UNIVERSITY ADMISSION PREDICTOR USING MACHINE LEARNING

MINI PROJECT REPORT

Submitted by

AMOGALEKSHMI S (3122213002008)

CHATHURYA V (3122213002020)

DEEPTHI R (3122213002023)

UEC1605

MACHINE LEARNING

Department of Electronics and Communication Engineering

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

Rajiv Gandhi Salai (OMR), Kalavakkam – 603 110

EVEN SEM 2023-2024

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this mini project titled "**UNIVERSITY ADMISSION PREDICTOR USING MACHINE LEARNING**" is the bonafide work of "**AMOGALEKSHMI S** (3122213002008), **CHATHURYA V** (3122213002020) and **DEEPTHI R** (3122213002023) **of VI Semester Electronics and Communication Engineering Branch during Even Semester 2023 – 2024 for UEC1605 Machine Learning**

Submitted for examination held on 22/05/2024

INTERNAL EXAMINER

ABSTRACT

This study presents a University Admission Predictor utilizing machine learning algorithms including Linear Regression, Decision Tree, K-Nearest Neighbors (KNN), and Random Forest Regression. The predictor analyzes a training dataset consisting of various parameters relevant to university admissions such as GRE score, TOEFL score, University rank, SOP, LOR, CGPA and number of research papers done. Each model is evaluated based on performance metrics including Normalized Absolute Error (NAE), Normalized Squared Error (NSE), Root Mean Squared Error (RMSE), and R-squared (R2) score to determine the most effective algorithm for the task.

Following model evaluation, the best-performing model is selected based on the highest combined score across the evaluation metrics. Subsequently, the chosen model undergoes training on the entire dataset to maximize predictive accuracy. Upon completion of training, the predictor is deployed on the Streamlit platform, allowing users to input their parameters and receive a probability estimate indicating the likelihood of admission to the university.

This predictor provides valuable insights into university admissions processes, assisting prospective students in making informed decisions regarding their application strategies. Additionally, it serves as a practical demonstration of machine learning techniques applied to real-world scenarios, showcasing the potential for data-driven solutions in education and beyond.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVATIONS

Symbols	Abbreviation
ML	Machine Learning
KNN	K- Nearest Neighbor
MAE	Mean Absolute Error
MSE	Mean Square Error
RMSE	Root Mean Square Error
CGPA	Cummulative Grade Point Average
SOP	Standard Operating Procedure
LOR	Letter Of Recommendation

CHAPTER 1

INTRODUCTION

In the intricate web of university admissions, prospective students often find themselves navigating through a labyrinth of uncertainties and anxieties. The pursuit of higher education, a cornerstone of personal and professional growth, is fraught with the perennial question: Will I be accepted?

In response to this perennial query, the fusion of predictive analytics and machine learning emerges as a beacon of enlightenment, promising to unravel the complexities of admission decisions. This study embarks on a scholarly endeavor to develop a robust and reliable University Admission Predictor, poised to guide aspiring scholars through the daunting terrain of the admissions process.

Grounded in a comprehensive dataset encompassing an array of admission criteria, including academic performance metrics, standardized test scores, extracurricular engagements, and personal statements, our investigation delves deep into the realm of algorithmic exploration. With a meticulous approach, we subject various machine learning algorithms—Linear Regression, Decision Trees, K-Nearest Neighbors (KNN), and Random Forest Regression—to rigorous scrutiny, evaluating their efficacy in predicting admission outcomes.

Driven by the imperative of precision and accuracy, our inquiry is anchored in a rigorous assessment framework. Performance metrics such as Normalized Absolute Error (NAE), Normalized Squared Error (NSE), Root Mean Squared Error (RMSE), and R-squared (R2) score serve as our guiding stars, illuminating the path towards discerning the most adept predictive model.

As the dust settles and the victor emerges from the crucible of evaluation, we undertake the arduous task of refining and honing our chosen model through exhaustive training on the entirety of our dataset. Armed with newfound insights and fortified by empirical evidence, our champion stands ready to empower prospective students with the invaluable gift of foresight.

Upon completion of training, the predictor finds its sanctuary within the Streamlit platform—a user-friendly interface designed to bridge the chasm between data and decision-making. Here, amidst the virtual corridors of technological innovation, students find solace and clarity as they confront the age-old question: What are my chances?

