

Early Prediction for Chronic Kidney Disease Detection: A Progressive Approach to Health Management

TEAM NO : 3

TEAM MEMBERS' : L SNEHA

NAME M SUJITHA

S KARTHIKA LAKSHMI

V VIGNESH

CLASS : III yr BSC COMPUTER SCIENCE

TABLE OF INDEX

S.NO	DESCRIPTION	PAGE NO
1.	Introduction	3
2.	Problem Solving & Design Thinking	4
3.	Result	8
4.	Advantage and Disadvantage	9
5.	Application	10
6.	Conclusion	11
7.	Future Scope	12
8.	Appendix	14

1. INTRODUCTION

Project Description

Chronic Kidney Disease (CKD) is a major medical problem and can be cured if treated in the early stages. Usually, people are not aware that medical tests we take for different purposes could contain valuable information concerning kidney diseases. Consequently, attributes of various medical tests are investigated to distinguish which attributes may contain helpful information about the disease. The information says that it helps us to measure the severity of the problem, the predicted survival of the patient after the illness, the pattern of the disease and work for curing the disease.

In today's world as we know most of the people are facing so many diseases and as this can be cured if we treat people in early stages this project can use a pretrained model to predict the Chronic Kidney Disease which can help in treatments of people who are suffering from this disease.

IDEATION & BRAINSTROMING MAP

Brainstorm & Idea Prioritization Template:


Under this activity our team member have gathered and discussed various ideas to solve our project problem. Each member contributed 6 to 10 ideas.

After gathering all ideas we have assessed the impact and feasibility of each point.

Reference: <https://www.mural.co/templates/empathy-map-canvas>

Step-1: Team Gathering, Collaboration and Select the Problem Statement

Template



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

⌚ 10 minutes to prepare
🕒 1 hour to collaborate
👥 2-6 people recommended

➔

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

⌚ 10 minutes

A

Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

C

Learn how to use the facilitation tools

Use the facilitation Superpowers to run a happy and productive session.

Open article ➔

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

⌚ 5 minutes

How might we Early Prediction of chronic Kidney disease Detection?

Key rules of brainstorming

To run a smooth and productive session

Stay in topic

Encourage wild ideas

Defer judgment

Listen to others

Go for volume

If possible, be visual

2

Brainstorm

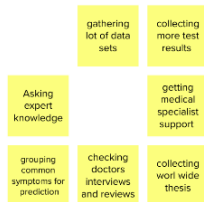
Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

