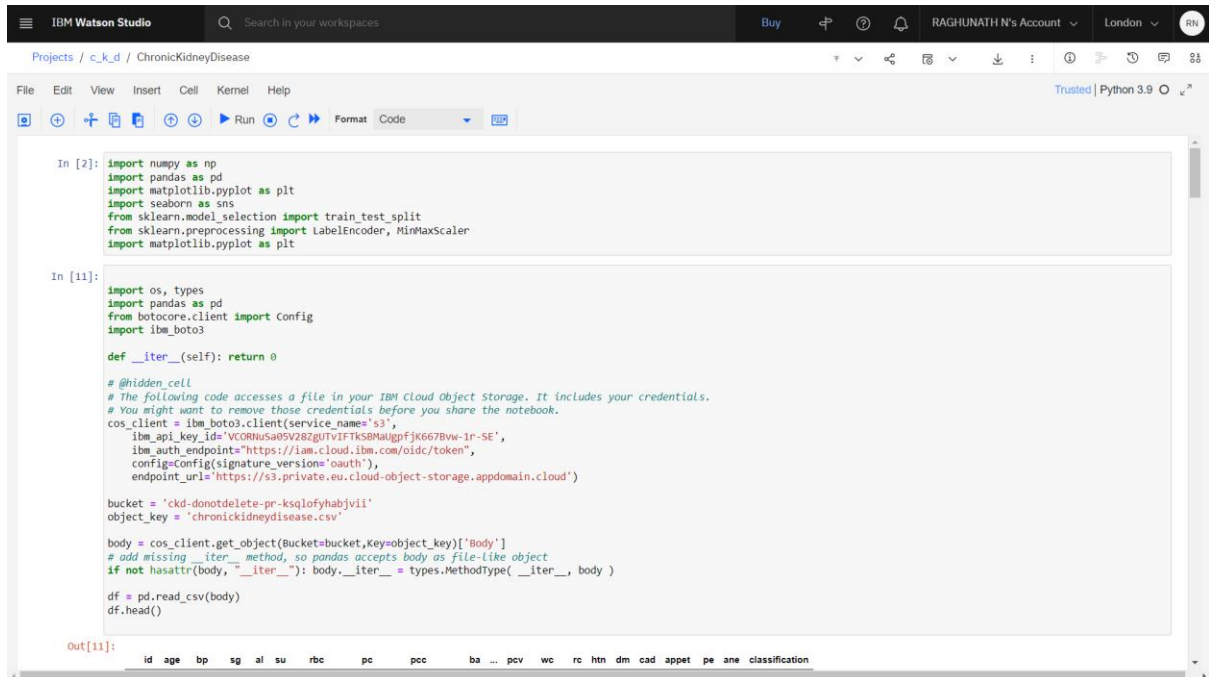


# Train the model on IBM Watson

Notebook link: <https://eu-gb.dataplatform.cloud.ibm.com/analytics/notebooks/v2/70d8ab7c-8e3a-429c-af1f-9986e9b9e8b1?projectid=151670a9-7074-4e8a-b1b6-215db4d345d9&context=cpdaas>



```
In [2]: 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.preprocessing import LabelEncoder, MinMaxScaler
import matplotlib.pyplot as plt

In [11]: import os, types
import pandas as pd
from boto3.client import Config
import boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = boto3.client(service_name='s3',
    iam_api_key_id='VCORNu8a05V28ZGUTVIFtK58MaUgpfjk667Bw-1r-SE',
    iam_auth_endpoint='https://iam.cloud.ibm.com/oidc/token',
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.eu.cloud-object-storage.appdomain.cloud')

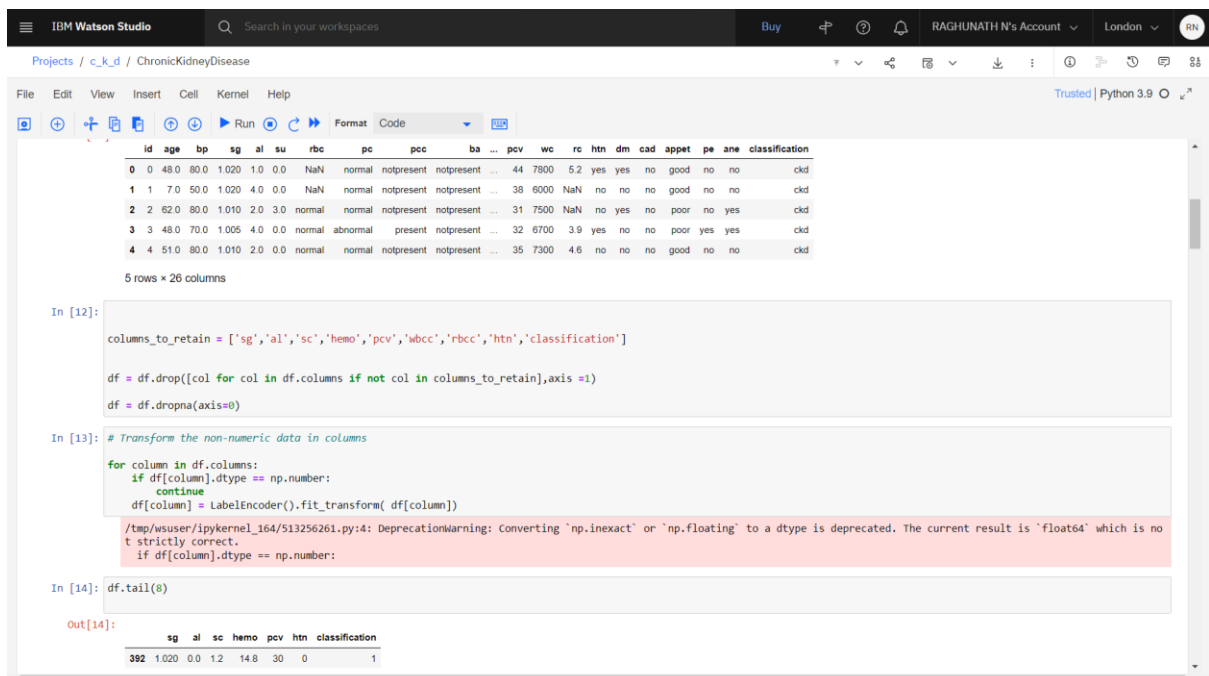
bucket = 'ckd-donotdelete-pr-ksqlofyhabvjii'
object_key = 'chronickidneydisease.csv'

body = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType(__iter__, body)

df = pd.read_csv(body)
df.head()
```

Out[11]:

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
--	----	-----	----	----	----	----	-----	----	-----	----	-----	-----	----	----	-----	----	-----	-------	----	-----	----------------



