**PROJECT DEVELOPMENT PHASE**

**Sprint 1**

|  |  |
| --- | --- |
| **date** | 18 november2022 |
| **Team id** | PNT2022TMID47075 |
| **Project name** | University admit eligibility predictor |

import numpy as np

import pandas as pd

import pickle

import warnings

# Loading the dataset

df = pd.read\_csv('./Data/Admission\_Predict.csv')

def clean\_dataset(df):

assert isinstance(df, pd.DataFrame), "df needs to be a pd.DataFrame"

df.dropna(inplace=True)

indices\_to\_keep = ~df.isin([np.nan, np.inf, -np.inf]).any(1)

return df[indices\_to\_keep].astype(np.float64)

df = clean\_dataset(df)

# Replacing the 0 values from ['Glucose','BloodPressure','SkinThickness','Insulin','BMI'] by NaN

df\_copy = df.copy(deep=True)

df\_copy[['GREScore','TOEFLScore','UniversityRating','SOP','LOR ','CGPA','Research','ChanceofAdmit ']] = df\_copy[['GREScore','TOEFLScore','UniversityRating','SOP','LOR ','CGPA','Research','ChanceofAdmit ']].replace(0,np.NaN)

df\_copy['GREScore'].fillna(df\_copy['GREScore'].median(),inplace=True)

df\_copy['TOEFLScore'].fillna(df\_copy['TOEFLScore'].median(),inplace=True)

df\_copy['UniversityRating'].fillna(df\_copy['UniversityRating'].median(),inplace=True)

df\_copy['SOP'].fillna(df\_copy['SOP'].median(),inplace=True)

df\_copy['LOR '].fillna(df\_copy['LOR '].median(),inplace=True)

df\_copy['CGPA'].fillna(df\_copy['CGPA'].median(),inplace=True)

df\_copy['Research'].fillna(df\_copy['Research'].median(),inplace=True)

df\_copy['ChanceofAdmit '].fillna(df\_copy['ChanceofAdmit '].median(),inplace=True)

# Replacing NaN value by mean, median depending upon distribution

X=df\_copy.drop('ChanceofAdmit ',axis=1)

Y=df\_copy['ChanceofAdmit ']

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=42)

X\_train.shape,X\_test.shape,y\_train.shape,y\_test.shape

from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor (n\_estimators =1000, max\_depth = 10, random\_state = 34)

regressor.fit (X\_train, np.ravel(y\_train, order = 'C'))

# Creating a pickle file for the classifier

filename = 'Model/prediction-rfc-model.pkl'

pickle.dump(regressor, open(filename, 'wb'))

filename = 'Model/prediction-rfc-model.pkl'

pickle.dump(regressor, open(filename, 'wb'))

filename = 'Model/prediction-rfc-model.pkl'

classifier = pickle.load(open(filename, 'rb'))

data = np.array([[337,118,4,4.5,4.5,9.65,1]])

my\_prediction = classifier.predict(data)

warnings.filterwarnings("ignore", category=DeprecationWarning)

print(my\_prediction[0])



