Intelligent Admissions :

The Future Of University Decision Making With Machine Learning

• INTRODUCTION

- University admission is the process by which students are selected to attend a college or university. The process typically involves several steps, including submitting an application, taking entrance exams, and participating in interviews or other evaluations.

- **OVERVIEW**

Intelligent admission future University of decision making in machine learning overview

Intelligent admission decision-making in universities using machine learning is a rapidly growing field that aims to automate and improve the admissions process. The process of selecting candidates for admission to universities is often complex and involves several subjective factors, making it challenging to make consistent and unbiased decisions.

Machine learning algorithms can be trained on historical admissions data to learn the patterns and relationships between different application components and the final admission decision. The algorithms can then be used to predict the likelihood of admission for new applicants based on their application components, such as academic records, test scores, extracurricular activities, and personal statements.

Intelligent admission decision-making systems can also be designed to improve the efficiency of the admissions process by automatically screening applications and filtering out those that do not meet the minimum requirements or are unlikely to be admitted. This can save time and resources for admissions officers, allowing them to focus on reviewing the most promising applications.

However, there are also potential challenges and ethical considerations that need to be addressed when implementing intelligent admission decision-making systems. One major concern is the risk of perpetuating bias and discrimination if the algorithms are not designed and implemented properly. It is essential to ensure that the data used to train the algorithms are diverse and representative of the applicant pool, and that the algorithms are regularly audited to detect and address any bases.Overall, intelligent admission decision-making using machine learning has the potential to improve the efficiency and fairness of the admission process in universities, but it requires careful design and implementation to avoid perpetuating bias and discrimination.

- **PURPOSE**

Intelligent admission decision making in universities using machine learning has the potential to streamline and improve the admission process. Here are some ways machine learning can be used for admission decision making:

Predictive modeling: Universities can use historical data on admission decisions and student outcomes to develop predictive models that can identify the characteristics of successful applicants. These models can then be used to evaluate new applications and make data-driven decisions on admission.

Natural language processing: Many universities receive thousands of applications each year, and reading and evaluating each one can be a time-consuming process. Natural language processing (NLP) can be used to analyze application essays and other written materials, identifying patterns and insights that can help admission officers make more informed decisions.

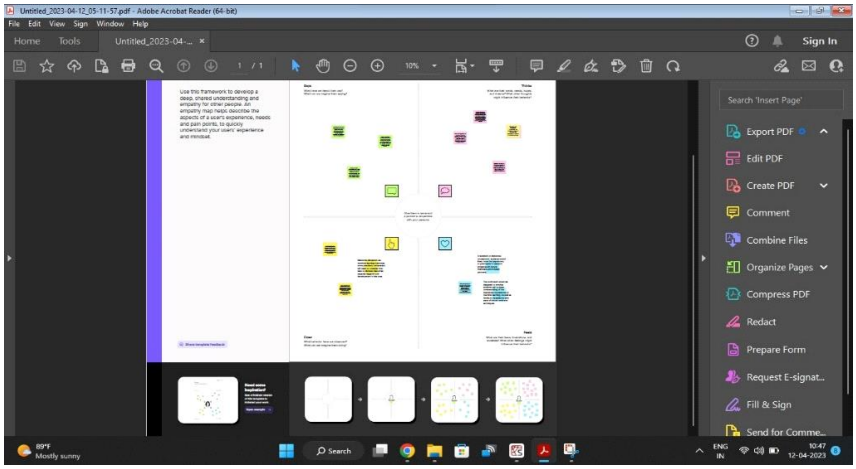
Image recognition: Some universities require applicants to submit videos or images as part of their application. Image recognition algorithms can be used to analyze these submissions, looking for patterns or features that may be associated with success in college.

Personalization: Machine learning can be used to personalize the admission process for each applicant, taking into account their unique background, skills, and experience. This could include recommending specific courses or programs that may be a good fit, or providing personalized feedback on areas where the applicant could improve.

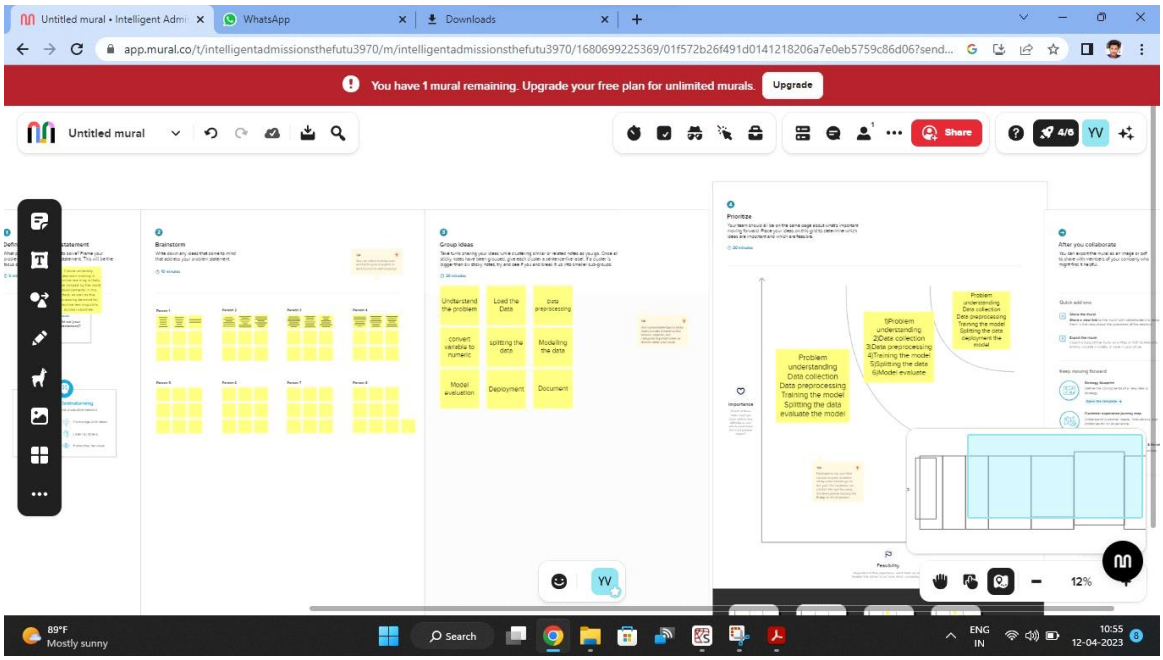
Overall, the use of machine learning in admission decision making has the potential to improve the efficiency and effectiveness of the process, helping universities to identify and admit the best candidates while also reducing bias and ensuring a fair and transparent process. However, it's important to note that any algorithmic decision making system should be developed and deployed with careful consideration of potential biases and ethical concerns.

