INTELLIGENT ADMISSIONS : THE FUTURE OF UNIVERSITY DECISION MAKING WITH MACHINE LEARNING

TEAM SIZE : 4

TEAM LEADER : 1. **KEERTHIKA LAKSHMI G**

TEAM MEMBERS : 1. **KARTHIKA T**

2. **KAVIYA S**

3. **ASIKA V**

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

1. INTRODUCTION

Admissions to universities and colleges are a critical process in the education system. Institutions want to admit the best students who are a good fit for their programs and will succeed academically. Traditionally, admissions decisions have been made by humans, who review applications, transcripts, and other materials, and make judgments based on their experience and expertise. However, with the development of machine learning algorithms, it is now possible to use artificial intelligence (AI) to assist with the admissions process. In this document, we will explore the potential benefits of using machine learning in university admissions and the challenges that must be addressed to ensure that the system is fair and unbiased.

1.1 Overview

Improved Efficiency: Using machine learning algorithms can improve the efficiency of the admissions process. Machines can process large volumes of data much faster than humans, which can reduce the time and resources required to make admissions decisions. This can result in quicker turnaround times for applicants and allow universities to make admissions decisions faster.

Increased Accuracy: Machine learning algorithms can analyze a wide range of data points to make admissions decisions, including grades, test scores, extracurricular activities, and personal statements. This can result in more accurate and objective assessments of applicants, which can lead to better admissions decisions.

Reduced Bias: Machine learning algorithms can be designed to reduce bias in the admissions process. Humans are subject to unconscious biases, which can impact their decisions. Machine learning algorithms, on the other hand, can be trained to be impartial and objective, which can reduce the impact of bias in the admissions process.

Personalized Admissions: Machine learning algorithms can analyze the data of individual applicants to identify their strengths, weaknesses, and fit with the university's programs. This can allow universities to provide more personalized admissions decisions that take into account an applicant's unique qualities and potential.

1.2 Purpose

Data Quality: The quality of the data used in machine learning algorithms is critical to the accuracy of the system. If the data is incomplete or inaccurate, the algorithm's output may also be flawed. Universities must ensure that the data they use in the admissions process is of high quality and that it accurately reflects an applicant's academic and personal achievements.

Algorithmic Bias: Machine learning algorithms can also be biased if they are trained on biased data or designed with biased assumptions. This can result in unfair and discriminatory admissions decisions that disadvantage certain groups of applicants. Universities must carefully design and test their algorithms to ensure that they are fair and unbiased.

Transparency: Machine learning algorithms can be difficult to understand and interpret, which can make it challenging to determine how decisions are being made. Universities must be transparent about how their algorithms work and provide clear explanations of the factors that are considered in admissions decisions.

Privacy: The use of machine learning algorithms in admissions requires the collection and analysis of large amounts of personal data. Universities must ensure that they protect the privacy of applicants and comply with relevant data protection laws.

2. Problem Definition & Design Thinking :

The traditional university admissions process is time-consuming, resource-intensive, and often subject to bias and human error. The increasing volume of applications received by universities has made it difficult for admissions officers to review each application thoroughly, leading to a potentially unfair or inaccurate assessment of applicants. Moreover, the current admissions process may not fully capture the unique strengths, qualities, and potential of each applicant. These issues have created a need for a more efficient, accurate and fair admissions process.

Design Thinking:

Design thinking is a problem-solving approach that involves empathizing with the user, defining the problem, ideating potential solutions, prototyping and testing those solutions, and iterating until a final solution is reached. In the context of developing an intelligent admissions system, the following design thinking steps could be taken

Empathize: Start by empathizing with the university and the applicants. Understand their goals, challenges, and pain points. Talk to admissions officers, students, faculty members, and other stakeholders to gain a deeper understanding of their needs and aspirations.