In traversing the nexus of academia and artificial intelligence, this study not only advances the frontiers of predictive analytics but also underscores the transformative potential of machine learning in guiding critical life decisions. By demystifying the admissions process and fostering transparency, our endeavor seeks to empower individuals in their pursuit of academic excellence and personal fulfillment.

CHAPTER 2

LITERATURE SURVEY

The Application of SVR-based Combination Algorithm in Applying for College in China

Xingbing Liu; Yingying Wang; Zhen Zhang; Bin Chai; Yun Zhou; Shaoshuai Zhang

2020 IEEE 2nd International Conference on Computer Science and Educational Informatization (CSEI)

Year: 2020 | Conference Paper | Publisher: IEEE

In their paper, "The Application of SVR-based Combination Algorithm in Applying for College in China," Liu et al. address the challenges of college admissions in China. They propose an SVR-based combination algorithm to predict admission results, aiming to provide students with a more informed decision-making process. The authors review existing methods, highlight the limitations of traditional approaches, and establish the theoretical foundation for their research. They emphasize the necessity for accurate prediction models in the highly competitive Chinese college admission system.

……..……

University Admission Prediction Using Google Vertex AI

Jayashree Katti; Jony Agarwal; Swapnil Bharata; Swati Shinde; Saral Mane; Vinod Biradar

2022 First International Conference on Artificial Intelligence Trends and Pattern Recognition (ICAITPR)

Year: 2022 | Conference Paper | Publisher: IEEE

In their paper, "University Admission Prediction Using Google Vertex

AI," Katti et al. propose the use of Google Vertex AI for predicting university admissions. They review existing methods, highlight the limitations of traditional approaches, and emphasize the importance of accurate prediction models in the admission process. The authors introduce Google Vertex AI and explain its application in admission prediction, establishing the theoretical foundation for their research.

……..……

Machine Learning Based Graduate Admission Prediction

Ananya R Patel; Bhuvana K S; Dhanya R S; D A Akshay; Shridhar B. Devamane; Tushar N

2023 International Conference on Computational Intelligence for Information, Security and Communication Applications (CIISCA)

Year: 2023 | Conference Paper | Publisher: IEEE

In their paper, "Machine Learning Based Graduate Admission Prediction," Patel et al. propose the use of machine learning techniques for predicting graduate admissions. They review existing methods, highlight the limitations of traditional approaches, and emphasize the importance of accurate prediction models in the admission process. The authors introduce various machine learning techniques and explain their application in admission prediction, establishing the theoretical foundation for their research.

……..……

Admission Predictor for MS in Foreign University using ML

B. Uday Kiran, B. Simon Paul, T. Pavan Kumar, Mr. M. Sathya Narayana

International Research Journal of Modernization in Engineering Technology and Science

In their paper, "Admission Predictor for MS in Foreign University using

ML," Kiran et al. propose a machine learning-based solution for predicting admission to MS programs in foreign universities. They review existing methods, highlight the limitations of traditional approaches, and introduce various machine learning techniques for admission prediction. The authors emphasize the importance of accurate prediction models in the admission process and establish the theoretical foundation for their research.

……..……

A Cloud-Based Data Analysis and Prediction System for University Admission

P K Binu; Ananthu Chandran; M Rahul

2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)

Year: 2019 | Conference Paper | Publisher: IEEE

In "A Cloud-Based Data Analysis and Prediction System for University Admission," Binu et al. introduce a cloud-based system for analyzing and predicting university admissions, highlighting its importance for accurate admission predictions.

CHAPTER 3

3.1 PROBLEM STATEMENT

Prospective university students face challenges in understanding their likelihood of admission based on various parameters such as GRE score, TOEFL score, university rank, SOP, LOR, CGPA, and research experience. To address this, a University Admission Predictor utilizing machine learning algorithms is developed. The predictor aims to assess an applicant's chances of admission accurately. It evaluates multiple machine learning algorithms, selecting the most effective one based on performance metrics. The selected model is then trained on a comprehensive dataset and deployed on the Streamlit platform, providing users with a probability estimate of their admission likelihood. This predictor aims to assist students in making informed decisions about their application strategies and demonstrates the potential of data-driven solutions in education.

3.2 USE CASE DIAGRAM

A use case diagram is a visual representation of the interactions between a system and its users. It illustrates the various ways users interact with a system to achieve specific goals or tasks. In a use case diagram, actors represent the different types of users or external systems, while use cases represent the functionalities or tasks the system provides. Use case diagrams help in understanding and defining system requirements from a user's perspective and provide a high-level overview of the system's functionality and its interactions with users.