- **PROBLEM DEFINITION & DESIGN THINKING**

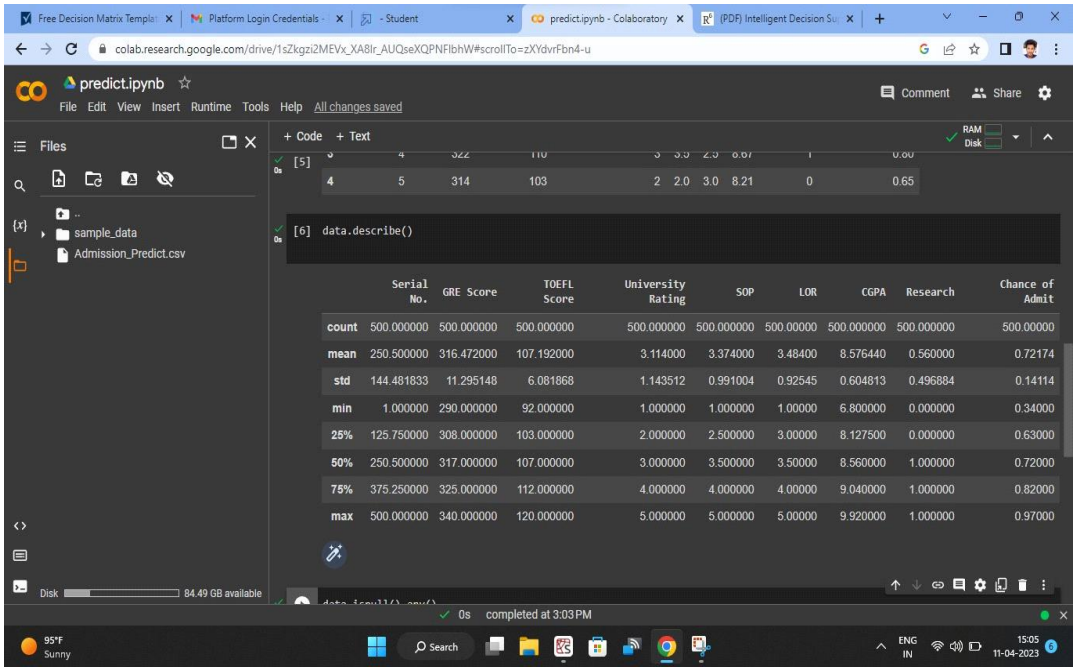
2.1 Empathy Map

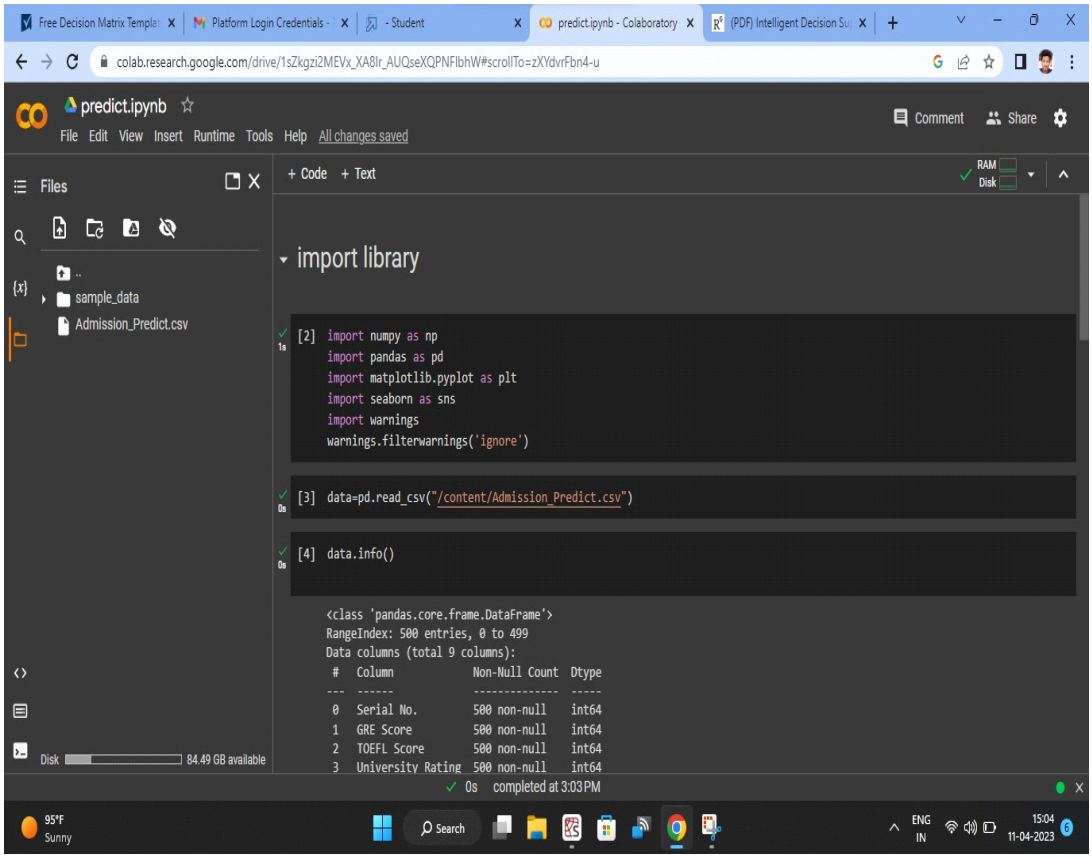


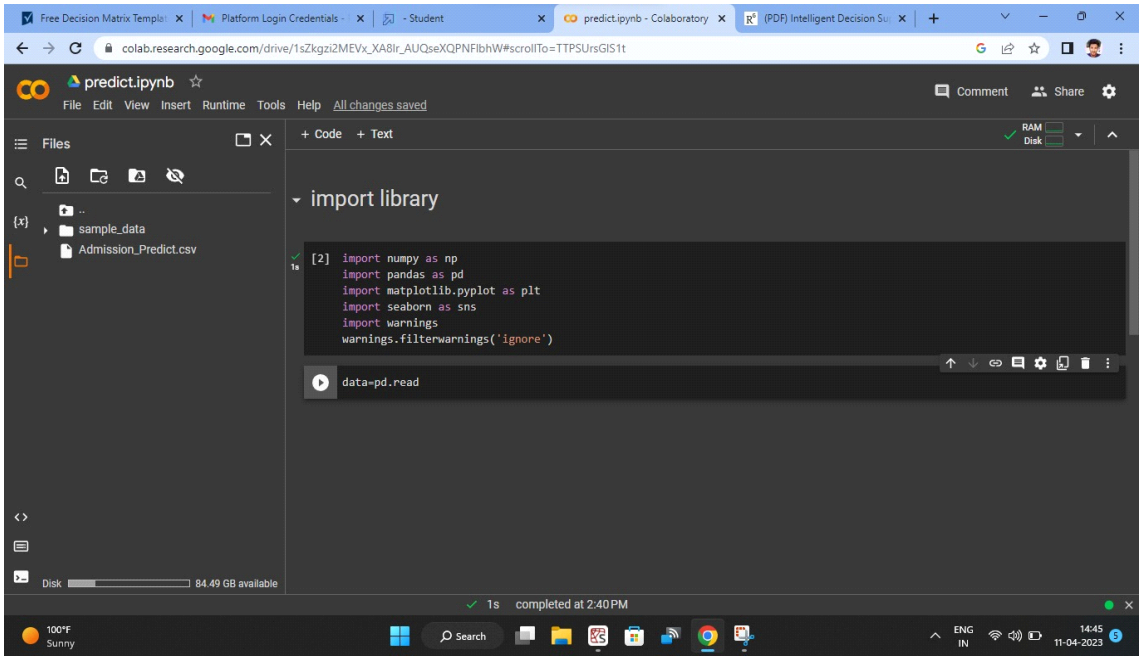
2.2 IDEATION & BRAINSTORMING MAP

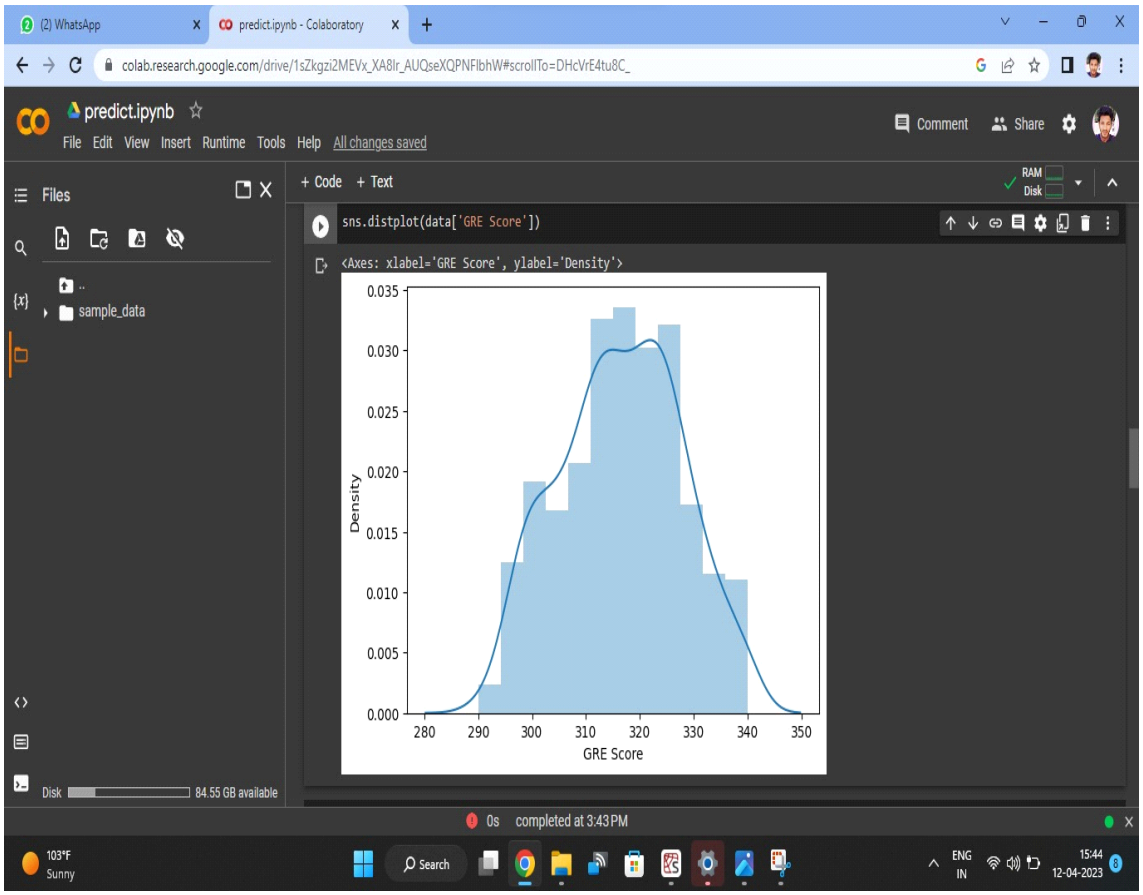

Untitled mural • Intelligent Admi
WhatsApp
Downloads
app.mural.co/t/intelligentadmissionsthefutu3970/m/intelligentadmissionsthefutu3970/1680699225369/01f572b26f491d0141218206a7e0eb5759c86d06?send...
You have 1 mural remaining. Upgrade your free plan for unlimited murals.
Upgrade
Untitled mural
Share
Brainstorm
Group ideas
Prioritize
After you collaborate
Understand the problem
Load the Data
data preprocessing
convert variable to numeric
splitting the data
Modelling the data
Model evaluation
Deployment
Document
Problem understanding
Data collection
Data preprocessing
Training the model
Splitting the data
evaluate the model
89°F
Mostly sunny
Search
ENG
IN
10:55
12-04-2023

