

Intelligent Admissions: The Future of University Decision Making with Machine Learning

1. 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. Students are often worried about their chances of admission in University. the university admission process for students can be demanding, but by being well-informed, prepared, and organized, students can increase their chances of being admitted to the university of their choice. The aim of this project is to help students in short listing universities with their profiles. Machine learning algorithms are then used to train a model on this data, which can be used to predict the chances of future applicants being admitted. With this project, students can make more informed decisions about which universities to apply to, and universities can make more efficient use of their resources by focusing on the most promising applicants. The predicted output gives them a fair idea about their admission chances in a particular university. This analysis should also help students who are currently preparing or will be preparing to get a better idea.

1.1 Overview

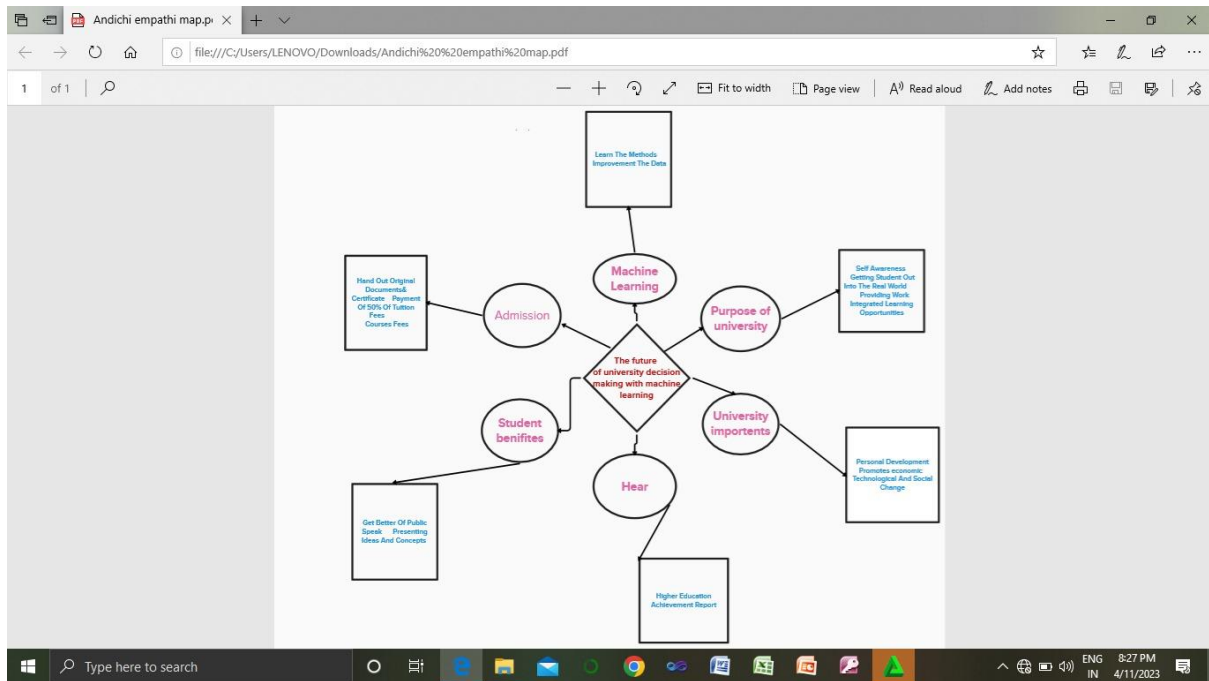
University admission is the process by which students are selected to attend a college or university. The process typically involves several steps, including submitting an application, taking entrance exams, and participating in interviews or other evaluations. Students are often worried about their chances of admission in University. The university admission process for students can be demanding, but by being well-informed, prepared, and organized, students can increase their chances of being admitted to the university of their choice.