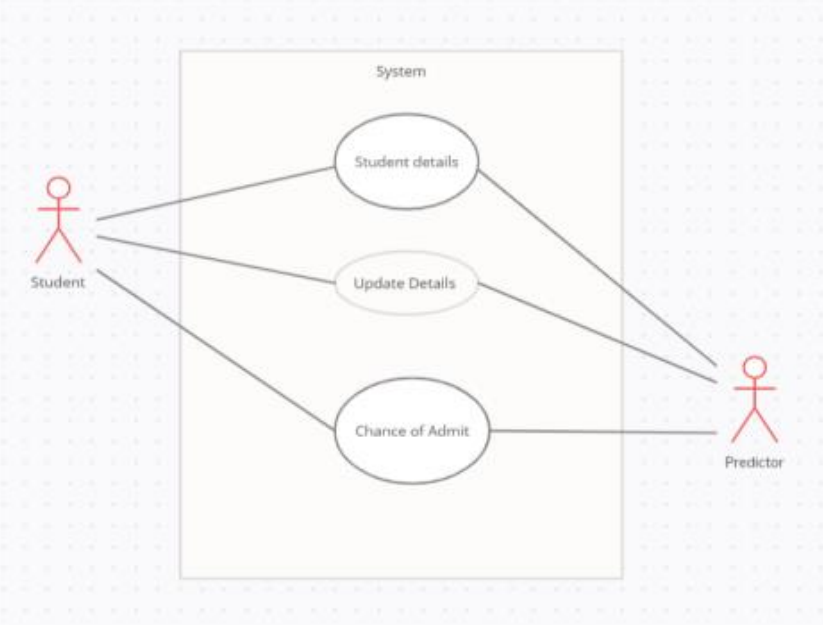

Figure 3.1 Use Case Diagram

3.3 PROCESS FLOW DIAGRAM

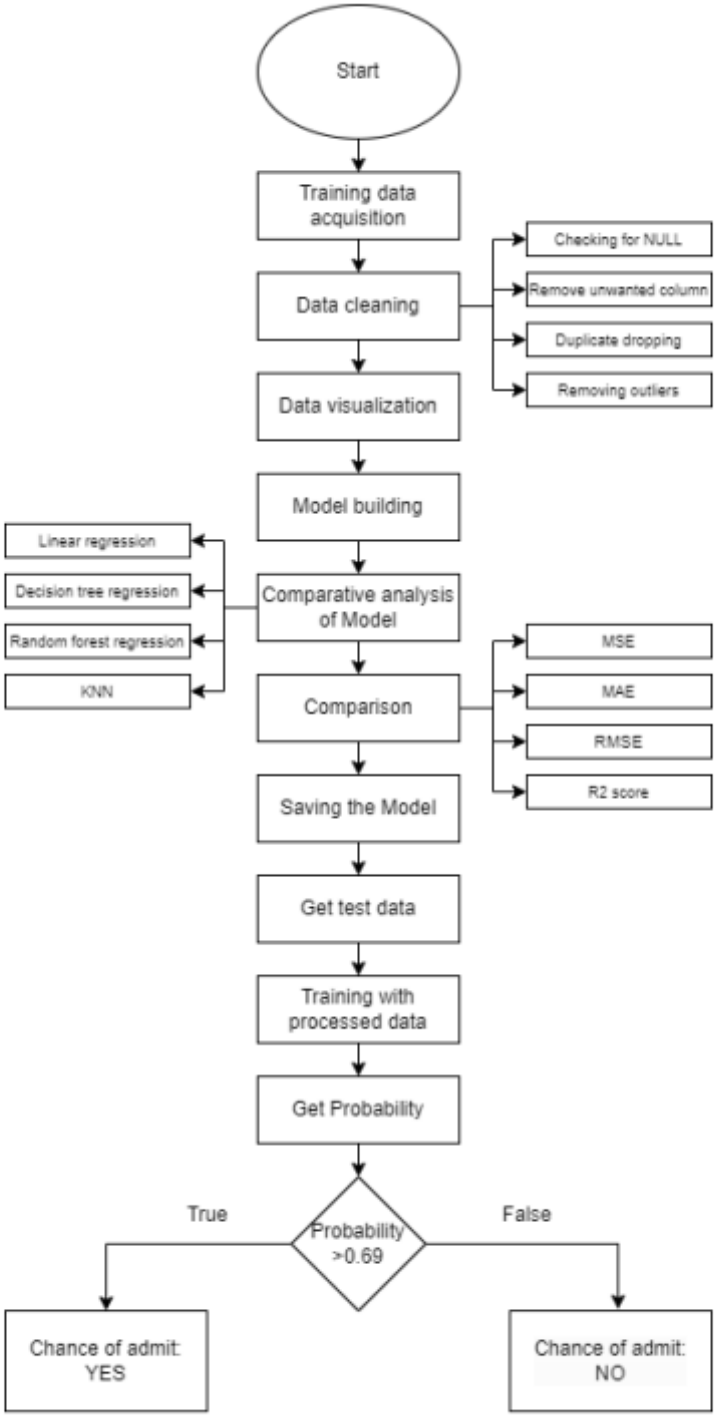

Figure 3.2 Process Flow Diagram

CHAPTER 4

In this chapter, the modules of College Admission Predictor are explained.

4.1 TRAINING AND TESTING

The dataset is processed before undergoing training and testing in the graduate admission predictor. Various methods are employed to determine the likelihood of admission.

4.2 DATA CLEANING

Data cleaning is the process of identifying and correcting errors, inconsistencies, and inaccuracies in a dataset to prepare it for analysis and modeling. In machine learning, data cleaning is a crucial step in the pipeline as the quality of the data directly impacts the performance and accuracy of the models.

Importance of Data Cleaning:

1. Improved Model Performance:

   Clean and high-quality data leads to more accurate and reliable models. Removing noise and inconsistencies from the dataset helps in building models that generalize well to new, unseen data.

2. Preventing Biases:

   Data cleaning helps in identifying and removing biases present in the dataset. Biased data can lead to biased models, which may produce unfair or inaccurate predictions.

3. Enhanced Feature Engineering:

   Clean data allows for better feature engineering. By handling missing values, outliers, and inconsistencies, data cleaning enables the creation of meaningful and informative features, which can improve the model's predictive power.

4. Reduced Overfitting:

   Overfitting occurs when a model learns noise and irrelevant patterns from the training data. By cleaning the data and removing noise and outliers, overfitting can be reduced, leading to more generalized and robust models.

5. Increased Efficiency:

   Clean data saves time and computational resources during the model training process. Models trained on clean data converge faster and require less computational power.

6. Enhanced Interpretability:

   Clean data leads to more interpretable models. When the data is clean and free from noise, it is easier to understand and interpret the model's predictions and decisions.

Data cleaning plays a critical role in the machine learning workflow, ensuring that the models are trained on high-quality, reliable data, leading to more accurate predictions and better insights. Here, we have tried to remove null values, columns that have unwanted data, duplicate drooping and remove outliers.

4.3 MODELS USED

4.3.1 LINEAR REGRESSION