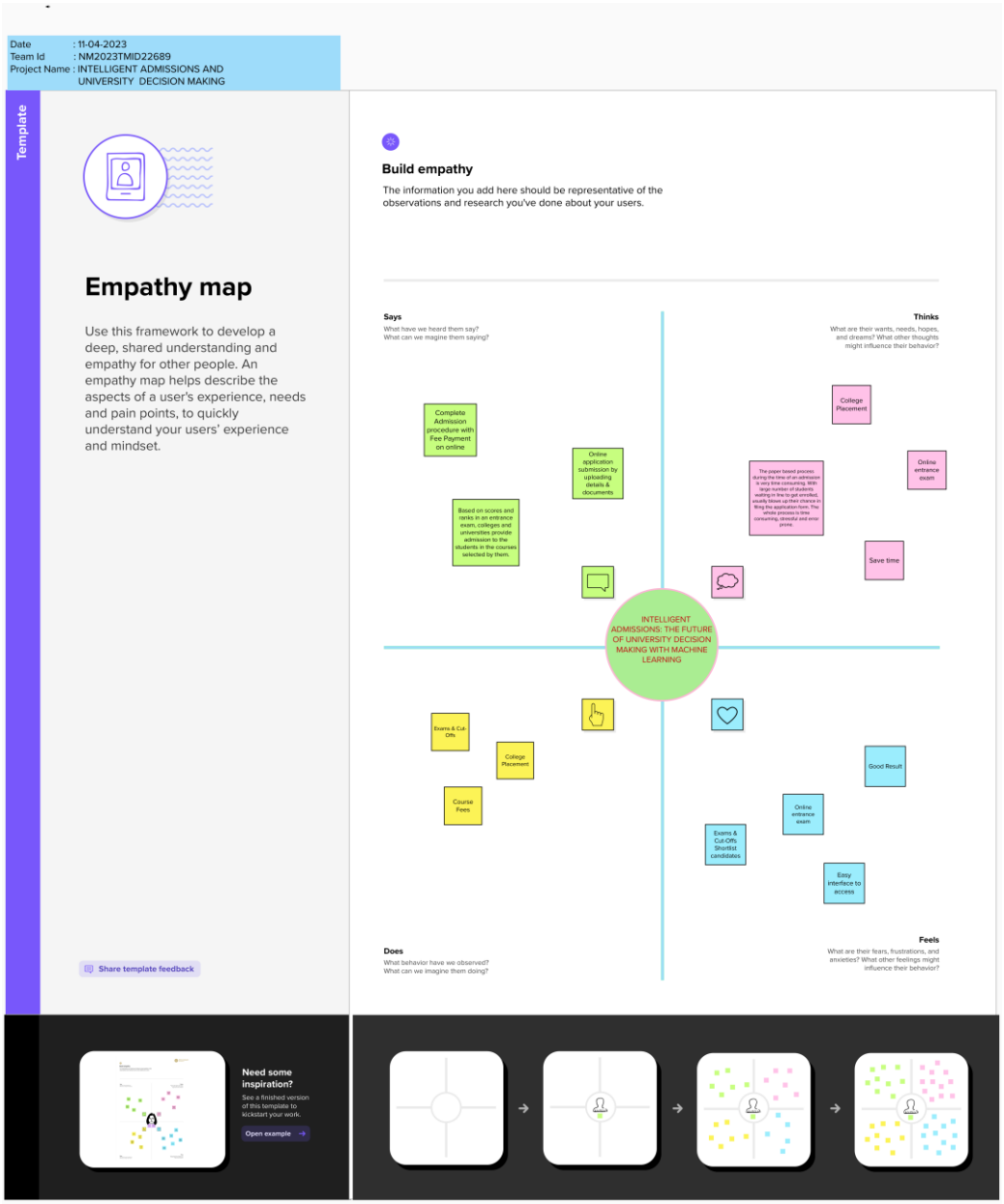
Define: Define the problem by synthesizing the insights gained during the empathy phase. Clearly define the problem statement and the goals of the intelligent admissions system. For example, the problem statement could be: "Design an intelligent admissions system that can efficiently and accurately assess applicants while reducing bias and ensuring fairness."

Ideate: Brainstorm potential solutions to the defined problem. Generate a wide range of ideas, no matter how unconventional they may seem. Consider different approaches, technologies, and tools that could be used to develop an intelligent admissions system. Use techniques like mind mapping, brainstorming, and rapid prototyping to generate and refine ideas.

