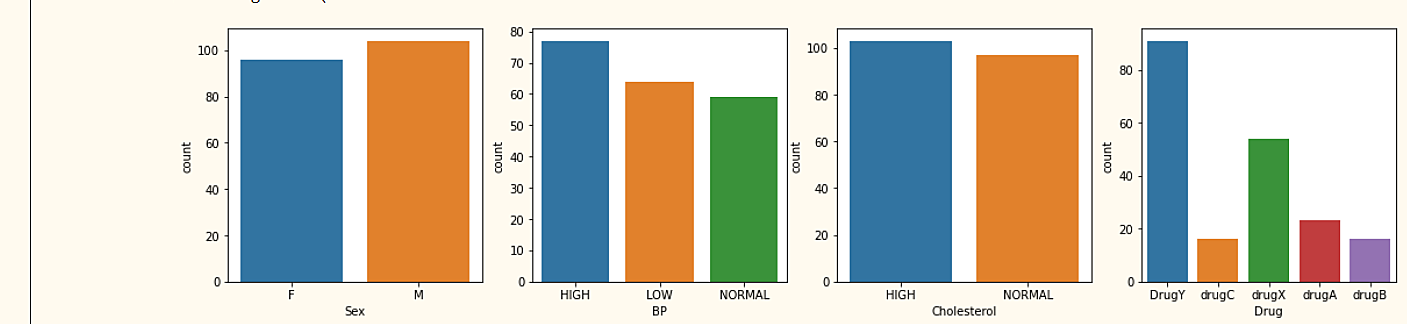
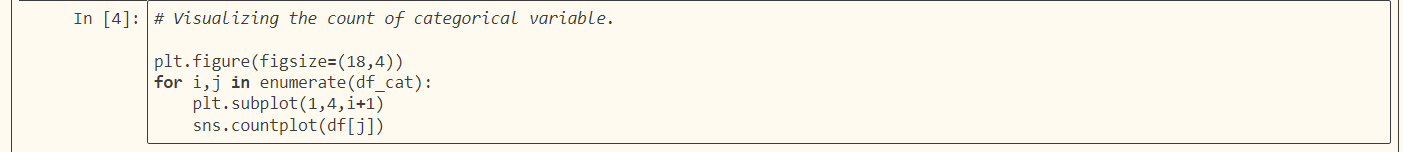
The model's accuracy is significantly influenced by the shape of the inputs.

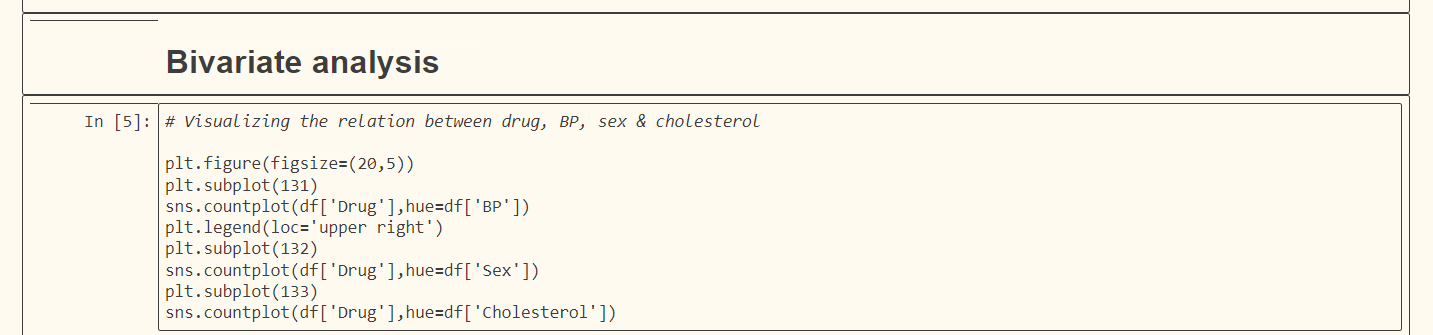
IBM Cloud supports the deployment of machine learning models and the validation of our model's accuracy.

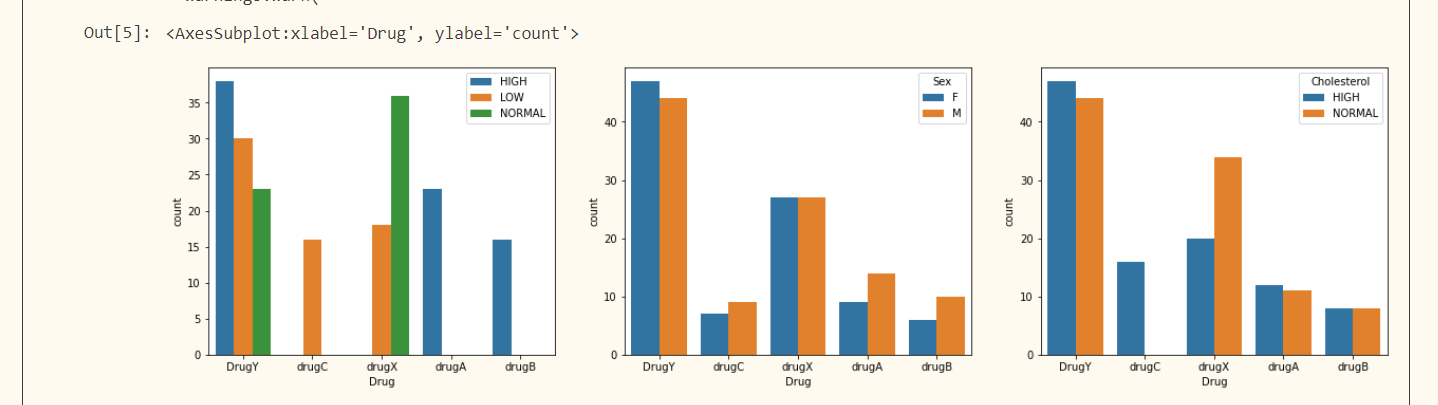
To produce accurate predictions, integrating Flask with the machine learning model requires extensive data preprocessing.