SUJITHA M



SNEHA L



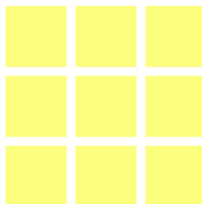
KARTHIKALAKSHMI S



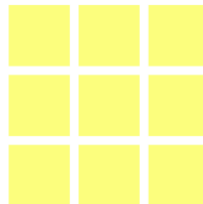
VIGNESH V



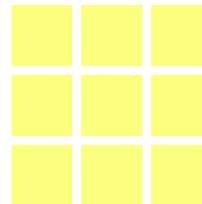
Person 5



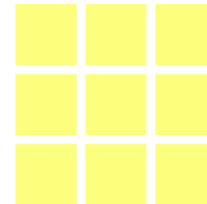
Person 6



Person 7



Person 8

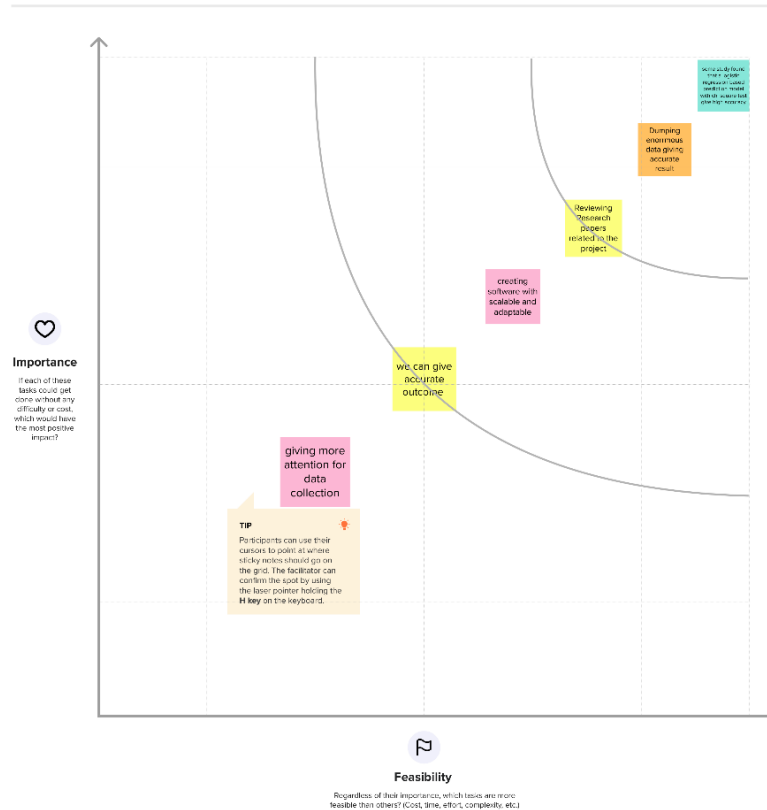


4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes



→

After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

- A Share the mural**
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.
- B Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

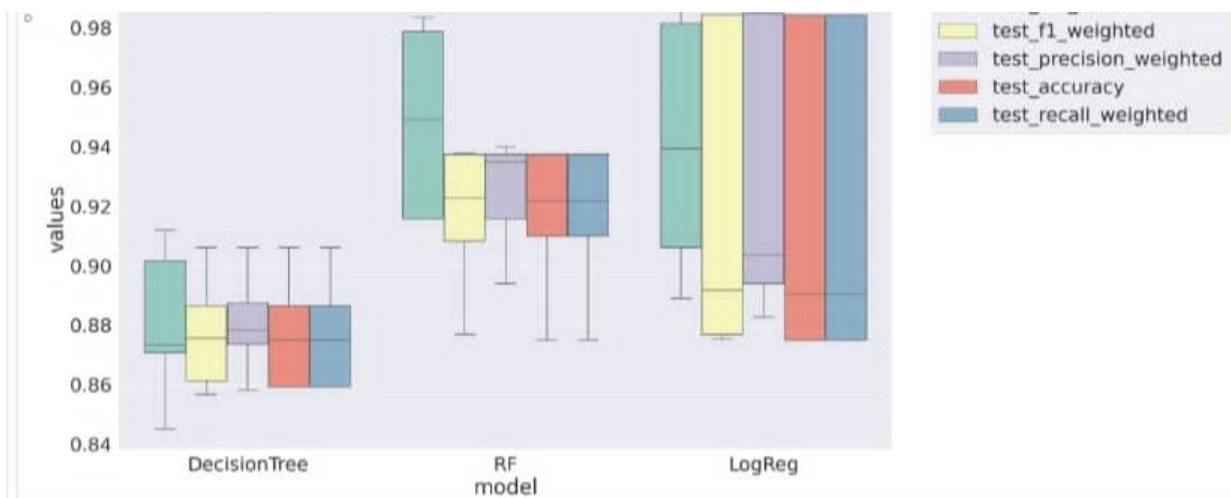
Keep moving forward

- Strategy blueprint**
Define the components of a new idea or strategy.
[Open the template →](#)
- Customer experience journey map**
Understand customer needs, motivations, and obstacles for an experience.
[Open the template →](#)
- Strengths, weaknesses, opportunities & threats**
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
[Open the template →](#)

[Share template feedback](#)

3. RESULT

LogReg				
	precision	recall	f1-score	support
NO CKD	1.00	0.89	0.94	54
CKD	0.81	1.00	0.90	26
accuracy			0.93	80
macro avg	0.91	0.94	0.92	80
weighted avg	0.94	0.93	0.93	80
RF				
	precision	recall	f1-score	support
NO CKD	0.96	0.96	0.96	54
CKD	0.92	0.92	0.92	26
accuracy			0.95	80
macro avg	0.94	0.94	0.94	80
weighted avg	0.95	0.95	0.95	80
DecisionTree				
	precision	recall	f1-score	support
NO CKD	0.93	0.94	0.94	54
CKD	0.88	0.85	0.86	26
accuracy			0.91	80
macro avg	0.90	0.90	0.90	80
weighted avg	0.91	0.91	0.91	80