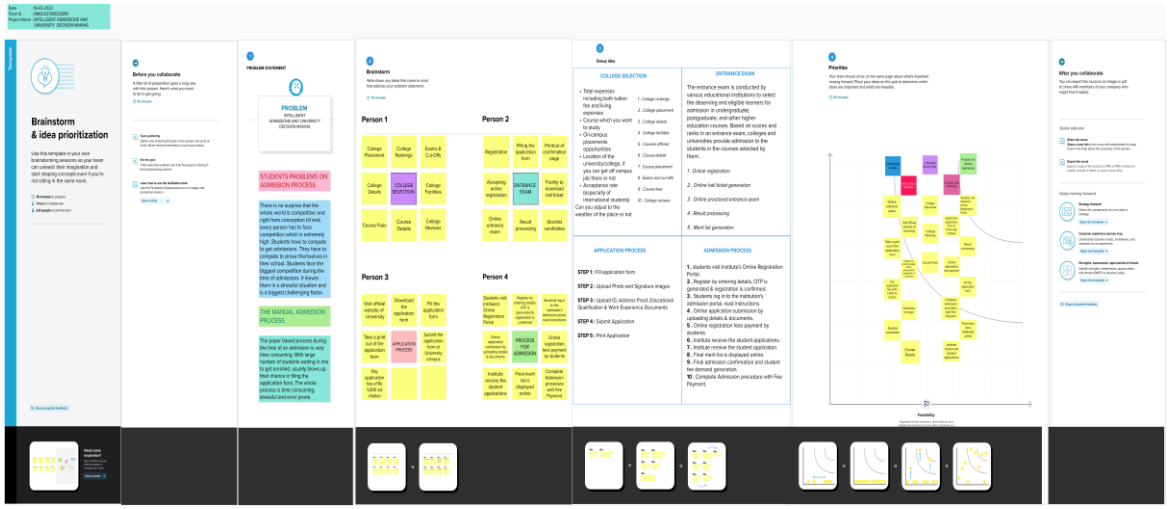
Prototype: Create a prototype or a minimum viable product (MVP) of the intelligent admissions system. This could be a simple machine learning model that uses historical data to predict an applicant's likelihood of success in a particular program. Test the prototype with a small group of users to get feedback and refine the solution.

Test and Iterate: Test the prototype with a larger group of users and collect feedback. Iterate the solution based on the feedback received, making necessary adjustments to improve the system's efficiency, accuracy, and fairness. Repeat the testing and iteration process until a final solution is developed.