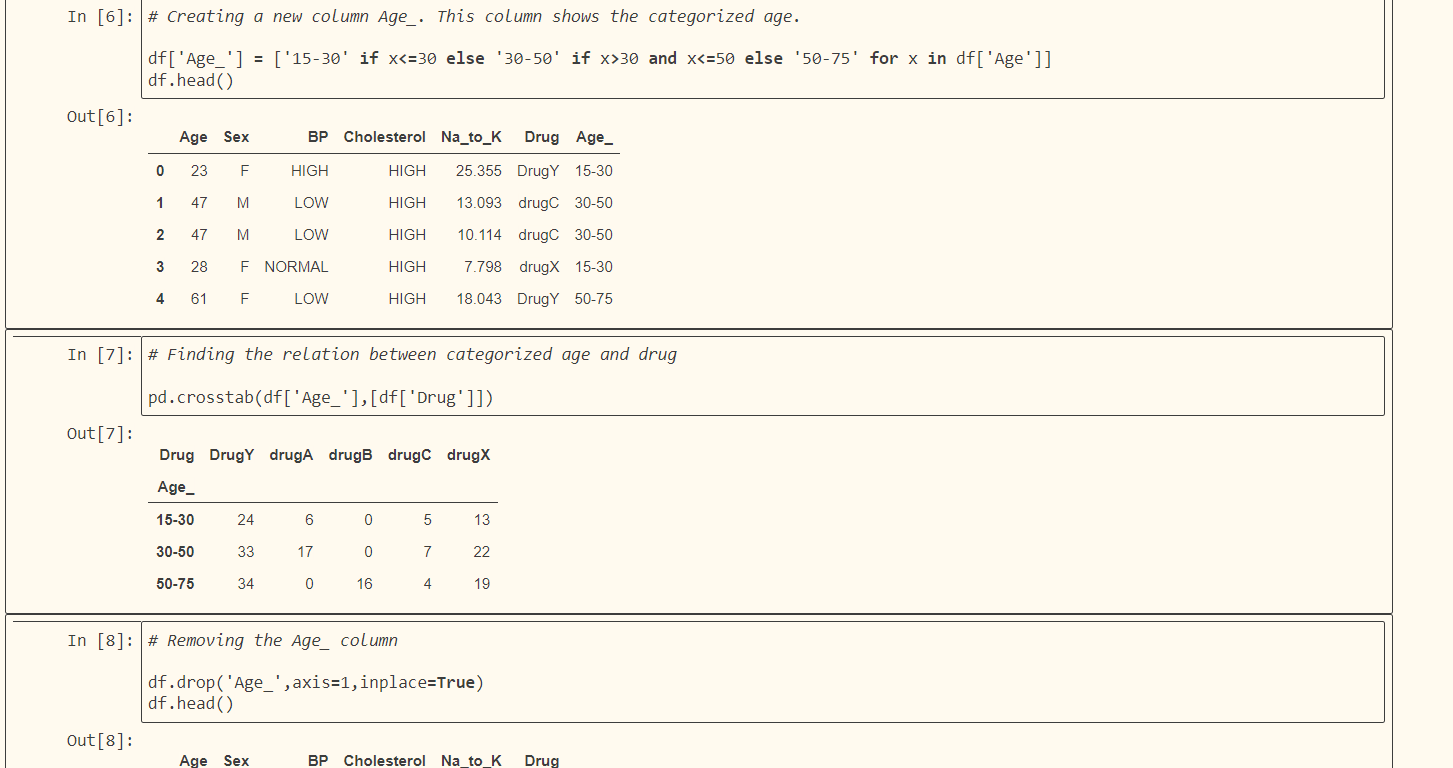


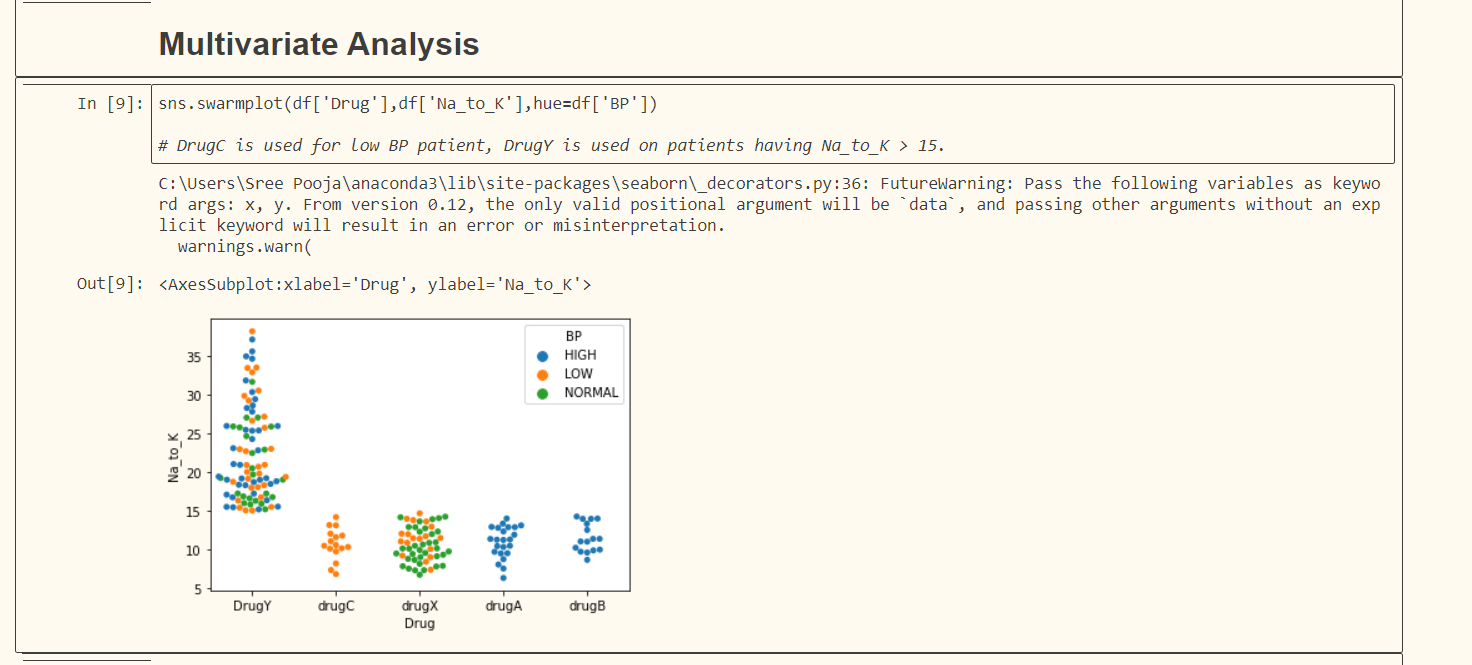


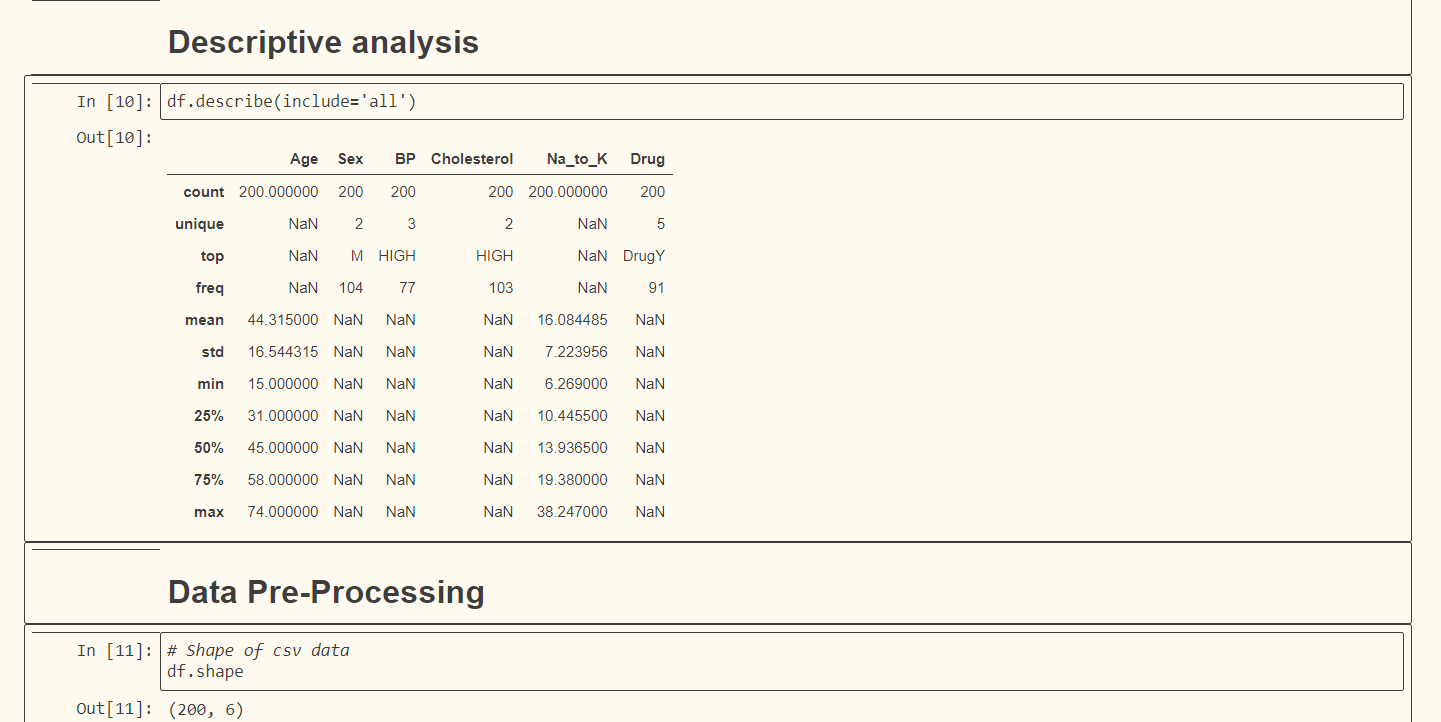


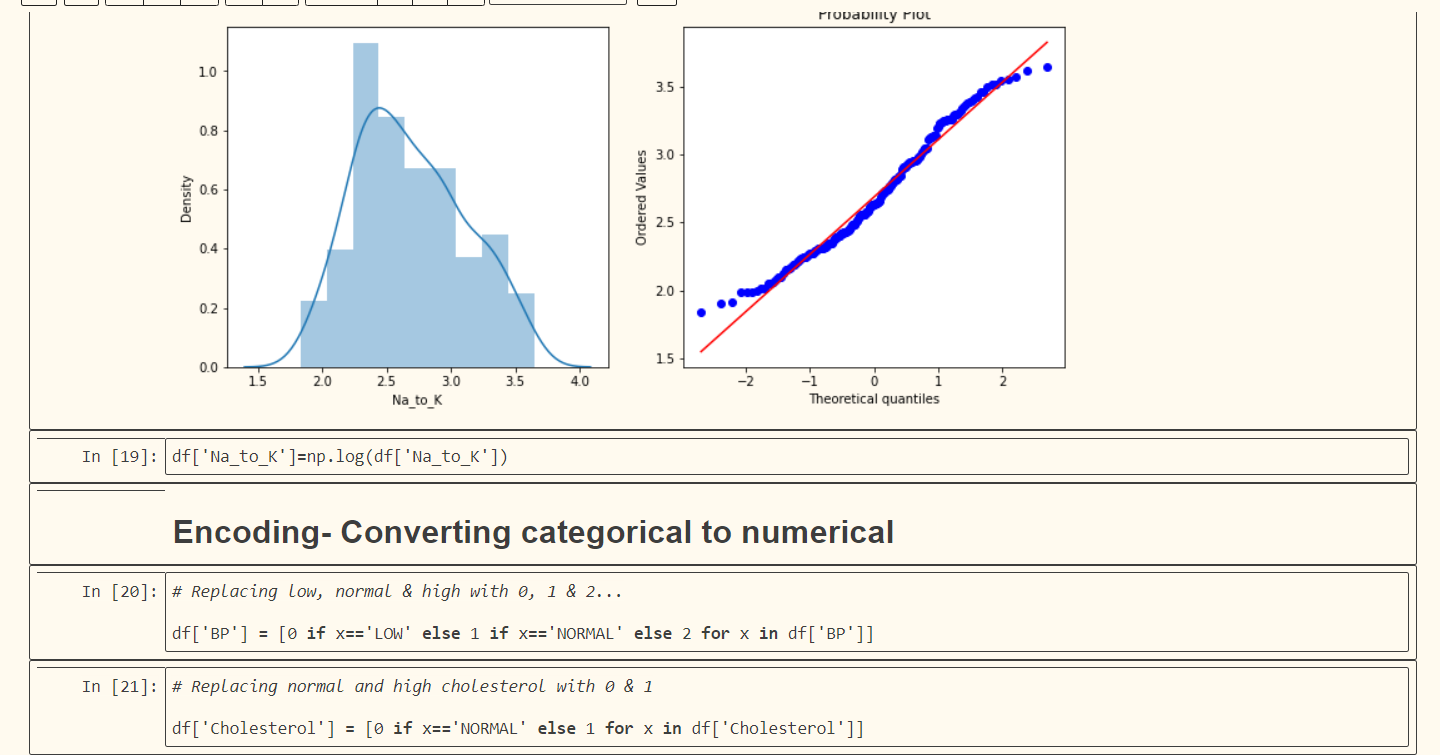


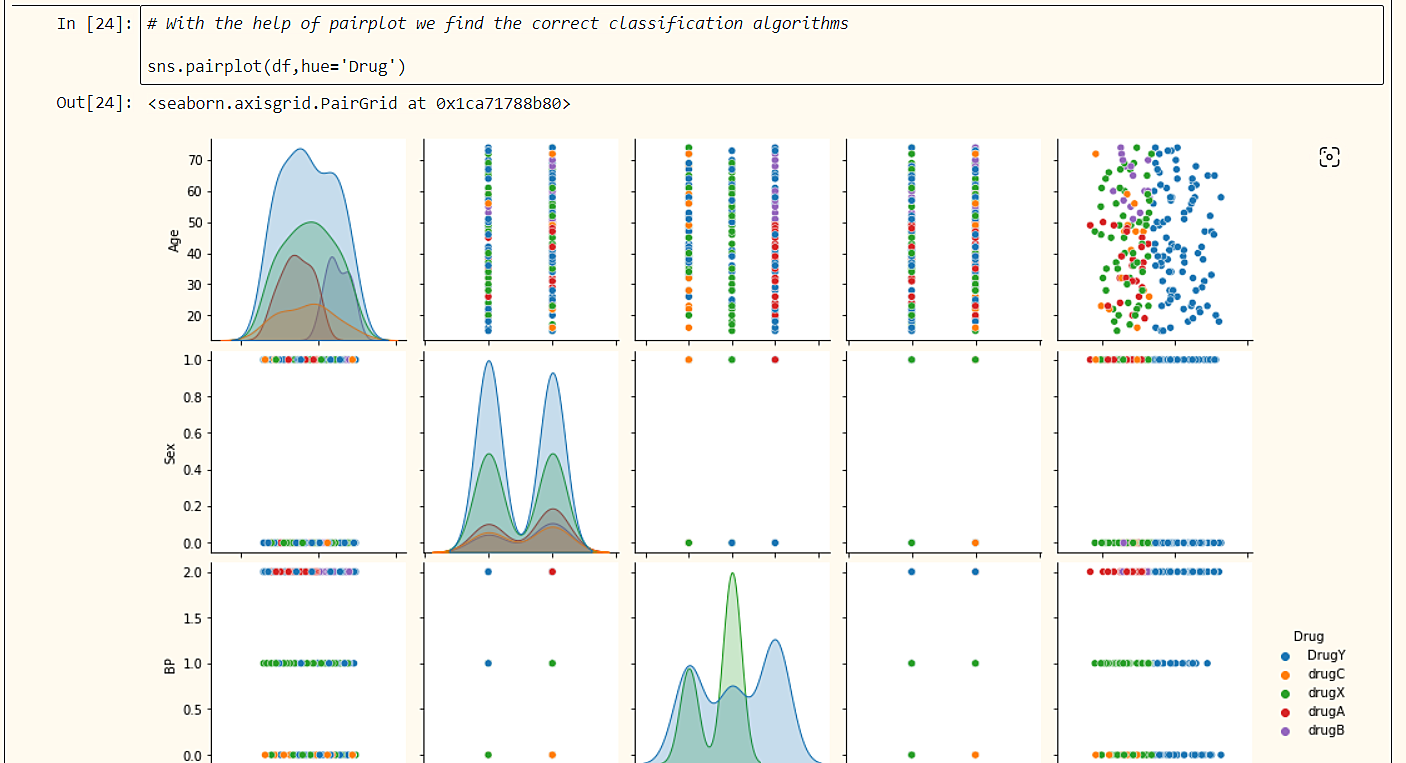


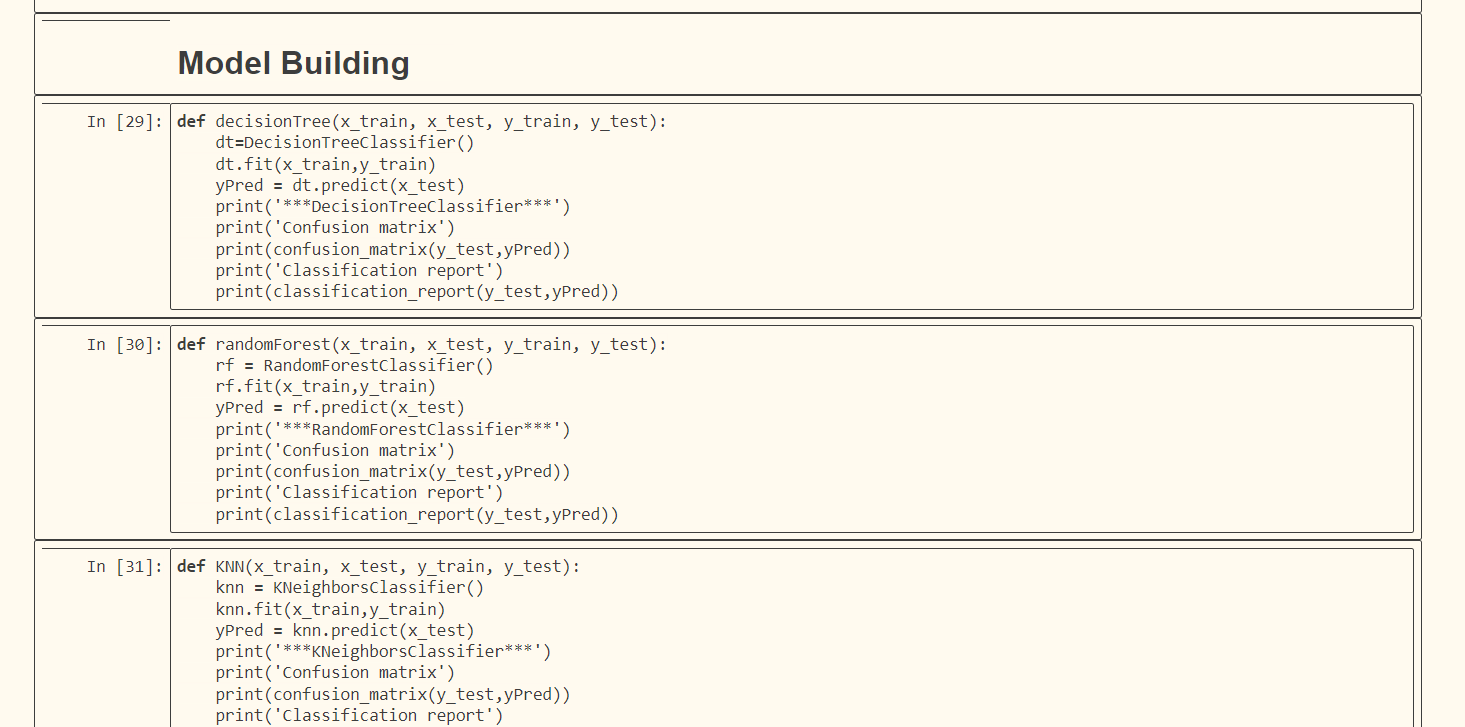


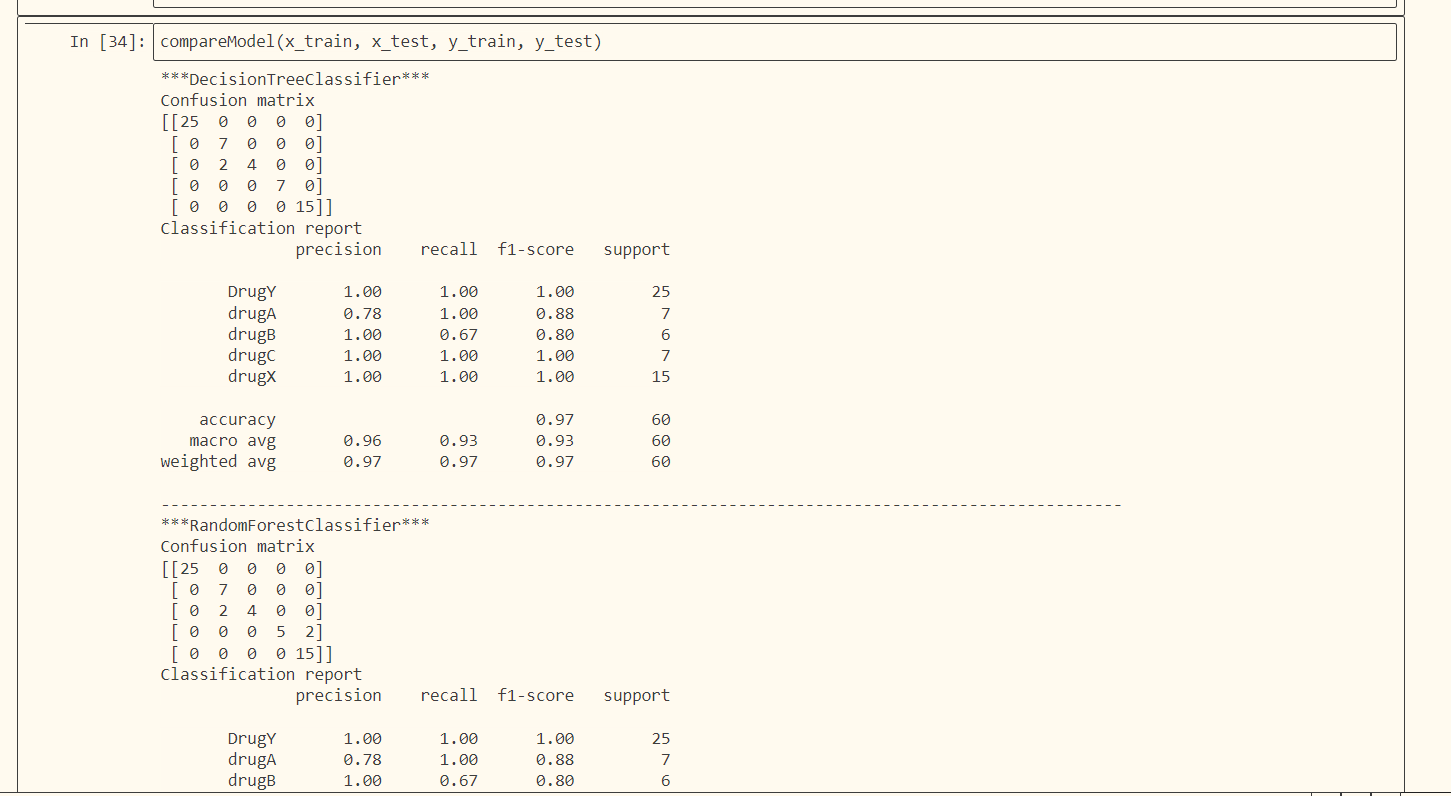




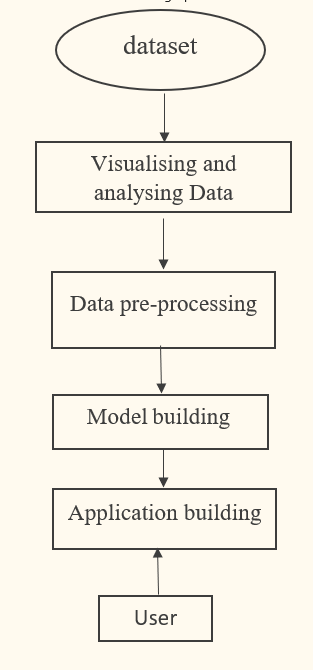


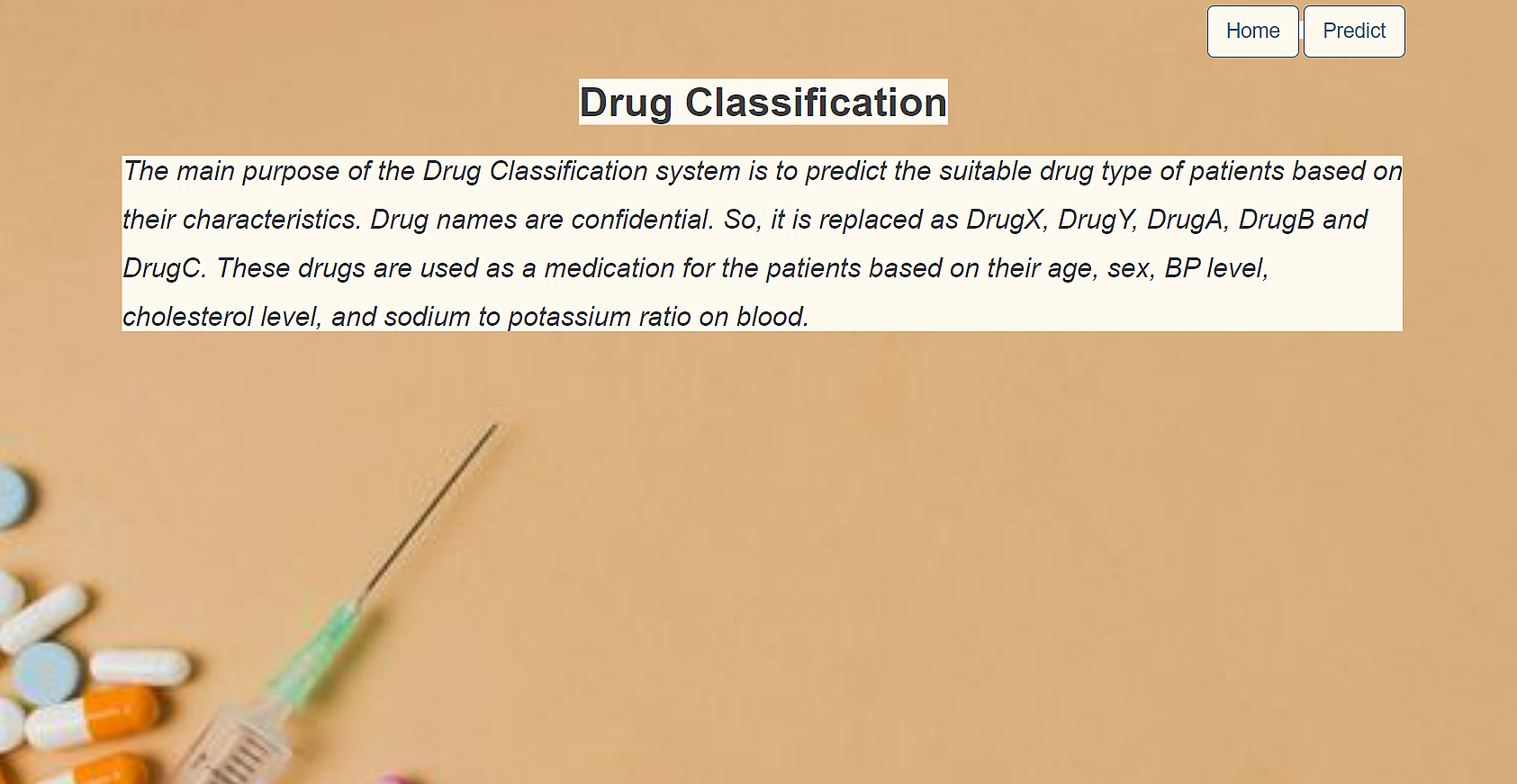


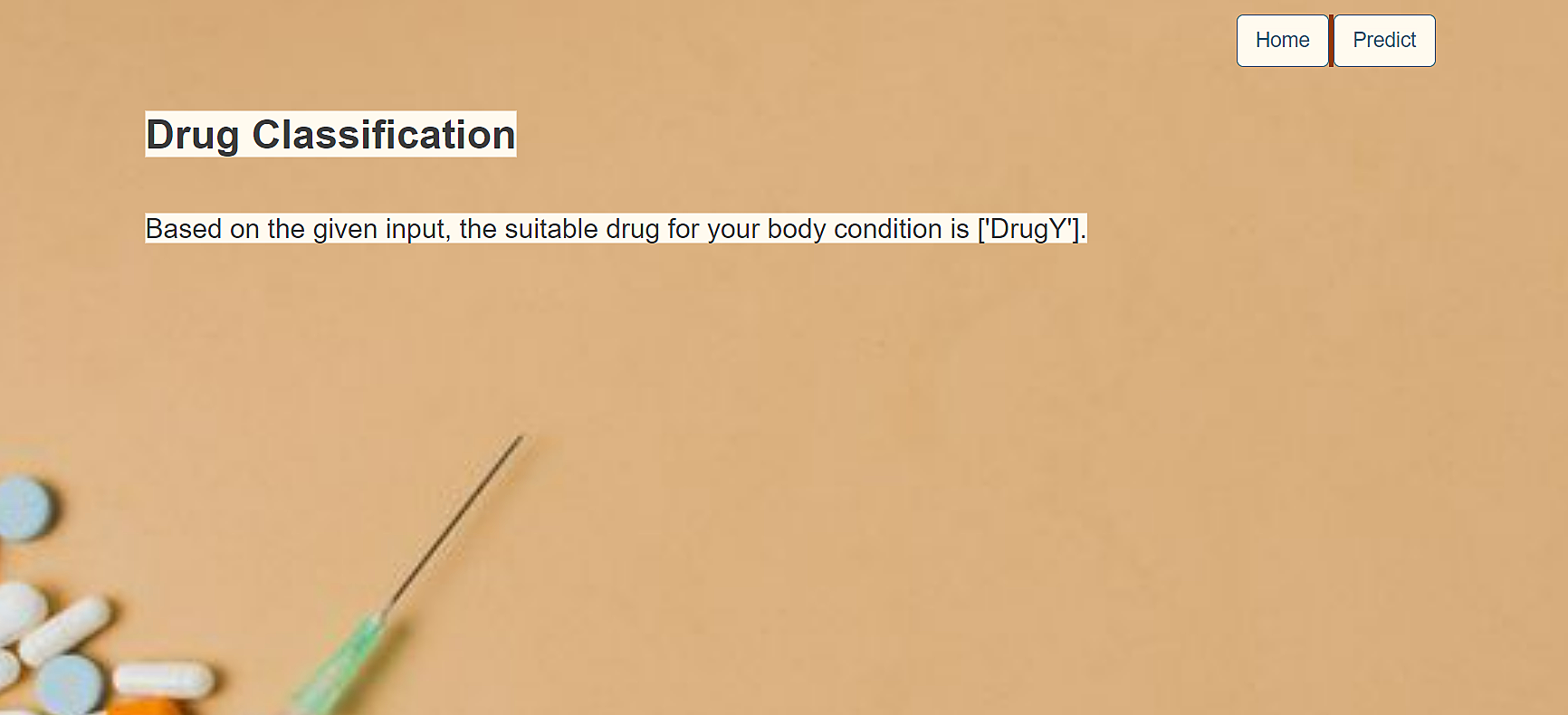
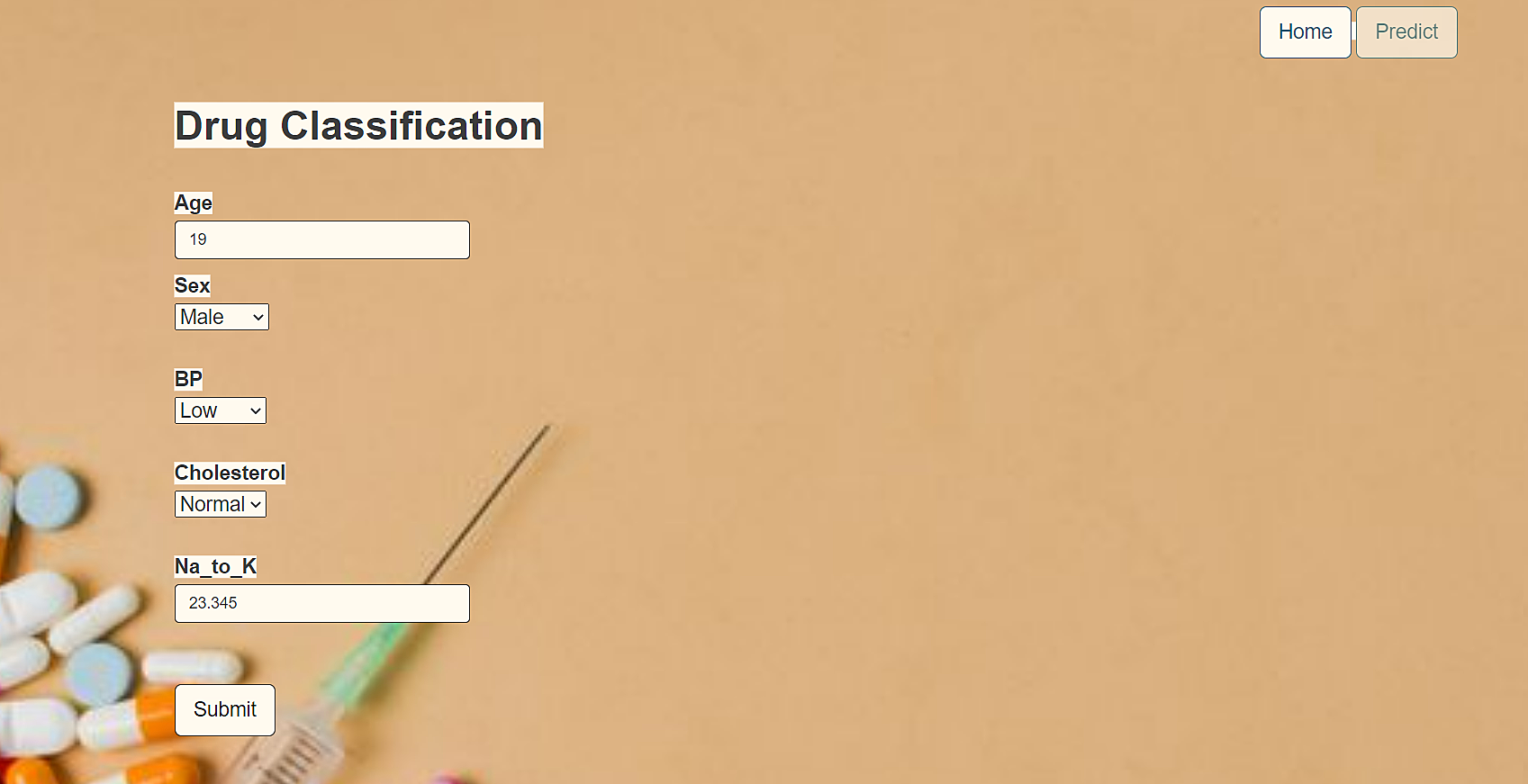




5 FLOWCHART



6 RESULT