4. ADVANTAGES

- **Early detection:** The proposed solution for chronic kidney disease prediction can help in the early detection of the disease, which can lead to timely treatment and better management of the disease.
- **Accurate diagnosis:** By using machine learning algorithms, the proposed solution can provide accurate predictions of chronic kidney disease, which can help healthcare professionals to provide targeted treatment plans to patients.
- **Cost-effective:** The solution can be cost-effective as it can reduce the need for expensive diagnostic tests and hospitalization.
- **Saves time:** The solution can save time for healthcare professionals, as they can quickly obtain predictions from the system, which can reduce the time it takes for diagnosis.
- **Easy to use:** The solution can be user-friendly and easy to use for healthcare professionals, which can increase its adoption in clinical settings.

DISADVANTAGE

- **Limited data:** The accuracy of the predictions may be limited by the availability and quality of data used to train the machine learning algorithms.
- **False positives:** The solution may generate false positives, leading to unnecessary treatment or interventions.
- **Technical issues:** Technical issues such as system crashes, bugs or errors could affect the effectiveness of the solution.
- **Lack of human touch:** The solution may lack the human touch that is required in healthcare, leading to a decrease in patient satisfaction.
- **Limited scope:** The solution may only be applicable to chronic kidney disease prediction and not other diseases, which limits its scope.

5. APPLICATIONS

- **Healthcare:** Healthcare professionals can use this prediction model to identify patients who are at risk of developing chronic kidney disease. This can help them to provide targeted treatment plans to patients and improve their overall health outcomes.
- **Clinical Research:** The prediction model can be used to identify patients who are most likely to benefit from specific treatment plans or clinical trials. This can help researchers to design more effective and efficient clinical studies.
- **Personal Health Monitoring:** Individuals can use the prediction model to monitor their own health status and take preventive measures to reduce the risk of developing chronic kidney disease.
- **Insurance:** Insurance companies can use the prediction model to assess the risk of chronic kidney disease in their policyholders. This can help them to design policies that are more tailored to the needs of their customers.
- **Public Health:** Government agencies and public health organizations can use the prediction model to identify populations that are at risk of chronic kidney disease. This can help them to design and implement targeted public health interventions to reduce the burden of the disease on the population.

6. CONCLUSION

According to the findings of the project, the Random Forest approach and logistic regression can be used to predict chronic kidney disease more accurately. According to the project, their precision was 96 percent, and their accuracy was 95 percent. This will aid in the achievement of improved outcomes as well as the accuracy and efficiency with which healthcare practitioners can anticipate kidney issues. This will enhance the dependability of the framework as well as the framework's presentation. The hope is that it would encourage people to seek early treatment for chronic renal disease and to make improvements in their lives.

7. FUTURE SCOPE

- Chronic kidney disease (CKD) is a prevalent and serious health problem worldwide, with increasing incidence and prevalence. The prediction of CKD is crucial for early diagnosis and intervention, which can prevent or delay the progression of CKD and reduce the risk of complications, such as cardiovascular disease, end-stage renal disease, and mortality.
- There are several approaches to predict CKD, including clinical risk factors, laboratory biomarkers, imaging techniques, and machine learning algorithms. Clinical risk factors such as age, gender, race, hypertension, diabetes, obesity, smoking, and family history are commonly used to predict CKD. Laboratory biomarkers such as serum creatinine, estimated glomerular filtration rate (eGFR), albuminuria, and urinary biomarkers are also useful in predicting CKD.
- Imaging techniques such as ultrasound, computed tomography (CT), and magnetic resonance imaging (MRI) can provide information on kidney structure and function, which can be used to predict CKD. Machine learning algorithms such as artificial neural networks, decision trees, and support vector machines have been developed to predict CKD based on multiple variables and complex interactions.
- The future of CKD prediction is promising with the advancement of technology and the availability of large-scale data sets. The integration of multiple approaches and the development of personalized prediction models can improve the accuracy and efficiency of CKD prediction. The use of wearable devices, mobile applications, and telemedicine can also facilitate the monitoring and management of CKD.