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the dependent variable and the independent variable(s). In simple linear regression, there is only one independent variable, while in multiple linear regression, there are multiple independent variables. The objective of linear regression is to find the best-fitting line that represents the relationship between the independent variable(s) and the dependent variable. This is achieved by estimating the coefficients that minimize the sum of the squared differences between the observed and predicted values of the dependent variable. Linear regression is widely used for prediction and forecasting in various fields such as economics, finance, social sciences, and engineering, as well as for understanding the relationship between variables and making inferences about the data.

4.3.2 DECISION TREE REGRESSION

Decision tree regression is a machine learning algorithm used for regression tasks. It works by recursively partitioning the feature space into a set of rectangles and fitting a simple model (usually a constant) in each one. In decision tree regression, the target variable is continuous, and the algorithm selects the best attribute to split the data at each node to minimize the variance of the target variable within each subset. Decision trees are easy to interpret and visualize, making them useful for understanding the underlying decision-making process. They can capture non-linear relationships between features and the target variable, making them suitable for tasks where the relationship is not linear. However,

decision trees are prone to overfitting, especially when the tree depth is not properly controlled, and small variations in the data can result in a completely different tree structure, making the model less stable. Decision tree regression is used in various fields, including finance, healthcare, and marketing, for tasks such as sales forecasting, risk assessment, and customer segmentation.

4.3.3 RANDOM FOREST REGRESSION

Random forest regression is a machine learning algorithm that extends the concept of decision tree regression. It operates by constructing a multitude of decision trees during training and outputs the mean prediction of the individual trees for regression tasks. Each tree in the random forest is trained on a random subset of the training data, and a random subset of the features is considered for each split in the tree. This randomness helps to reduce overfitting and improve the model's performance. Random forest regression is highly accurate and robust, capable of handling large datasets with high dimensionality. It is widely used in various fields, including finance, healthcare, and bioinformatics, for tasks such as predicting stock prices, medical diagnosis, and gene expression analysis. Random forest regression is known for its versatility, scalability, and ability to handle non-linear relationships between features and the target variable.