```
In [12]: columns_to_retain = ['sg', 'al', 'sc', 'hemo', 'pcv', 'wbcc', 'rbcc', 'htn', 'classification']

df = df.drop([col for col in df.columns if not col in columns_to_retain], axis=1)
df = df.dropna(axis=0)

In [13]: # Transform the non-numeric data in columns
for column in df.columns:
    if df[column].dtype == np.number:
        continue
    df[column] = LabelEncoder().fit_transform(df[column])

/tmp/vsuser/ipykernel_164/513256261.py:4: DeprecationWarning: Converting 'np.inexact' or 'np.floating' to a dtype is deprecated. The current result is 'float64' which is not strictly correct.
    if df[column].dtype == np.number:

In [14]: df.tail(8)
```

Out[14]:

	sg	al	sc	hemo	pcv	htn	classification
392	1.020	0.0	1.2	14.8	30	0	1

IBM Watson Studio

Search in your workspaces

Buy

RAGHUNATH N's Account

London

RM

Projects / c\_k\_d / ChronicKidneyDisease

File Edit View Insert Cell Kernel Help

Trusted | Python 3.9

sg

al

sc

hemo

pcv

htn

classification

392	1.020	0.0	1.2	14.8	30	0	1
393	1.025	0.0	0.7	13.0	38	0	1
394	1.020	0.0	0.8	14.1	29	0	1
395	1.020	0.0	0.5	15.7	31	0	1
396	1.025	0.0	1.2	16.5	38	0	1
397	1.020	0.0	0.6	15.8	33	0	1
398	1.025	0.0	1.0	14.2	35	0	1
399	1.025	0.0	1.1	15.8	37	0	1