1.2 Purpose

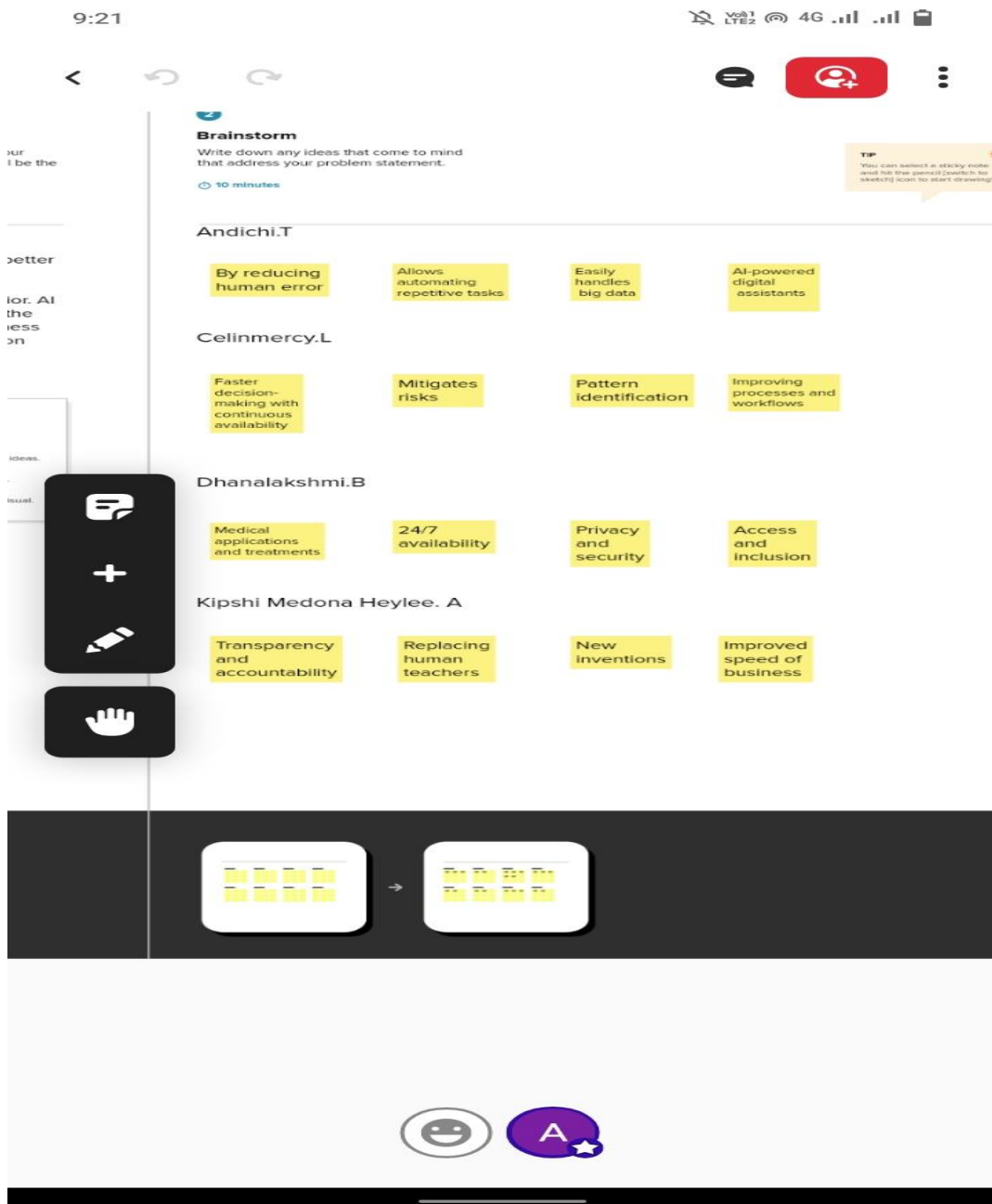
The aim of this project is to help students in short listing universities with their profiles. Machine learning algorithms are then used to train a model on this data, which can be used to predict the chances of future applicants being admitted. With this project, students can make more informed decisions about which universities to apply to, and universities can make more efficient use of their resources by focusing on the most promising applicants. The predicted output gives them a fair idea about their admission chances in a particular university. This analysis should also help students who are currently preparing or will be preparing to get a better idea.