4.3.4 K- NEAREST NEIGHBORS

K-Nearest Neighbors (KNN) is a simple and intuitive machine learning algorithm used for both classification and regression tasks. In KNN, the prediction for a new data point is made by considering the majority class (for classification) or the average (for regression) of the K nearest data points in the feature space. The distance metric, typically Euclidean

distance, is used to measure the similarity between data points. KNN is a non-parametric and lazy learning algorithm, meaning it does not make any assumptions about the underlying data distribution during training. Instead, it memorizes the training data and makes predictions based on the similarity of new data points to the training data. KNN is easy to understand and implement, making it a popular choice for simple classification and regression tasks. However, it can be computationally expensive, especially when dealing with large datasets, as it requires calculating the distance between the new data point and all training data points for each prediction.

4.4 PERFORMANCE METRICES USED

MSE (Mean Squared Error), MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), and R-squared (R2) score are commonly used performance metrics for evaluating regression models

4.4.1 Mean Squared Error (MSE):

MSE measures the average of the squared differences between predicted and actual values. It penalizes large errors more than small errors.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

4.4.2 Mean Absolute Error (MAE):

MAE measures the average of the absolute differences between predicted and actual values. It is less sensitive to outliers compared to MSE.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

4.4.3 Root Mean Squared Error (RMSE):

RMSE is the square root of the average of the squared differences

between predicted and actual values. It gives an estimate of the standard deviation of the prediction errors.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

4.4.4 R2 SCORE:

R2 score represents the proportion of the variance in the dependent variable that is predictable from the independent variables. It indicates the goodness of fit of the model.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

CHAPTER 5

5.1 DATASET DESCRIPTION

Serial No.	GRE Score	TOEFL Sco	University	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	4.5	8.87	1	0.76
3	316	104	3	3	3.5	8	1	0.72
4	322	110	3	3.5	2.5	8.67	1	0.8
5	314	103	2	2	3	8.21	0	0.65
6	330	115	5	4.5	3	9.34	1	0.9
7	321	109	3	3	4	8.2	1	0.75
8	308	101	2	3	4	7.9	0	0.68
9	302	102	1	2	1.5	8	0	0.5
10	323	108	3	3.5	3	8.6	0	0.45
11	325	106	3	3.5	4	8.4	1	0.52
12	327	111	4	4	4.5	9	1	0.84
13	328	112	4	4	4.5	9.1	1	0.78
14	307	109	3	4	3	8	1	0.62
15	311	104	3	3.5	2	8.2	1	0.61
16	314	105	3	3.5	2.5	8.3	0	0.54
17	317	107	3	4	3	8.7	0	0.66
18	319	106	3	4	3	8	1	0.65
19	318	110	3	4	3	8.8	0	0.63
20	303	102	3	3.5	3	8.5	0	0.62
21	312	107	3	3	2	7.9	1	0.64
22	325	114	4	3	2	8.4	0	0.7
23	328	116	5	5	5	9.5	1	0.94
24	334	119	5	5	4.5	9.7	1	0.95
25	336	119	5	4	3.5	9.8	1	0.97
26	340	120	5	4.5	4.5	9.6	1	0.94

Admission_Predict_Ver1.1

Figure 5.1 Dataset

This dataset contains information on university applicants for a graduate program, likely a Master's program in India. The data, potentially obtained from a source like Kaggle, is in a format like CSV and aims to predict admission chances. Each applicant record includes a unique identifier, scores on standardized tests (GRE, TOEFL), ratings for application materials (SOP, LOR), their CGPA, a binary variable indicating research experience, and the predicted probability of admission.