7 ADVANTAGES AND DISADVANTAGES

7.1 ADVANTAGES

It decreases time and costs money.

It facilitates drug type prediction.

7.2 DISADVANtAGES

Without a doctor's prescription, the drug poses a small risk of decreased comfort and satisfaction and could result in allergies.

8 APPLICATION

In Medical Store

9 CONCLUSION

The Drug Classification system help us to predict the suitable drug type confidently for the patients based on their characteristics. It help us to save time and money.Prescription drugs are a key component in the healthcare.Discusses the challenges to providing accessible prescription drugs

Explores technological advances and new business models in providing pharmacy services

10 FUTURE SCOPE

The Drug Classification helps us to save time and money and without consulting a doctor we will able to predict the drug type.We can also add doctor prescriptions as well as test results such as blood test, ECG, Bp ect. At the same time users can view their old data.

11 BIBILIOGRAPHY

<https://medlineplus.gov/cholesterolmedicines.html>

<https://www.nhs.uk/conditions/high-cholesterol/medicines-for-high-cholesterol/>

<https://medlineplus.gov/ency/article/007278.htm>

12 APPENDIX

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import f1\_score

from sklearn.metrics import classification\_report, confusion\_matrix

import warnings

import pickle

from scipy import stats

warnings.filterw

warnings.filterwarnings('ignore')

plt.style.use('fivethirtyeight')

df = pd.read\_csv(r'C:\Users\drug200 (1).csv')

df.head()