2. Problem definition & Design Thinking

2.1 Empathy map



2.2 Ideation & Brainstorming Map





24/7
availability

24/7
availability

Privacy and security

Access and inclusion

By reducing human error

Improved speed of business

Replacing human teachers

Transparency and accountability

New inventions

AI-powered digital assistants

Easily handles big data

- Allows automating repetitive tasks

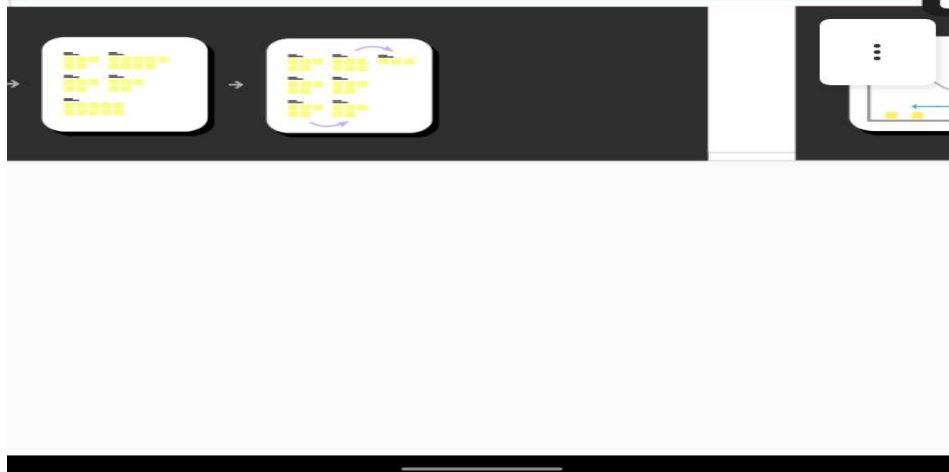
Faster decision-making with continuous availability

Medical applications and treatments

Mitigates risks

Pattern identification

Improving
processes and
workflows



9:34

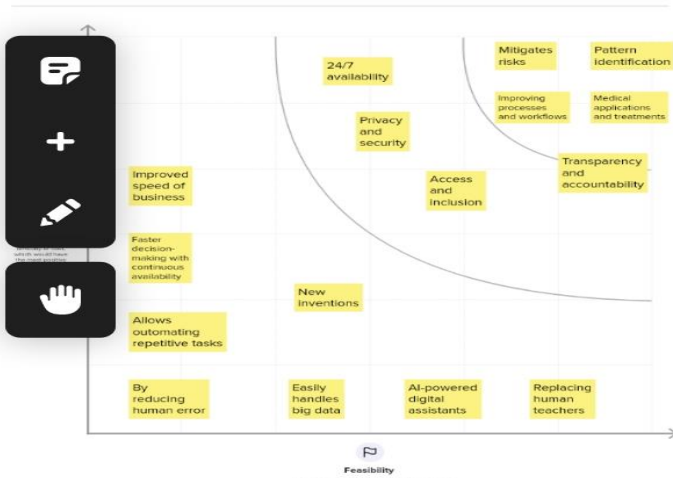
VoLTE 1 4G



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

You can end to share with might find it

Quick add

- Share it
- Share it
- Export it

Keep moving



Share it



3. RESULT

UNIVERSITY ADMISSION PREDICTION SYSTEM

Enter your details and get probability of your admission

Enter GRE Score

Enter TOEFL Score

Select University no

☐ 1

☒ 2

☐ 3

☐ 4

☐ 5

Enter SOP

Enter LOR

Enter GPA

Research


☐ Research

☒ NO Research



Predicting Chance of Admission

A Machine Learning Web App using Flask.

Prediction : **You have a chance** 



4. ADVANTAGES & DISADVANTAGES

ADVANTAGES

Intelligent Admissions has several advantages over traditional university admissions processes. Firstly, it eliminates human bias and subjectivity by using objective data analysis to make decisions. This reduces the chances of discrimination based on factors such as race, gender, or socioeconomic status.

Secondly, the system can process large amounts of data quickly and accurately, making the admissions process more efficient. This means that universities can admit more students and reduce the time and resources required for manual review of applications.