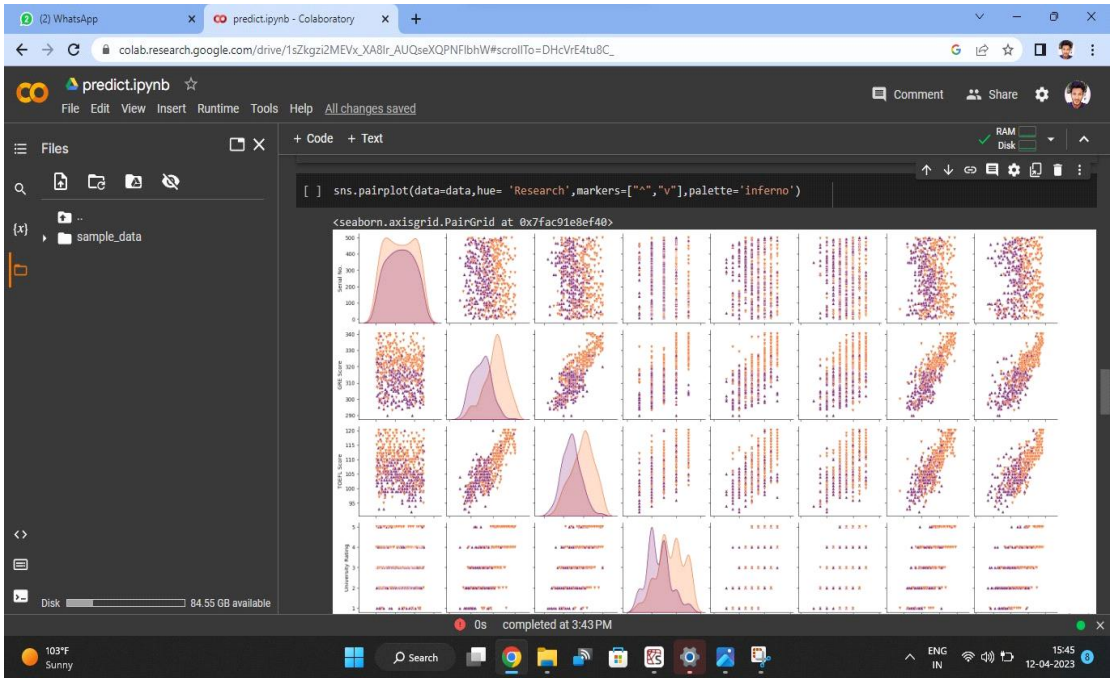
- **RESULT**

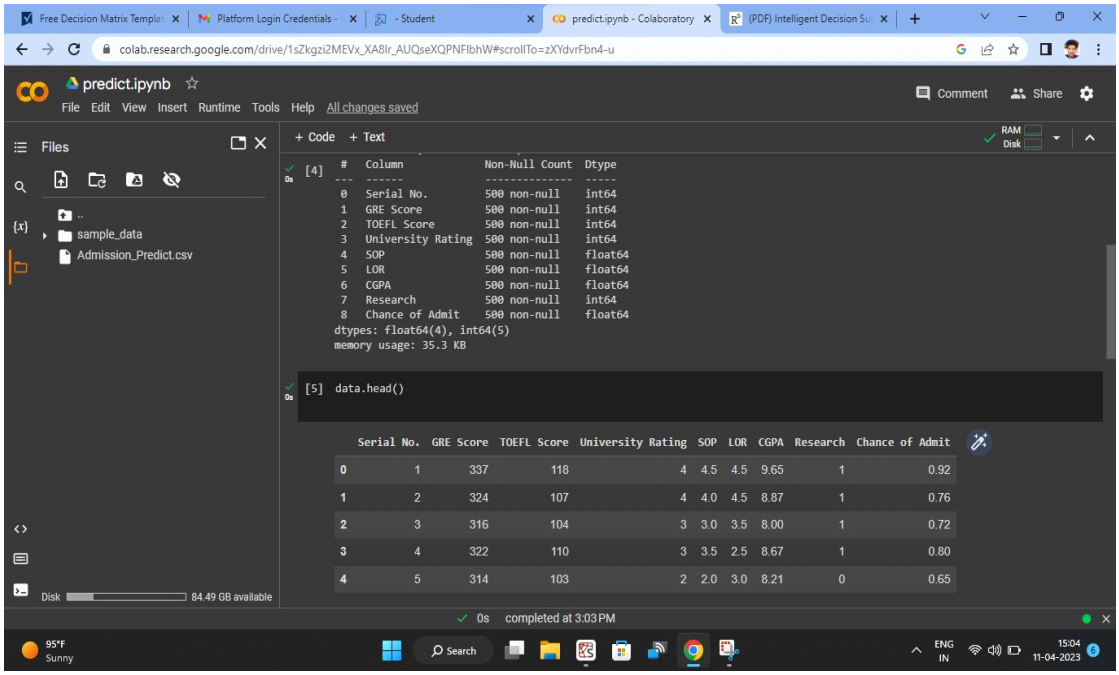

Free Decision Matrix Templat
Platform Login Credentials -
- Student
predict.ipynb - Colaboratory
(PDF) Intelligent Decision Su
colab.research.google.com/drive/1sZkgzi2MEVx_XA8lr_AUQseXQPNFlbhW#scrollTo=zXYdvrFbn4-u
predict.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
sample_data
Admission_Predict.csv
[5]
4 5 314 103 2 2.0 3.0 8.21 0 0.65
[6] data.describe()
Serial No. GRE Score TOEFL Score University Rating SOP LOR CGPA Research Chance of Admit
count 500.000000 500.000000 500.000000 500.000000 500.000000 500.00000 500.000000 500.000000 500.00000
mean 250.500000 316.472000 107.192000 3.114000 3.374000 3.48400 8.576440 0.560000 0.72174
std 144.481833 11.295148 6.081868 1.143512 0.991004 0.92545 0.604813 0.496884 0.14114
min 1.000000 290.000000 92.000000 1.000000 1.000000 1.00000 6.800000 0.000000 0.34000
25% 125.750000 308.000000 103.000000 2.000000 2.500000 3.00000 8.127500 0.000000 0.63000
50% 250.500000 317.000000 107.000000 3.000000 3.500000 3.50000 8.560000 1.000000 0.72000
75% 375.250000 325.000000 112.000000 4.000000 4.000000 4.00000 9.040000 1.000000 0.82000
max 500.000000 340.000000 120.000000 5.000000 5.000000 5.00000 9.920000 1.000000 0.97000
Disk
84.49 GB available
0s completed at 3:03 PM
95°F
Sunny
Search
ENG
IN
15:05
11-04-2023