- The scope of CKD prediction is not limited to diagnosis but also extends to risk stratification, prognosis, and treatment response prediction. The identification of high-risk individuals and the implementation of preventive strategies can reduce the burden of CKD on individuals and society. The prediction of CKD progression and complications can guide clinical decision-making and improve patient outcomes. The prediction of treatment response can optimize therapy and reduce adverse effects.
- In conclusion, CKD prediction is a critical component of CKD management, which can improve early detection, risk stratification, prognosis, and treatment response. The integration of multiple approaches and the development of personalized prediction models can enhance the accuracy and efficiency of CKD prediction. The future of CKD prediction is promising with the advancement of technology and the availability of large-scale data sets.

8. APPENDIX

#Importing the libraries

```
import pandas as pd
import numpy as np
from collections import Counter as c
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
import pickle
import warnings
warnings.filterwarnings('ignore')
```

Read the Dataset

```
data=pd.read_csv("/content/kidney_disease.csv")
data.head(n=500)
```

Rename the columns

```
data.columns
```

Handling missing values

```
data.info()
data.isnull().sum()
```

```
data['age'].fillna(data['age'].mean(), inplace=True)
data['blood_pressure'].fillna(data['blood_pressure'].mean(), inplace=True)
data['specific_gravity'].fillna(data['specific_gravity'].mean(), inplace=True)
data['albumin'].fillna(data['albumin'].mean(), inplace=True)
data['sugar'].fillna(data['sugar'].mean(), inplace=True)
data['red_blood_cell'].fillna(data['red_blood_cell'].mode()[0], inplace=True)
data['sugar'].fillna(data['sugar'].mode()[0], inplace=True)
data['pus_cell'].fillna(data['pus_cell'].mode()[0], inplace=True)
data['pus_cell_clumps'].fillna(data['pus_cell_clumps'].mode()[0], inplace=True)
data['bacteria'].fillna(data['bacteria'].mode()[0], inplace=True)
```

```

data['blood_glucose_random'].fillna(data['blood_glucose_random'].mean(),inplace=True)
data['blood_urea'].fillna(data['blood_urea'].mean(),inplace=True)
data['serum_creatinine'].fillna(data['serum_creatinine'].mean(),inplace=True)
data['sodium'].fillna(data['sodium'].mean(),inplace=True)
data['potassium'].fillna(data['potassium'].mean(),inplace=True)
data['hemoglobin'].fillna(data['hemoglobin'].mean(),inplace=True)
data['packed_cell_volume'].fillna(data['packed_cell_volume'].mode()[0],inplace=True)
data['red_blood_cell_count'].fillna(data['red_blood_cell_count'].mode()[0],inplace=True)
data['white_blood_cell_count'].fillna(data['white_blood_cell_count'].mode()[0],inplace=True)
data['hypertension'].fillna(data['hypertension'].mode()[0],inplace=True)
data['diabetesmellitus'].fillna(data['diabetesmellitus'].mode()[0],inplace=True)
data['coronary_artery_disease'].fillna(data['coronary_artery_disease'].mode()[0],inplace=True)
data['appetite'].fillna(data['appetite'].mode()[0],inplace=True)
data['pedal_edema'].fillna(data['pedal_edema'].mode()[0],inplace=True)
data['anemia'].fillna(data['anemia'].mode()[0],inplace=True)

```

Let's now check the count of null values after filling all null values using **isnull.sum()**

```
data.isnull().sum()
```

Handling Categorical columns

```

catcols=set(data.dtypes[data.dtypes=='O'].index.values) #selecting non
numerical columns in the dataset
print(catcols)

```

```

for i in catcols:
print("columns:",i)
print(c(data[i])) # grouping the data in column
print('*'*100+'\n')