plt.figure(figsize=(12,5))

plt.subplot(121)

sns.distplot(df['Age'],color='r')

plt.subplot(122)

sns.distplot(df['Na\_to\_K'])

plt.show()

df\_cat = df.select\_dtypes(include='object')

df\_cat.head()

plt.figure(figsize=(18,4))

for i,j in enumerate(df\_cat):

plt.subplot(1,4,i+1)

sns.countplot(df[j])

plt.figure(figsize=(20,5))

plt.subplot(131)

sns.countplot(df['Drug'],hue=df['BP'])

plt.legend(loc='upper right')

plt.subplot(132)

sns.countplot(df['Drug'],hue=df['Sex'])

plt.subplot(133)

sns.countplot(df['Drug'],hue=df['Cholesterol'])

df['Age\_'] = ['15-30' if x<=30 else '30-50' if x>30 and x<=50 else '50-75' for x in df['Age']]

df.head()

pd.crosstab(df['Age\_'],[df['Drug']])

sns.swarmplot(df['Drug'],df['Na\_to\_K'],hue=df['BP'])

df.describe(include='all')

df.shape

(200,6)

df.isnull().sum()

plt.figure(figsize=(12,5))

sns.boxplot(df['Na\_to\_K'])

plt.show()

print(stats.mode(df['Na\_to\_K']))

print(np.mean(df['Na\_to\_K']))

q1 = np.quantile(df['Na\_to\_K'],0.25)

q3 = np.quantile(df['Na\_to\_K'],0.75)