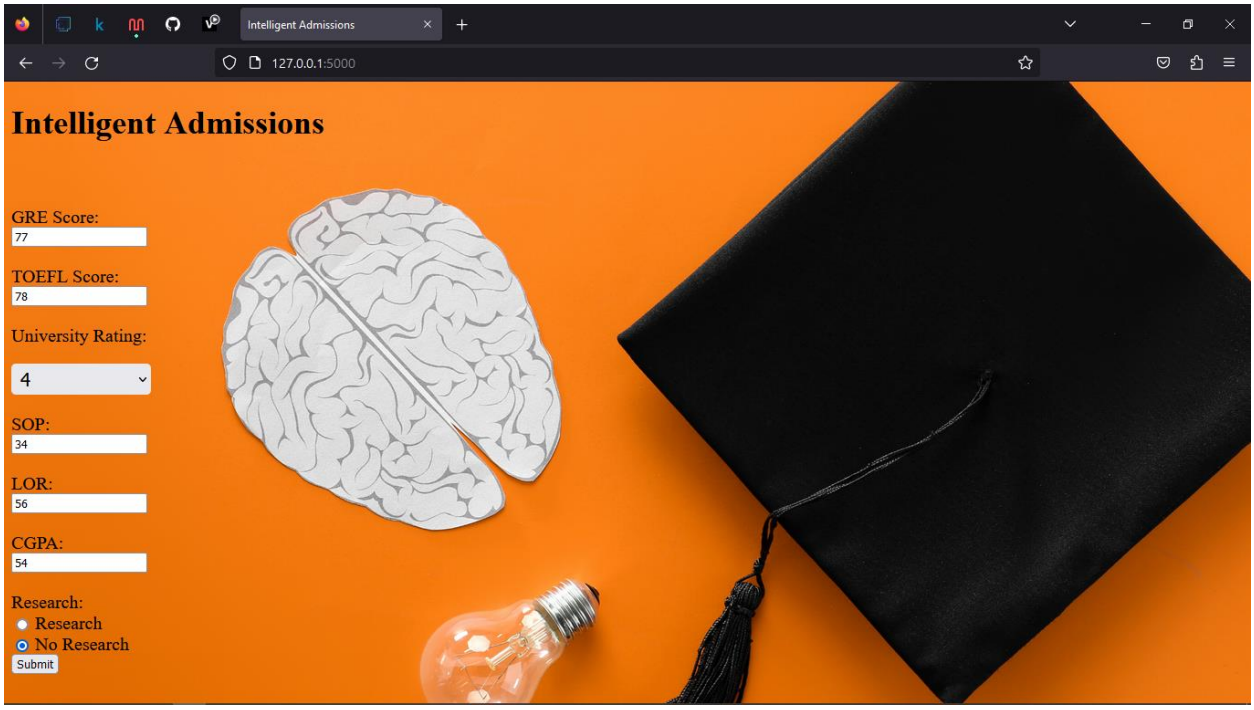
2.1 Empathy Map

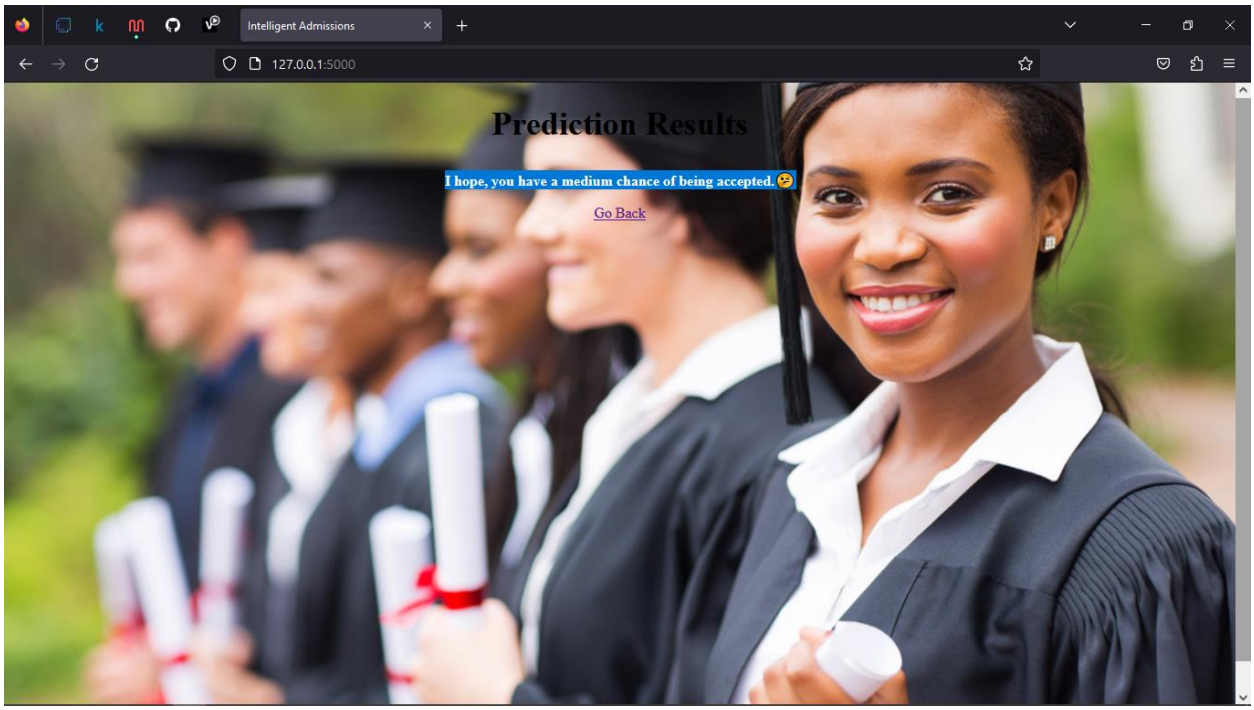


2.2 Ideation & Brainstorming Map



3. RESULT





4. ADVANTAGES & DISADVANTAGES

Efficiency: An intelligent admissions system can process applications at a much faster rate than traditional manual processes, saving time and resources for both applicants and admissions officers.