```

```

data['coronary_artery_disease']=data.coronary_artery_disease.replace('\tno','no')
data['diabetesmellitus']=data.diabetesmellitus.replace('\tno','no')
data['diabetesmellitus']=data.diabetesmellitus.replace('\tyes','yes')
data['diabetesmellitus']=data.diabetesmellitus.replace(' yes','yes')
data['classification']=data.classification.replace('ckd\t','ckd')

```

Label Encoding for categorical columns

```
# label encoding = convert categorical to numerical eg: yes - 1, no- 0
catcols = ['pus_cell', 'hypertension', 'diabetesmellitus', 'anemia',
'coronary_artery_disease', 'red_blood_cell_count', 'appetite',
'packed_cell_volume', 'classification', 'bacteria',
'white_blood_cell_count', 'pus_cell_clumps', 'red_blood_cell',
'pedal_edema']
from sklearn.preprocessing import LabelEncoder
for i in catcols:
print("LABEL ENCODING OF:",i)
LEi= LabelEncoder()
print(c(data[i]))
data[i] = LEi.fit_transform(data[i])
print(c(data[i]))
print('*'*100)
```

Handling Numerical columns

```
contcols=set(data.dtypes[data.dtypes!='O'].index.values)
print(contcols)

for i in contcols:
print("Continuous Column :",i)
print(c(data[i]))
print('*'*120+'\n')
```

Descriptive statistical Analysis

```
data.describe()
```

Univariate analysis

```
sns.distplot(data.diabetesmellitus)
```

Bivariate analysis

```
import matplotlib.pyplot as plt
fig=plt.figure(figsize=(5,5))
plt.scatter(data['age'], data['diabetesmellitus'], color='red')
plt.xlabel('age')
plt.ylabel('diabetesmellitus')
plt.title("age vs diabetesmellitus")
```


Multivariate analysis

Age vs all continuous columns

```
plt.figure(figsize=(20,15), facecolor='white')
plotnumber = 1
for column in catcols:
    if plotnumber<=11:
        ax=plt.subplot(3,4,plotnumber)
        plt.scatter(data['age'],data[column])
        plt.xlabel(column, fontsize=20)
        plotnumber+=1
plt.show()
```

Finding correlation between the independent Columns

```
f,ax=plt.subplots(figsize=(28,20))
sns.heatmap(data.corr(),annot=True,fmt=".2f",ax=ax,linewidths=0.7,linec
olor="pink")
plt.xticks(rotation=45)
plt.yticks(rotation=45)
plt.show()
```

```
sns.countplot(x='classification',data=data)
```

Scaling the Data

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_bal=sc.fit_transform(data)
```

Separate independent and dependent variable

```
selcols=['red_blood_cell','pus_cell','blood_glucose_random','blood_urea','
pedal_edema','anemia','diabettesmellitus','coronary_artery_disease']
x=pd.DataFrame(data,columns=selcols)
y=pd.DataFrame(data,columns=['classification'])
print(x.shape)
print(y.shape)
```

Splitting data into train and test

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,
random_state=2)
```

ANN Model

```
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
classification = Sequential()
classification.add(Dense(30,activation='relu'))
classification.add(Dense(128,activation='relu'))
classification.add(Dense(64,activation='relu'))
classification.add(Dense(32,activation='relu'))
classification.add(Dense(1,activation='sigmoid'))
classification.compile(optimizer='adam',loss='binary_crossentropy',metrics
=['accuracy'])
classification.fit(x_train,y_train,batch_size=10,validation_split=0.2,epoc
hs=100)
```

Random Forest model

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(n_estimators=10, criterion='entropy')
rfc.fit(x_train,y_train)

y_predict = rfc.predict(x_test)
y_predict_train=rfc.predict(x_train)
```

Decision tree model

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier(max_depth=4,
splitter='best',criterion='entropy')
dtc.fit(x_train, y_train)

y_predict=dtc.predict(x_test)
y_predict

y_predict_train=dtc.predict(x_train)
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
lgr=LogisticRegression()
lgr.fit(x_train,y_train)

from sklearn.metrics import accuracy_score,classification_report
y_pred=lgr.predict(x_test)
```