IQR = q3-q1

upper\_bound = q3+(1.5\*IQR)

lower\_bound = q1-(1.5\*IQR)

print('q1 :',q1)

print('q3 :',q3)

print('IQR :',IQR)

print('Upper Bound :',upper\_bound)

print('Lower Bound :',lower\_bound)

print('Skewed data :',len(df[df['Na\_to\_K']>upper\_bound]))

print('Skewed data :',len(df[df['Na\_to\_K']<lower\_bound]))

def transformationPlot(feature):

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)

sns.distplot(feature)

plt.subplot(1,2,2)

stats.probplot(feature,plot=plt)

df['Na\_to\_K']=np.log(df['Na\_to\_K'])

df['BP'] = [0 if x=='LOW' else 1 if x=='NORMAL' else 2 for x in df['BP']]

df['Cholesterol'] = [0 if x=='NORMAL' else 1 for x in df['Cholesterol']]

df['Sex'] = [0 if x=='F' else 1 for x in df['Sex']]

df.head()

sns.pairplot(df,hue='Drug')

x = df.drop('Drug',axis=1)

x.head()

y = df['Drug']

y.head()

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.3, random\_state=10)

print('Shape of x\_train {}'.format(x\_train.shape))

print('Shape of y\_train {}'.format(y\_train.shape))

print('Shape of x\_test {}'.format(x\_test.shape))

print('Shape of y\_test {}'.format(y\_test.shape))

x\_train.drop("Age\_",axis=1,inplace=True)

x\_test.drop("Age\_",axis=1,inplace=True)

x\_train

def decisionTree(x\_train, x\_test, y\_train, y\_test):

dt=DecisionTreeClassifier()

dt.fit(x\_train,y\_train)

yPred = dt.predict(x\_test)

print('\*\*\*DecisionTreeClassifier\*\*\*')

print('Confusion matrix')

print(confusion\_matrix(y\_test,yPred))

print('Classification report')

print(classification\_report(y\_test,yPred))

def randomForest(x\_train, x\_test, y\_train, y\_test):

rf = RandomForestClassifier()

rf.fit(x\_train,y\_train)

yPred = rf.predict(x\_test)

print('\*\*\*RandomForestClassifier\*\*\*')

print('Confusion matrix')

print(confusion\_matrix(y\_test,yPred))

print('Classification report')

print(classification\_report(y\_test,yPred))

def KNN(x\_train, x\_test, y\_train, y\_test):

knn = KNeighborsClassifier()

knn.fit(x\_train,y\_train)

yPred = knn.predict(x\_test)

print('\*\*\*KNeighborsClassifier\*\*\*')

print('Confusion matrix')

print(confusion\_matrix(y\_test,yPred))

print('Classification report')

print(classification\_report(y\_test,yPred))

def xgboost(x\_train, x\_test, y\_train, y\_test):

xg = GradientBoostingClassifier()

xg.fit(x\_train,y\_train)

yPred = xg.predict(x\_test)

print('\*\*\*GradientBoostingClassifier\*\*\*')

print('Confusion matrix')

print(confusion\_matrix(y\_test,yPred))

print('Classification report')

print(classification\_report(y\_test,yPred))

def compareModel(x\_train, x\_test, y\_train, y\_test):

decisionTree(x\_train, x\_test, y\_train, y\_test)

print('-'\*100)

randomForest(x\_train, x\_test, y\_train, y\_test)

print('-'\*100)

KNN(x\_train, x\_test, y\_train, y\_test)

print('-'\*100)

xgboost(x\_train, x\_test, y\_train, y\_test)

compareModel(x\_train, x\_test, y\_train, y\_test)

from sklearn.model\_selection import cross\_val\_score

rf = RandomForestClassifier()

rf.fit(x\_train,y\_train)

yPred = rf.predict(x\_test)

f1\_score(yPred,y\_test,average='weighted')

confusion\_matrix(y\_test,ypred)

print(classification\_report(y\_test,ypred))

pickle.dump(rf,open('model.pkl','wb'))

pickle.dump(rf,open('model.pkl','wb'))

!tar -zcvf drug-classification-model\_new.tgz model.pkl

ls -1

!pip install watson-machine-learning-client --upgrade

from ibm\_watson\_machine\_learning import APIClient

wml\_credentials = {

"url": "[https://us-south.ml.cloud.ibm.com](https://us-south.ml.cloud.ibm.com/)",

"apikey": "exj3cobWRr1L6iRO1AKBNruQ2Q6MPpiELRhxYF0t5R-D"

}

client = APIClient(wml\_credentials)

def guid\_from\_space\_name(client, space\_name):

space = client.spaces.get\_details()

#print(space)

return(next(item for item in space['resources'] if item['entity']["name"] == space\_name)['metadata']['id'])

space\_uid = guid\_from\_space\_name(client, space\_name='drug')

print("Space UID = " + space\_uid)

client.set.default\_space(space\_uid)

client.software\_specifications.list()

import sklearn

sklearn.\_\_version\_\_

software\_spec\_uid= client.software\_specifications.get\_uid\_by\_name("default\_py3.9")

software\_spec\_uid

#Save model

model\_details = client.repository.store\_model(model='drug-classification-model\_new.tgz',

meta\_props={client.repository.ModelMetaNames.NAME:"DrugClassification",

client.repository.ModelMetaNames.TYPE:"scikit-learn\_1.0",

client.repository.ModelMetaNames.SOFTWARE\_SPEC\_UID:software\_spec\_uid

},

training\_data=x\_train,

training\_target=y\_train)

model\_id = client.repository.get\_model\_id(model\_details)

model\_id

# Deploy

deployment = client.deployments.create(

artifact\_uid=model\_id,

meta\_props={client.deployments.ConfigurationMetaNames.NAME:"Drugclassification",

client.deployments.ConfigurationmetaNames : ONLINE: {}

})