Accuracy: Machine learning models can analyze large volumes of data and make more accurate predictions and assessments than humans, reducing errors and improving the quality of admissions decisions.

Fairness: An intelligent admissions system can help reduce bias and ensure fairness by removing subjective judgments and considering a broader range of factors in the admissions process.

Personalization: Machine learning models can personalize the admissions process by analyzing applicant data and identifying individual strengths, interests, and potential.

Scalability: An intelligent admissions system can be scaled up to process a large number of applications without sacrificing quality, reducing the need for additional resources or personnel.

Disadvantages of an Intelligent Admissions System using Machine Learning

Data Bias: Machine learning models are only as good as the data they are trained on. If the data used to train the model contains bias, the model can perpetuate that bias in the admissions process.

Lack of Transparency: Machine learning models can be complex and difficult to understand, making it challenging to explain decisions to applicants and admissions officers.

Limited Contextual Understanding: Machine learning models may struggle to fully capture the nuances and complexities of an applicant's background, experiences, and potential, leading to potential inaccuracies or biases in the admissions process.

Ethical Concerns: The use of facial recognition technology and other data analysis techniques raises ethical concerns around privacy, consent, and the potential for discrimination or unfair treatment.

Cost: Developing and maintaining an intelligent admissions system can be costly, requiring investment in technology, personnel, and ongoing training and development.

Overall, an intelligent admissions system using machine learning can bring significant benefits to the university admissions process, but it is important to consider and address potential disadvantages and ethical concerns to ensure fairness and accuracy in the process.

5. APPLICATIONS

Here are some potential applications of an intelligent admissions system using machine learning in university admissions:

1. Application Screening: An intelligent admissions system can be used to screen applications quickly and accurately, identifying top candidates for further consideration.
2. Candidate Selection: Machine learning models can be used to assess the suitability of applicants for a particular program, course or scholarship based on their academic record, test scores, and other relevant factors.
3. Personalized Recommendations: An intelligent admissions system can generate personalized recommendations for academic programs, scholarships or other opportunities based on an applicant's strengths, interests, and potential.
4. Diversity and Inclusion: Machine learning models can help reduce bias and promote diversity and inclusion in the admissions process by considering a broader range of factors beyond traditional metrics such as test scores and grades.
5. Fraud Detection: An intelligent admissions system can use machine learning to identify fraudulent or misleading information in an application, such as fake transcripts or recommendation letters.
6. Chatbots and Virtual Assistants: Intelligent admissions systems can incorporate chatbots and virtual assistants to provide applicants with personalized guidance and support throughout the admissions process.

7. Predictive Analytics: Machine learning models can be used to predict an applicant's likelihood of success in a particular program, providing universities with valuable insights for strategic planning and resource allocation.
8. Continuous Improvement: An intelligent admissions system can be designed to continuously learn and improve over time, incorporating feedback from applicants and admissions officers to refine its algorithms and decision-making processes.

Overall, an intelligent admissions system using machine learning has the potential to transform the university admissions process by providing greater efficiency, accuracy, fairness, and personalization

6. Conclusion:

The use of machine learning in university admissions has the potential to improve the efficiency, accuracy, and fairness of the admissions process. However, universities must carefully consider the challenges and risks associated with using machine learning algorithms in admissions and take steps to address them. By doing so, they can create a more objective and personalized admissions process that benefits both applicants and institutions.

7. FUTURE SCOPE

The future scope of intelligent admissions using machine learning in universities is vast and promising. Here are some potential future developments:

1. Increased Personalization: As machine learning algorithms continue to improve, the future of intelligent admissions may include even more personalized recommendations and support for applicants.
2. Integration with Other Systems: Intelligent admissions systems can be integrated with other university systems, such as student information and learning management systems, to provide a more seamless and holistic experience for students.
3. Improved Bias Detection and Reduction: Future developments in machine learning algorithms may include improved bias detection and reduction techniques to ensure fairness and equity in the admissions process.
4. Enhanced Security and Privacy Measures: As the use of facial recognition and other biometric data becomes more common in admissions systems, future developments may focus on improving security and privacy measures to protect applicant data.
5. Collaboration with Industry Partners: Universities may partner with industry leaders in machine learning and artificial intelligence to continue to develop and improve intelligent admissions systems.
6. Integration with Learning Analytics: Future developments may include integrating intelligent admissions systems with learning analytics to provide a more comprehensive understanding of student success and program outcomes.