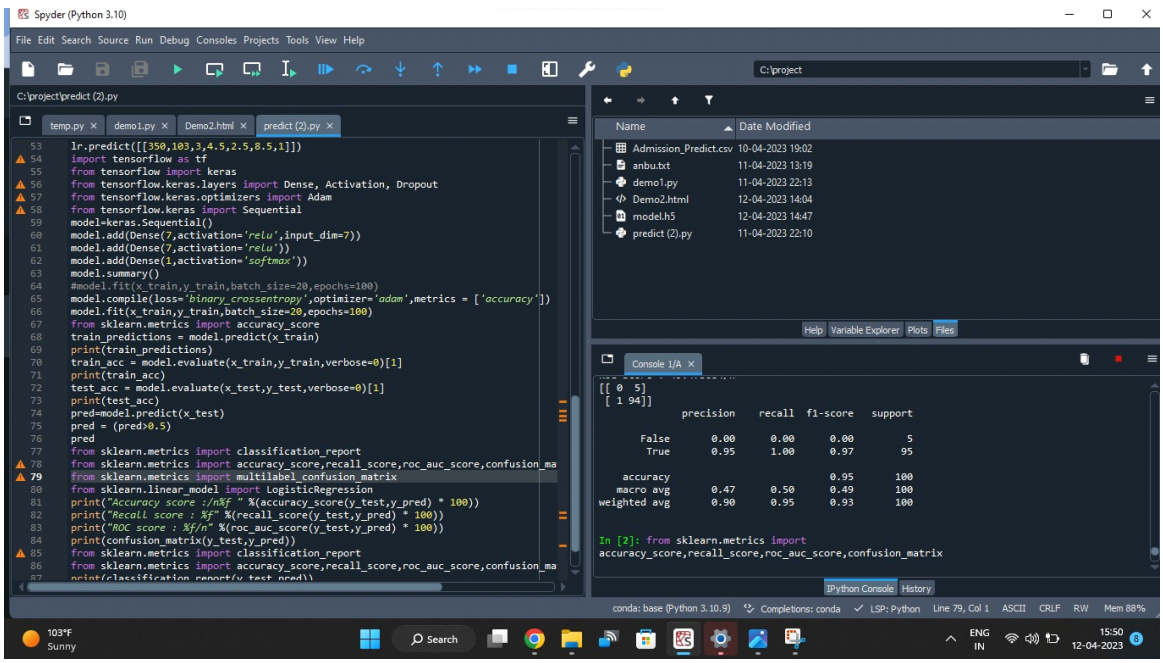

predict.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
sample_data
Admission_Predict.csv
+ Code + Text
RAM
Disk
import library
[2] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
[3] data=pd.read_csv("/content/Admission_Predict.csv")
[4] data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
Column Non-Null Count Dtype
0 Serial No. 500 non-null int64
1 GRE Score 500 non-null int64
2 TOEFL Score 500 non-null int64
3 University Rating 500 non-null int64
Disk 84.49 GB available
0s completed at 3:03 PM
95°F
Sunny
Search
ENG
IN
15:04
11-04-2023


predict.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
sample_data
Admission_Predict.csv
+ Code + Text
RAM
Disk
import library
[2] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
data=pd.read
Disk 84.49 GB available
1s completed at 2:40 PM
100°F
Sunny
Search
ENG
IN
14:45
11-04-2023


predict.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
sns.distplot(data['GRE Score'])
<Axes: xlabel='GRE Score', ylabel='Density'>
Density
GRE Score
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
280
290
300
310
320
330
340
350
sample_data
Disk
84.55 GB available
0s completed at 3:43 PM
103°F
Sunny
Search
ENG
IN
15:44
12-04-2023


predict.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
[ ] sns.pairplot(data=data,hue= 'Research',markers=["^","v"],palette='inferno')
<seaborn.axisgrid.PairGrid at 0x7fac91e8ef40>
sample_data
Disk
84.55 GB available
0s completed at 3:43 PM
103°F
Sunny
Search
ENG
IN
15:45
12-04-2023


predict.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
sample_data
Admission_Predict.csv
+ Code + Text
RAM
Disk
[4]
0 Serial No. 500 non-null int64
1 GRE Score 500 non-null int64
2 TOEFL Score 500 non-null int64
3 University Rating 500 non-null int64
4 SOP 500 non-null float64
5 LOR 500 non-null float64
6 CGPA 500 non-null float64
7 Research 500 non-null int64
8 Chance of Admit 500 non-null float64
dtypes: float64(4), int64(5)
memory usage: 35.3 KB
[5] data.head()
Serial No. GRE Score TOEFL Score University Rating SOP LOR CGPA Research Chance of Admit
1 337 118 4 4.5 4.5 9.65 1 0.92
2 324 107 4 4.0 4.5 8.87 1 0.76
3 316 104 3 3.0 3.5 8.00 1 0.72
4 322 110 3 3.5 2.5 8.67 1 0.80
5 314 103 2 2.0 3.0 8.21 0 0.65
Disk 84.49 GB available
0s completed at 3:03 PM
95°F Sunny
Search
ENG IN