5.2 MODEL CODING

5.2.1 TRAINING DATA CODING

```
 #importing the nessescary packages
 import pandas as pd
 import numpy as np
 import seaborn as sns
 import matplotlib.pyplot as plt
%matplotlib inline
df=pd.read_csv("train_datasets_14872_228180_Admission_Predict_Ver
1.1.csv")
sns.heatmap(df.isnull(),yticklabels = False)
plt.title('Missing Data')
sns.heatmap(df.isnull(),yticklabels = False)
plt.title('Missing Data')
# imputing the fields according to the need
 def impute_TANCET(cols):
     cgpa=cols[0]
     return(int(8.8*cgpa))
 def impute_GATE(cols):
     GATE=cols[0]
   return(int((GATE-2.56)*10))
 df['Tancet'] = df[["CGPA"]].apply(impute_TANCET,axis=1)
df['GATE']   = df[["CGPA"]].apply(impute_GATE,axis=1)
 df.drop_duplicates(inplace=True)
```

```
df.shape
# as Serial No. has no co-relation we can drop that
del df["Serial No."]
 f,ax=plt.subplots(figsize = (10,10))
sns.heatmap(df.corr(method="pearson"),annot=True,fmt
='.2g',square=True,cmap='coolwarm',cbar=False,ax=ax)
df.columns
 # spliting of dependent and independent features
 X = df.drop("Chance_of_Admit",axis=1)
y  = df["Chance_of_Admit"]
#train test split split ratio (test:train)=70:30
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,rando_s
tate=1)
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X_train,y_train)
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_predict)
 from sklearn import metrics
 print("Model Report Using Linear Regression")
 print('MAE:', metrics.mean_absolute_error(y_test, y_predict))
 print('MSE:', metrics.mean_squared_error(y_test, y_predict))
 print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_predict)))
```

```
from sklearn.metrics import r2_score
print('r2_score:',r2_score(y_test, y_predict))
from sklearn.tree import DecisionTreeRegressor
dtree=DecisionTreeRegressor(criterion="mse")
reg_dtree=dtree.fit(X_train,y_train)
y_pred_dtree = reg_dtree.predict(X_test)
print("Model Report Using Decision TreeRegressor")
print('MAE:', metrics.mean_absolute_error(y_test, y_pred_dtree))
print('MSE:', metrics.mean_squared_error(y_test, y_pred_dtree))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test,
y_pred_dtree)))
from sklearn.metrics import r2_score
print('r2_score:',r2_score(y_test, y_pred_dtree))
from sklearn.ensemble import RandomForestRegressor
rfregressor = RandomForestRegressor(n_estimators = 10, random_state
= 0)
rfregressor.fit(X_train,y_train)
y_pred_randomforest = rfregressor.predict(X_test)
#MODEL EVALUATION
print('MAE:',metrics.mean_absolute_error(y_test,y_pred_randomforest)
)
print('MSE:',metrics.mean_squared_error(y_test,y_pred_randomforest))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,y_pred_rando
mforest)))
print('r2_score:',r2_score(y_test, y_pred_randomforest))
from sklearn.neighbors import KNeighborsRegressor
```

```
knn_reg = KNeighborsRegressor(n_neighbors=1)
knn_reg.fit(X_train, y_train)
y_pred_knn=knn_reg.predict(X_test)
print('MAE:', metrics.mean_absolute_error(y_test, y_pred_knn))
print('MSE:', metrics.mean_squared_error(y_test, y_pred_knn))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,y_pred_knn))
print('r2_score:',r2_score(y_test, y_pred_knn))
A = LinearRegression()
B = DecisionTreeRegressor(criterion="mse")
C = RandomForestRegressor(n_estimators = 10, random_state = 0)
D = KNeighborsRegressor(n_neighbors=3)
objects = ('LReg', 'DTree', 'RForest', 'KNN')
# function to train regressor
def train_classifier(reg, X_train, y_train):
    reg.fit(X_train, y_train)
# function to predict features
def predict_labels(reg, features):
return (reg.predict(features))
reg = [A,B,C,D]
pred_val = [0,0,0,0]
for a in range(0,4):
    train_classifier(reg[a], X_train, y_train)
    y_pred = predict_labels(reg[a],X_test)
    pred_val[a] =metrics.mean_absolute_error(y_test, y_pred)
    print("--"*50)
```

```
    print(pred_val[a])
y_pos = np.arange(len(objects))
y_val = [ x for x in pred_val]
plt.bar(y_pos,y_val, align='center', alpha=0.9)
plt.xticks(y_pos, objects)
plt.ylabel('MAE Score')
plt.title('MAE of Models')
plt.show()
reg = [A,B,C,D]
pred_val = [0,0,0,0]
for a in range(0,4):
    train_classifier(reg[a], X_train, y_train)
    y_pred = predict_labels(reg[a],X_test)
    pred_val[a] =metrics.mean_squared_error(y_test, y_pred)
    print("--"*50)
    print(pred_val[a])
y_pos = np.arange(len(objects))
y_val = [ x for x in pred_val]
plt.bar(y_pos,y_val, align='center', alpha=0.9)
plt.xticks(y_pos, objects)
plt.ylabel('MSE Score')
plt.title('MSE of Models')
plt.show()
reg = [A,B,C,D]
pred_val = [0,0,0,0]
```

```
for a in range(0,4):
    train_classifier(reg[a], X_train, y_train)
    y_pred = predict_labels(reg[a],X_test)
    pred_val[a] =np.sqrt(metrics.mean_squared_error(y_test, y_pred))
    print("--"*50)
    print(pred_val[a])
y_pos = np.arange(len(objects))
y_val = [ x for x in pred_val]
plt.bar(y_pos,y_val, align='center', alpha=0.9)
plt.xticks(y_pos, objects)
plt.ylabel('RMSE Score')
plt.title('RMSE of Models')
plt.show()
from sklearn.metrics import r2_score
reg = [A,B,C,D]
pred_val = [0,0,0,0]

for a in range(0,4):
    train_classifier(reg[a], X_train, y_train)
    y_pred = predict_labels(reg[a],X_test)
    pred_val[a] =r2_score(y_test, y_pred)
    print("--"*50)
    print(pred_val[a])
y_pos = np.arange(len(objects))
```

```
y_val = [ x for x in pred_val]
plt.bar(y_pos,y_val, align='center', alpha=0.9)
plt.xticks(y_pos, objects)
plt.ylabel('RMSE Score')
plt.title('RMSE of Models')
plt.show()
import pickle
model=open("Admission_predict.pkl","wb")
pickle.dump(dtree,model)
model.close()
```