Testing the model

```
y_pred=lgr.predict([[1,1,121.000000,36.0,0,0,1,0]])
print(y_pred)
(y_pred)

y_pred=dtc.predict([[1,1,121.000000,36.0,0,0,1,0]])
print(y_pred)
(y_pred)

y_pred=rfc.predict([[1,1,121.000000,36.0,0,0,1,0]])
print(y_pred)
(y_pred)
```

In ANN we first have to save the model to the test the inputs

```
classification.save("ckd.h5")
y_pred=classification.predict(x_test)
y_pred

y_pred=(y_pred>0.5)
y_pred

def predict_exit(sample_value):
    sample_value=np.array(sample_value)
    sample_value=sample_value.reshape(1,-1)
    sample_value=sc.transform(sample_value)
    return classifier.predict(sample_value)
test=classification.predict([[1,1,121.000000,36.0,0,0,1,0]])
if test==1:
    print('Prediction: High chance of CKD!')
else:
    print('Prediction: Low chance of CKD. ')
```

Testing model with multiple evaluation metrics

```
from sklearn import model_selection
dfs = []
models =[
    ('LogReg', LogisticRegression()),
    ('RF', RandomForestClassifier()),
    ('DecisionTree', DecisionTreeClassifier()),
]
results = []
names = []
scoring=['accuracy','precision_weighted','recall_weighted',
        'roc_auc','f1_weighted']
target_names = ['NO CKD', 'CKD']
for name, model in models:
    kfold = model_selection.KFold(n_splits=5, shuffle=True,
    random_state=90210)
    cv_results = model_selection.cross_validate(model, x_train,
    y_train, cv=kfold, scoring=scoring)
    clf = model.fit(x_train, y_train)
    y_pred = clf.predict(x_test)
    print(name)
    print(classification_report(y_test, y_pred,
    target_names=target_names))
    results.append(cv_results)
    names.append(name)
    this_df = pd.DataFrame(cv_results)
    this_df['model'] = name
    dfs.append(this_df)
    final=pd.concat(dfs,ignore_index=True)
    print(final)

from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_predict)
cm

plt.figure(figsize=(8,6))
sns.heatmap(cm,cmap='Blues',annot=True,
xticklabels=['nckd','ckd'],yticklabels=['nckd','ckd'])
plt.xlabel('Predicted values')
plt.ylabel('Actual values')
plt.title('confusion Matrix for Logistic Regression model')
plt.show()
```

```

from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
cm

plt.figure(figsize=(8,6))
sns.heatmap(cm,cmap='Blues',annot=True,
xticklabels=['nockd','ckd'],yticklabels=['nockd','ckd'])
plt.xlabel('Predicted values')
plt.ylabel('Actual values')
plt.title('confusion Matrix for ANN model')
plt.show()

```

FOR ANN

```
print(classification_report(y_test,y_pred))
```

Evaluate the results

```

bootstraps=[]
for model in list(set(final.model.values)):
    model_df = final.loc[final.model == model]
    bootstrap = model_df.sample(n=30, replace=True)
    bootstraps.append(bootstrap)
    bootstrap_df=pd.concat(bootstraps, ignore_index=True)
    results_long = pd.melt(bootstrap_df,id_vars=["model"],var_name="metrics",
    value_name="values")
    time_metrics=['fit_time','score_time']
    results_long_nofit =
    results_long.loc[~results_long['metrics'].isin(time_metrics)]
    results_long_nofit = results_long_nofit.sort_values(by='values')
    results_long_fit=results_long.loc[results_long['metrics'].isin(time_metric
s)]
    results_long_fit = results_long_fit.sort_values(by="values")
plt.figure(figsize=(20,12))
sns.set(font_scale=2.5)
g=sns.boxplot(x="model", y="values",hue="metrics",data=results_long_nofit,
palette='Set3')
plt.legend(bbox_to_anchor=(1.05,1), loc=2, borderaxespad=0.)
plt.title('Comparison of Model by Classification Metric')
plt.savefig('./benchmark_models_performance.png',dpi=300)

```