15:04 11-04-2023


Spyder (Python 3.10)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\project\predict (2).py
temp.py demo1.py Demo2.html predict (2).py
lr.predict([[350,103,3,4.5,2.5,8.5,1]])
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import Sequential
model=keras.Sequential()
model.add(Dense(7,activation='relu',input_dim=7))
model.add(Dense(7,activation='relu'))
model.add(Dense(1,activation='softmax'))
model.summary()
#model.fit(x_train,y_train,batch_size=20,epochs=100)
model.compile(loss='binary_crossentropy',optimizer='adam',metrics = ['accuracy'])
model.fit(x_train,y_train,batch_size=20,epochs=100)
from sklearn.metrics import accuracy_score
train_predictions = model.predict(x_train)
print(train_predictions)
train_acc = model.evaluate(x_train,y_train,verbose=0)[1]
print(train_acc)
test_acc = model.evaluate(x_test,y_test,verbose=0)[1]
print(test_acc)
pred=model.predict(x_test)
pred = (pred>0.5)
pred
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_ma
from sklearn.metrics import multilabel_confusion_matrix
from sklearn.linear_model import LogisticRegression
print("Accuracy score :/n%f " %(accuracy_score(y_test,y_pred) * 100))
print("Recall score : %f" %(recall_score(y_test,y_pred) * 100))
print("ROC score : %f/n" %(roc_auc_score(y_test,y_pred) * 100))
print(confusion_matrix(y_test,y_pred))
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_ma
C:\project
Name Date Modified
Admission_Predict.csv 10-04-2023 19:02
anbu.txt 11-04-2023 13:19
demo1.py 11-04-2023 22:13
Demo2.html 12-04-2023 14:04
model.h5 12-04-2023 14:47
predict (2).py 11-04-2023 22:10
Help Variable Explorer Plots Files
Console 1/A
[[0 5]
[1 94]]
precision recall f1-score support
False 0.00 0.00 0.00 5
True 0.95 1.00 0.97 95
accuracy 0.95 100
macro avg 0.47 0.50 0.49 100
weighted avg 0.90 0.95 0.93 100
In [2]: from sklearn.metrics import
accuracy_score,recall_score,roc_auc_score,confusion_matrix
IPython Console History
conda: base (Python 3.10.9) Completions: conda LSP: Python Line 79, Col 1 ASCII CRLF RW Mem 88%
103°F Sunny
Search
ENG IN
15:50 12-04-2023

4. ADVANTAGE & DISADVANTAGE

ADVANTAGE

Machine learning algorithms can quickly process large amounts of data and make accurate predictions about which applicants are most likely to succeed. This can save universities time and resources when reviewing applications.