5.2.2 TESTING DATA CODING

```
#importing the nessescary packages
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
#importing the nessescary packages
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
df.columns = ['Time_of_collection',
'Name_of_the_student',"CLG_name","type_of_clg","Contact",'GRE
```

```
Score','TOEFL Score','GATE','Tancet','CGPA','Research','University
Rating','SOP','LOR ']
df1=df[['GRE Score','TOEFL
Score','GATE','Tancet','CGPA','Research','University
Rating','SOP','LOR ']]
df1 = df1[['GRE Score','TOEFL Score','University Rating','SOP','LOR
','CGPA','Research','Tancet','GATE']]
import pickle
filename = 'Admission_predict.pkl'
loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model.predict(df1)
df["Probability"]=result
chance_of_admit_threshold=0.69
def chance(var):
    a = []
    for i in df[var]:
        if i >=chance_of_admit_threshold:
            i="YES"
        else:
            i="NO"
        a.append(i)
    return a
df["chance_of_admit"]=chance("Probability")
sns.countplot(df["chance_of_admit"])
```

5.2.3 APP IMPLEMENTATION CODING

```
import numpy as np
import pickle
import pandas as pd
import streamlit as st
pickle_in = open("Admission_predict.pkl","rb")
admis_predicter=pickle.load(pickle_in)
#@app.route('/')
def welcome():
    return "Welcome All"
def
admisssion_chance(GRE_Score,TOEFL_Score,University_Rating,SOP,
LOR,CGPA,Research,Tancet,GATE):
prediction=admis_predicter.predict([[GRE_Score,TOEFL_Score,Univrs
ity_Rating,SOP,LOR,CGPA,Research,Tancet,GATE]])
    prediction=prediction.astype('float64')
    return prediction
def main():
    st.title("ADMISSSION PREDICTOR")
    html_temp = """
    <div style="background-color:tomato;padding:10px">
    <h2 style="color:white;text-align:center;">ADMISSION
 PREDICTION WEB APP  </h2>
    st.markdown(html_temp,unsafe_allow_html=True)
    GRE_Score = st.text_input("GRE Score","Type Here")
    TOEFL_Score = st.text_input("TOEFL_Score","Type Here")
```

```
    University_Rating = st.text_input(",University_Rating","Type Here")
    SOP= st.text_input("SOP","Type Here")
    LOR= st.text_input("LOR","Type Here")
    CGPA= st.text_input("CGPA","Type Here")
    Research= st.text_input("Research","Type Here")
    Tancet= st.text_input("Tancet","Type Here")
    GATE=st.text_input("gate score","Type Here")
    result="YET TO BE PREDICTED"
    result1 = "YET TO BE PREDICTED"
    s=0.000
    if st.button("Predict"):
result=admisssion_chance(float(GRE_Score),float(TOEFL_Score),float
(University_Rating),float(SOP),float(LOR),float(CGPA),float(Research
),float(Tancet),float(GATE))
         s=result[0]
        print(s,type(s))
    if(float(s)>=float(0.69)):
        result1="YOU HAVE MORE PROBABBILITY TO GET
ADMISSION"
    else:
        result1="ITS LITTLE TOUGH TO GET ADMISSSION IN TOP
UNIVERSITIES "
    st.success(result1)
    if st.button("About"):
        st.text("Lets LEarn")
        st.text("Built BY A2 TEAM")if _name=='main_': main()
```