DISADVANTAGES

Despite its advantages, Intelligent Admissions also has some potential drawbacks. One concern is the accuracy of the predictions made by the system. While machine learning algorithms are highly advanced, they are not infallible and may make incorrect predictions.

Another concern is the lack of transparency in the decision-making process. Since the system uses complex algorithms to analyze data, it may be difficult for applicants to understand why they were accepted or rejected. This could lead to frustration and mistrust in the system.

5. APPLICATION

Artificial Intelligence And Machine Learning Improves User Experience. AI does not necessitate the use of a different app or device. It enhances the intelligence of the services we use on a daily basis. A mix of AI technology, such as chatbots, emulation, and virtual assistants like Google Assistant, is helping to increase user experience by incorporating many helpful features to an existing app.

6. CONCLUSION

In this paper, the authors illustrated the activities regarding admissions in the higher institutes where decision support systems are required for taking the admissions. The final section comprises the proposed architecture of Decision Support System, a modern approach to the decision making processes. ERP with IDSS provides good support for decision making otherwise the limitation of ERP will reduce the effectiveness system of decision making. Future work will focus on implementing the proposed architecture for the education system. In addition Specific Data Mining Techniques like Clustering can Profile Student intake and identify Promising Groups for Targeting in future intakes. Decision Trees can identify Loan needy students. Integrated educational Planning in the area can thus be facilitated by drilling down the Data to Schools in the area.

7. FUTURE SCOPE

Machine learning algorithms have the ability to analyze large amounts of data quickly and accurately. In the context of university admissions, this means that Intelligent Admissions can evaluate thousands of applications in a fraction of the time it would take for humans to do so. Furthermore, machine learning algorithms are not subject to the same biases and prejudices as human evaluators. This means that Intelligent Admissions can provide a more objective evaluation of each applicant, ensuring that universities are admitting the most qualified candidates regardless of their background or identity.

8. APPENDIX

A. Source code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
data = pd.read_csv('/content/Admission_Predict.csv')
```

```
data.info()
```

```
data.isnull().any()
```

```
data=data.rename(columns = {'Change of admit':'Change of  
Admit'})
```

```
data.describe()
```

```
sns.displot(data['GRE Score'])
```

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

```
sns.scatterplot(x='University  
Rating',y='CGPA',data=data,color='Red',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  
for 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=sc.fit_transform(x)  
x
```

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

```
y=data.iloc[:,7:].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=0.30,random_state=101)
```

```
y_train=(y_train>0.5)
y_train
```

```
y_test=(y_test>0.5)
```

```
from sklearn.linear_model.logistic import
LogisticRegression
cls = LogisticRegression (random_state =0)
```

```
lr=cls.fit(x_train,y_train)
y_pred =lr.predict(x_test)
y_pred
```

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dens, Activation,
Dropout
from tensorflow.keras.optimizers import Adam
```

```
model=keras.sequential()
model.add(Dens(7,activation ='relu',input_dim=7))
```

```
model.add(Dense(7,activation='relu'))
model.add(Dense(1,activation='linear'))
model.summary()
model:"sequential"
```

```
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_predictions)
```

```
test_acc = model.evaluate(x_test,y_test, verbose=0)[1]
print(test_acc)
```

```
print(classification_report(y_test.pred))
```

```
from sklearn.metrics import
accuracy_score,recall_score,roc_auc_score,confusion_matrix
print("\nAccuracy score:
%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  
accuracy_score,recall_score,confusion_matrix  
print(classification_report(y_train,pred))
```

```
from sklearn.metrics import  
accuracy_score,recall_score,roc_auc_  
print(classification_report(y_test,pred))
```

```
model.save('model.h5')
```

```
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')
```

```
def home():  
    return render_template('Demo2.html')
```

```
def home():  
    return render_template('Demo.html')  
def y_predict():
```



```

for rendering results on HTML GUI
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.from.values()]
p=[]
for i in range(7):
    l=(k[i]-min1[i]/(max1[i]))
    p.append(l)
prediction = model.prdict([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 gettin f admis')
if __name__=="__main__":
    app.run(debug=False)

```