Objectivity: Machine learning algorithms are unbiased and can make decisions based solely on data, without being influenced by personal biases or prejudices. This can help universities ensure a fair and equitable admissions process.

Personalization: Machine learning algorithms can analyze data on individual applicants and make personalized recommendations based on their strengths and weaknesses. This can help universities identify applicants who are a good fit for their programs and offer them tailored support and resources.

Predictive accuracy: Machine learning algorithms can analyze a wide range of data points, such as academic records, extracurricular activities, and demographic information, to accurately predict which applicants are most likely to succeed in their programs.

Continuous improvement: As machine learning algorithms process more data over time, they can learn from their mistakes and improve their predictive accuracy. This can help universities continuously refine their admissions criteria and identify new factors that contribute to student success.

Overall, using machine learning for intelligent admission decision-making can help universities streamline their admissions process, reduce bias, and make more accurate predictions about which applicants are most likely to succeed in their programs.

DISADVANTAGE

One disadvantage is that machine learning algorithms may reinforce existing biases in the admissions process. If the algorithm is trained on data that contains historical biases, such as underrepresentation of certain groups of people, it may continue to perpetuate these biases in the future. This can lead to unfair treatment of applicants who belong to underrepresented groups.

Another disadvantage is that machine learning algorithms can be opaque and difficult to interpret. The decisions made by the algorithm may not be easily explainable, which can make it challenging to identify and address any issues or errors in the decision-making process. This lack of transparency can also erode trust in the admissions process.

Additionally, there is a risk that the use of machine learning algorithms in admissions may create a "black box" effect, where decisions are made based solely on data and algorithms, without any human involvement. This can lead to a lack of accountability and responsibility, as well as a potential loss of the personal touch in the admissions process.

Finally, it's important to note that machine learning algorithms are only as good as the data they are trained on. If the data is incomplete, inaccurate, or biased, the algorithm's decisions may not be reliable or fair. This highlights the need for careful data collection and curation, as well as ongoing monitoring and evaluation of the algorithm's performance.

5. APPILICATIONS

Intelligent admission decision-making in universities using machine learning applications is an emerging field that has the potential to revolutionize the university admissions process. Machine learning algorithms can analyze vast amounts of data to identify patterns and make predictions, which can help universities make better and more objective admission decisions.

One potential application of machine learning in university admissions is in the analysis of student performance data. By analyzing the academic records of past students, machine learning algorithms can identify the factors that are most strongly correlated with success in a particular program. This information can be used to develop predictive models that can help universities identify students who are most likely to succeed in their programs.

Another potential application of machine learning in university admissions is in the analysis of applicant data. By analyzing data such as test scores, essays, and letters of recommendation, machine learning algorithms can identify patterns that are indicative of a student's potential for success. This information can be used to develop predictive models that can help universities identify the most promising applicants.

Additionally, machine learning algorithms can be used to identify bias in the admissions process. By analyzing data on past admissions decisions, machine learning algorithms can identify patterns of bias that may be present in the process. This information can be used to develop more objective and fair admission policies.

However, it is important to note that machine learning algorithms are only as good as the data that is used to train them. If the data used to train the algorithms is biased or incomplete, the resulting models will also be biased or incomplete. Therefore, it is crucial to ensure that the data used to train machine learning algorithms is representative and unbiased.

In summary, intelligent admission decision-making in universities using machine learning applications has the potential to improve the admissions process by identifying patterns and making predictions based on vast amounts of data. However, it is important to ensure that the data used to train the algorithms is unbiased and representative to ensure fair and objective admission decisions.

- ## CONCLUSION

The future of university admissions decision-making is likely to be heavily influenced by machine learning and other intelligent technologies. These technologies can help universities to more effectively evaluate and select the most promising applicants from a large pool of candidates.

Machine learning algorithms can be trained on large datasets of historical admissions data, allowing them to identify patterns and trends in applicant qualifications, characteristics, and performance. This information can then be used to develop predictive models that can accurately assess the likelihood of success for different applicants.

By leveraging these predictive models, universities can make more informed decisions about which applicants to admit, and which to reject. This can help to ensure that the most qualified and promising students are selected for admission, while also minimizing the risk of admitting students who may not perform well academically.

Overall, the use of machine learning in university admissions decision-making has the potential to greatly improve the efficiency and effectiveness of the admissions process, while also helping to ensure that universities are able to identify and select the most promising students for admission.

- ## FUTURE SCOPE

Intelligent admission decision-making in universities is an exciting field that can greatly benefit from the use of machine learning. With the increasing amount of data available, machine learning algorithms can analyze and learn from past admission data to help make more informed decisions about which students to accept.