7. Integration with Other Admission Processes: Intelligent admissions systems can be integrated with other admission processes, such as financial aid and housing, to provide a more integrated and streamlined experience for students.

Overall, the future scope of intelligent admissions using machine learning is vast and exciting, with potential to transform the university admissions process and provide greater efficiency, accuracy, fairness, and personalization

8. APPENDIX

SOURCE CODE :

app.py Intellegence.py 2

Intellegence.py > ...

```python
# Importing required libraries
import numpy as np
from keras import optimizers
from sklearn.preprocessing import LabelBinarizer
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from tensorflow import keras
from tensorflow.keras.layers import Dense
from sklearn.metrics import accuracy_score, classification_report,recall_score,roc_auc_score,confusion_matrix,
precision_score

# Loading the data
try:
    data = pd.read_csv('Admission_Predict.csv')
except FileNotFoundError:
    print("File not found")


# Renaming column to remove extra space
print(data.info())
data = data.rename(columns = {'Chance of Admit ':' Chance of Admit'})


# Checking for missing values
```

```
Run Cell | Run Below | Debug Cell
# In[21]:


print(data.describe())


Run Cell | Run Above | Debug Cell
# In[22]:


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


Run Cell | Run Above | Debug Cell
# In[23]:


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


Run Cell | Run Above | Debug Cell
# In[24]:


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

```
Run Cell | Run Above | Debug Cell
# In[42]:


category = ['GRE Score','TOEFL Score','University Rating','SOP','LOR','CGPA','Research','ChanceofAdmit']
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()


Run Cell | Run Above | Debug Cell
# In[28]:
sc=MinMaxScaler()
x=sc.fit_transform(data.iloc[:,0:7].values)
print(x)
```

```
78
    Run Cell | Run Above | Debug Cell
79  # In[29]:
80
81
82  x=data.iloc[:,0:7].values
83  print(x)
84
85
    Run Cell | Run Above | Debug Cell
86  # In[30]:
87
88
89  y=data.iloc[:,7:].values
90  print(y)
91
92
    Run Cell | Run Above | Debug Cell
93  # In[31]:
94
95
96
97  x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.30,random_state=101)
98
99
    Run Cell | Run Above | Debug Cell
100 # In[32]:
101
```

```
102
103 y_train=(y_train>0.5)
104 print(y_train)
105
106
    Run Cell | Run Above | Debug Cell
107 # In[33]:
108
109
110 y_test=(y_test>0.5)
111
112
    Run Cell | Run Above | Debug Cell
113 # In[37]:
114 cls = LogisticRegression(random_state =0)
115
116 lr = cls.fit(x_train, y_train[:, 0])
117
    Run Cell | Run Above | Debug Cell
118 # In[ ]:
119
120
121 y_pred = lr.predict(x_test)
122 print(y_pred)
123
    Run Cell | Run Above | Debug Cell
124 # In[39]:
125
```