In [15]: # Split the data into independent (x) dataset (the features) and dependent (Y) dataset (the target)  
X = df.drop(['classification'],axis=1)  
Y = df['classification']

In [16]: X.columns  
  
Out[16]: Index(['sg', 'al', 'sc', 'hemo', 'pcv', 'htn'], dtype='object')

In [17]: x\_scaler = MinMaxScaler()  
x\_scaler.fit(X)  
column\_names = X.columns  
X[column\_names] = x\_scaler.transform(X)  
X[column\_names]

Out[17]:

	sg	al	sc	hemo	pcv	htn
0	0.75	0.2	0.033898	0.836735	0.717949	1.0
1	0.75	0.8	0.016949	0.557823	0.564103	0.0
2	0.25	0.4	0.059322	0.442177	0.384615	0.0
3	0.00	0.8	0.144068	0.551020	0.410256	1.0
4	0.25	0.4	0.042373	0.578731	0.487179	0.0

IBM Watson Studio

Search in your workspaces

Buy

RAGHUNATH N's Account

London

RM

Projects / c\_k\_d / ChronicKidneyDisease

File Edit View Insert Cell Kernel Help

Trusted | Python 3.9

In [18]: X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,test\_size=0.2,shuffle=True)

In [19]: def models(X\_train,Y\_train):  
# Logistic Regression  
  
from sklearn.linear\_model import LogisticRegression  
log = LogisticRegression(random\_state=0)  
log.fit(X\_train,Y\_train)  
  
# Decision Tree Classifier  
  
from sklearn.tree import DecisionTreeClassifier  
tree = DecisionTreeClassifier(criterion = "entropy" , random\_state=0)  
tree.fit(X\_train,Y\_train)  
  
# Random Forest Classifier  
  
from sklearn.ensemble import RandomForestClassifier  
forest = RandomForestClassifier(n\_estimators = 10 , criterion = "entropy",random\_state=0)  
forest.fit(X\_train,Y\_train)  
  
# Print the models accuracy on the training data  
  
print('[0] Logistic Regression Training Accuracy : ',log.score(X\_train,Y\_train))  
print('[1] Decision Tree Classifier Training Accuracy : ',tree.score(X\_train,Y\_train))  
print('[2] Random Forest Classifier Training Accuracy : ',forest.score(X\_train,Y\_train))  
  
return log,tree,forest

In [20]: model = models(X\_train,Y\_train)  
  
[0] Logistic Regression Training Accuracy : 0.9781659388646288  
[1] Decision Tree Classifier Training Accuracy : 1.0  
[2] Random Forest Classifier Training Accuracy : 1.0

IBM Watson Studio

Search in your workspaces

Buy

RAGHUNATH N's Account

London

RB

Projects / c\_k\_d / ChronicKidneyDisease

File Edit View Insert Cell Kernel Help

Trusted | Python 3.9

In [24]:

```
def SaveModel():
    import pickle
    with open("CKD_Model", "wb") as f:
        pickle.dump(model[1], f)
    SaveModel()
import pickle
with open("CKD_Model", "rb") as f:
    randomForest = pickle.load(f)

pred = randomForest.predict(X_test)
print(pred)
print()
print(Y_test)

[1 1 0 1 1 0 0 1 1 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 1 0 1 0 0 1 1 0 0 0 0 1 0
 1 1 1 1 0 0 0 1 0 0 1 1 1 1 1 1 0 1 1 0 0]

267 1
331 1
74 0
320 1
254 1
163 0
24 0
337 1
251 1
54 0
14 0
49 0
182 0
304 1
366 1
266 1
15 0
47 0
137 0
144 0
213 0
318 1
170 0
```