Some potential future applications of machine learning in university admissions could include:

Predictive modeling: Machine learning algorithms can be used to predict the likelihood of a student's success based on a variety of factors such as their academic history, test scores, extracurricular activities, and personal characteristics. This could help universities make more accurate predictions about which students are most likely to thrive at their institution.

Applicant screening: Machine learning algorithms can be used to automatically screen applications and identify candidates who meet specific criteria. For example, an algorithm could be trained to identify students who have a strong academic background, relevant work experience, or specific skills that are required for a particular program.

Personalization: Machine learning algorithms can be used to personalize the admissions process for individual students. For example, an algorithm could analyze a student's academic history and extracurricular activities to identify areas where they could improve their application, and provide personalized feedback on how to do so.

Diversity and inclusion: Machine learning algorithms can be used to identify biases in the admissions process and help universities ensure that they are admitting a diverse group of students. For example, an algorithm could be trained to identify patterns of bias in the admissions process, such as underrepresentation of certain demographic groups, and suggest ways to address these issues.

Overall, the future scope of machine learning in university admissions is promising, and has the potential to greatly improve the fairness and accuracy of the admissions process. However, it is important to ensure that these algorithms are used ethically and transparently, and that the decisions they make are not biased or discriminatory.

- **APPENDIX**

Source Code :

predict.ipynp :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
data=pd.read_csv('/project/Admission_Predict.csv')
data.head()
```

```
data.info()

data.describe()

data.isnull().any()

sns.distplot(data['GRE Score'])

sns.pairplot(data=data,hue= 'Research',markers=["^","v"],palette='inferno')

sns.scatterplot(x='University Rating',y='CGPA',data=data,color='black',s=100)

category=['GRE Score','TOEFL Score','University Rating','SOP','LOR','CGPA','Research','Chance of Admit']

color=['yellowgreen','gold','lightskyblue','pink','red','purple','orange','gray']

start= True

i=0

i in np.arange(4)

fig = plt.figure(figsize=(14,8))

plt.subplot2grid((4,2),(i,0))

data[category[2*i]].hist(color=color[2*i],bins=10)

plt.title(category[2*i])

plt.subplot2grid((4,2),(i,1))

data[category[2*i+1]].hist(color=color[2*i+1],bins=10)

plt.title(category[2*i+1])

plt.subplots_adjust(hspace =0.7, wspace =0.2)

plt.show()

from sklearn.preprocessing import MinMaxScaler

sc=MinMaxScaler()

x=data.iloc[:,0:7].values

x

y=data.iloc[:,8].values

y

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.20,random_state=101)
```

```
y_train=(y_train>0.5)
y_train
y_test=(y_test>0.5)
y_test
k=y_test.astype(int)
print("shape of independent training data is{}".format(x_train.shape))
print("shape of dependent training data is{}".format(y_train.shape))
from sklearn.linear_model import LogisticRegression
cls=LogisticRegression()
lr=cls.fit(x_train, y_train)
y_pred = lr.predict(x_test)
y_pred
m=y_pred.astype(int)
from sklearn.metrics import r2_score
#r2_score(y_test,y_pred,force_finite=True)
r2_score(k,m)
lr.predict([[350,103,3,4.5,2.5,8.5,1]])
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import Sequential
model=keras.Sequential()
model.add(Dense(7,activation='relu',input_dim=7))
model.add(Dense(7,activation='relu'))
model.add(Dense(1,activation='softmax'))
model.summary()
#model.fit(x_train,y_train,batch_size=20,epochs=100)
model.compile(loss='binary_crossentropy',optimizer='adam',metrics = ['accuracy'])
```

```
model.fit(x_train,y_train,batch_size=20,epochs=100)
from sklearn.metrics import accuracy_score
train_predictions = model.predict(x_train)
print(train_predictions)
train_acc = model.evaluate(x_train,y_train,verbose=0)[1]
print(train_acc)
test_acc = model.evaluate(x_test,y_test,verbose=0)[1]
print(test_acc)
pred=model.predict(x_test)
pred = (pred>0.5)
pred
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matrix
from sklearn.metrics import multilabel_confusion_matrix
from sklearn.linear_model import LogisticRegression
print("Accuracy score :/n%f " %(accuracy_score(y_test,y_pred) * 100))
print("Recall score : %f" %(recall_score(y_test,y_pred) * 100))
print("ROC score : %f/n" %(roc_auc_score(y_test,y_pred) * 100))
print(confusion_matrix(y_test,y_pred))
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matrix
print(classification_report(y_test,pred))
model.save('model.h5')
```

app.py :

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
```

```
app = Flask(_name_)
from tensorflow.keras.models import load_model
model=load_model('model.h5')
@app.route('/')
def home():
    return render_template('Demo2.html')
@app.route('/y_predict',methods=['POST'])
def y_predict():
    '''
    for rendering results on HTML GUI
    '''
    #min max scaling
    min1=[290.0, 92.0, 1.0, 1.0, 1.0, 6.8, 0.0]
    max1=[340.0, 120.0, 5.0, 5.0, 5.0, 9.92, 1.0]
    k=[float(x) for x in request.form.values()]
    p=[]
    for i in range(7):
        l=(k[i]-min1[i])/(max1[i]-min[i])
        p.append(l)
        prediction = model.predict([p])
        print(prediction)
        output=prediction[0]
        if(output==False):
            return render_template('noChance.html',prediction_text='you dont have a chance of gettin')
        else:
            return render_template('Chance.html',prediction_text='you have a chance of getting admit')
    if _name_ == "_main_":
        app.run(debug=False)
```