```
126
127  model=keras.Sequential()
128  model.add(Dense(7,activation = 'relu',input_dim=7))
129  model.add(Dense(7,activation='relu'))
130  model.add(Dense(1,activation='linear'))
131
132  model.summary()
133
134
     Run Cell | Run Above | Debug Cell
135  # In[40]:
136
137
138  model.compile(optimizer=optimizers.Adam(lr=0.001),loss='mse')
139
140  model.fit(x_train, y_train, batch_size=20, epochs=100)
141
142  model.compile(loss = 'binary_crossentropy',optimizer = 'adam',metrics = ['accuracy'])
143
144  from sklearn.metrics import accuracy_score
145
146  train_predictions = model.predict(x_train)
147
148  print(train_predictions)
149
150
     Run Cell | Run Above | Debug Cell
151  # In[40]:
```

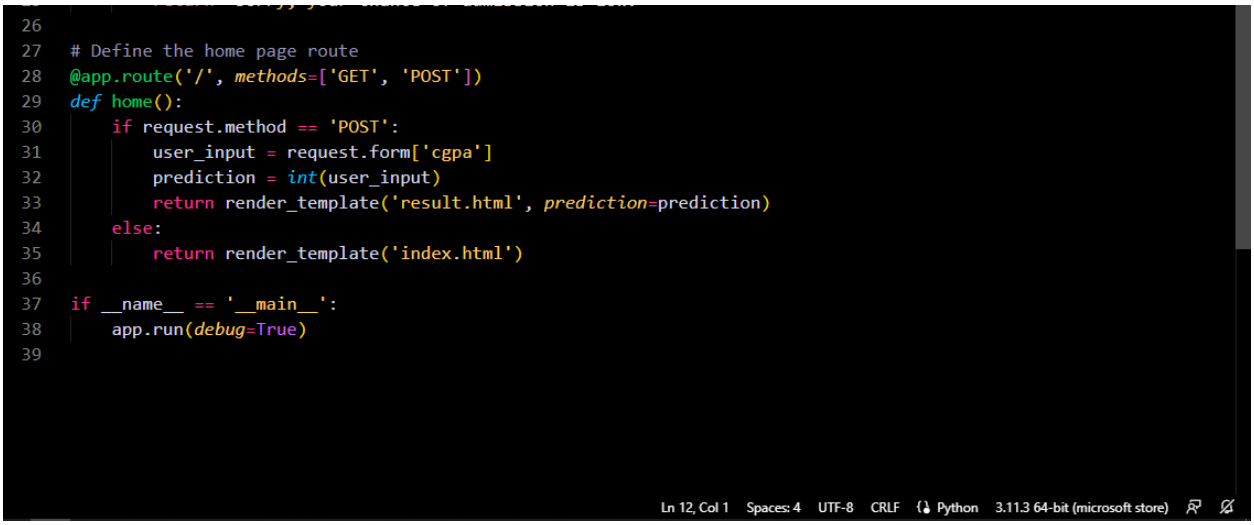
```
Intelligence.py > ...
152
153
154  train_acc = model.evaluate(x_train, y_train, verbose=0)[1]
155
156
     Run Cell | Run Above | Debug Cell
157  # In[41]:
158
159
160  test_acc = model.evaluate(x_test, y_test, verbose=0)[1]
161
162
     Run Cell | Run Above | Debug Cell
163  # In[43]:
164
165
166  pred=model.predict(x_test)
167  pred=(pred>0.5)
168  pred
169
170
     Run Cell | Run Above | Debug Cell
171  # In[44]:
172
```

```
172
173
174  y_pred_prob = model.predict(x_test)
175
176  # convert probabilities to class labels
177  y_pred = np.argmax(y_pred_prob, axis=1)
178
179  from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matrix
180  print("\nAccuracy score:%f" %(accuracy_score(np.argmax(y_test, axis=1),y_pred) * 100 ))
181  print("Recall score: %f" %(recall_score(np.argmax(y_test, axis=1),y_pred) * 100))
182  print("ROC score: %f\n" %(roc_auc_score(np.argmax(y_test, axis=1),y_pred) * 100))
183
184  print(confusion_matrix(np.argmax(y_test, axis=1),y_pred))
185
     Run Cell | Run Above | Debug Cell
186  # In[21]:
187  model.save('Admission.h5')
188
```

Ln 125, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.3 64-bit (microsoft store)

app.py > home

```
1   from flask import Flask, render_template, request
2   import numpy as np
3   import keras
4   import scipy
5   from sklearn.preprocessing import StandardScaler
6
7
8   app = Flask(__name__)
9
10  # Load the trained model
11  model = keras.models.load_model('Admission.h5')
12
13  # Define a function to preprocess the user input
14  def preprocess_input(user_input):
15      user_input = np.array(user_input).reshape(1, -1)
16      return user_input
17
18  # Define a function to make a prediction
19  def make_prediction(user_input):
20      user_input = preprocess_input(user_input)
21      prediction = model.predict(user_input)
22      if prediction >= 0.5:
23          return "Congratulations! You have a high chance of admission."
24      else:
25          return "Sorry, your chance of admission is low."
26
```



```python

# Define the home page route
@app.route('/', methods=['GET', 'POST'])
def home():
    if request.method == 'POST':
        user_input = request.form['cgpa']
        prediction = int(user_input)
        return render_template('result.html', prediction=prediction)
    else:
        return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True)

```