CHAPTER 6

6.1 OUTPUTS

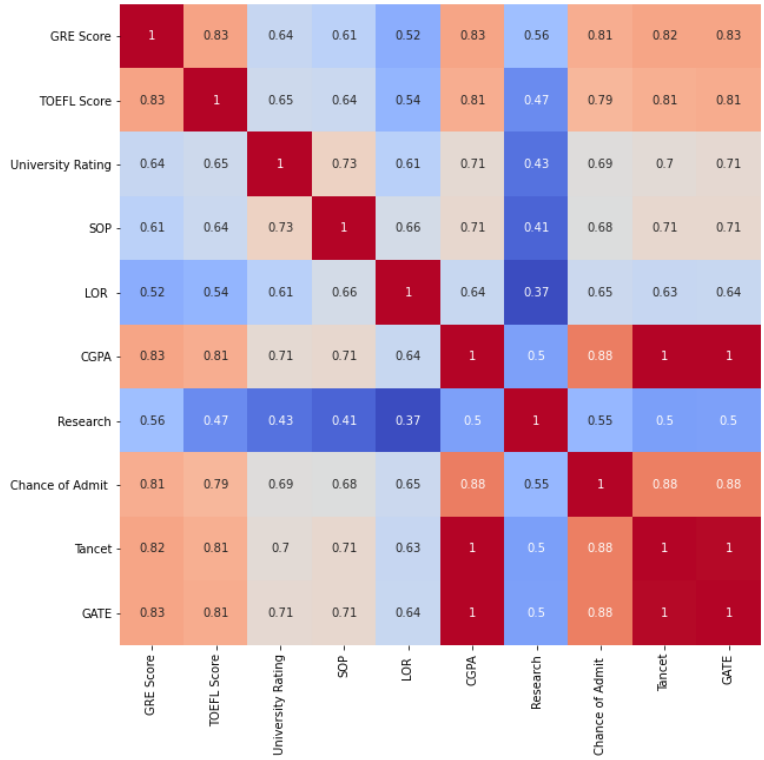

Figure 6.1 Heatmap of Training data

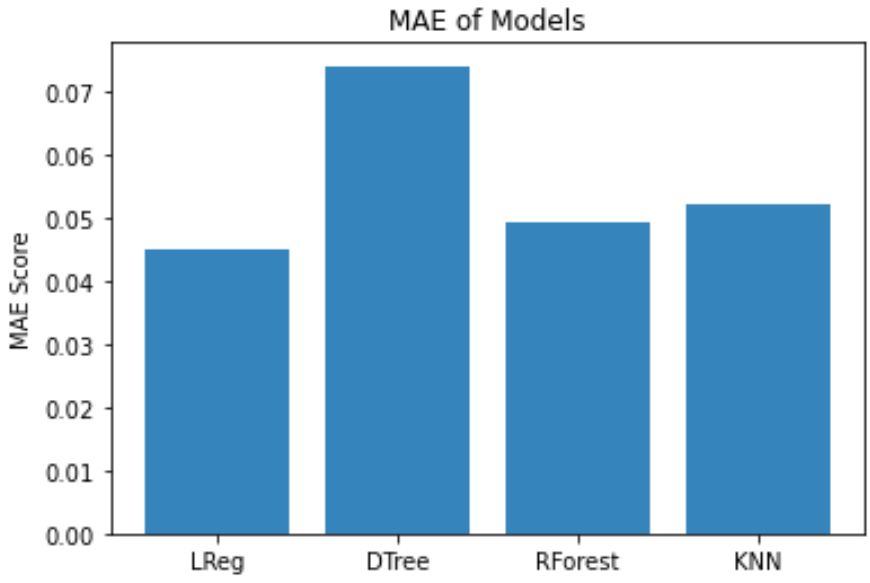

Figure 6.2 MAE of Models

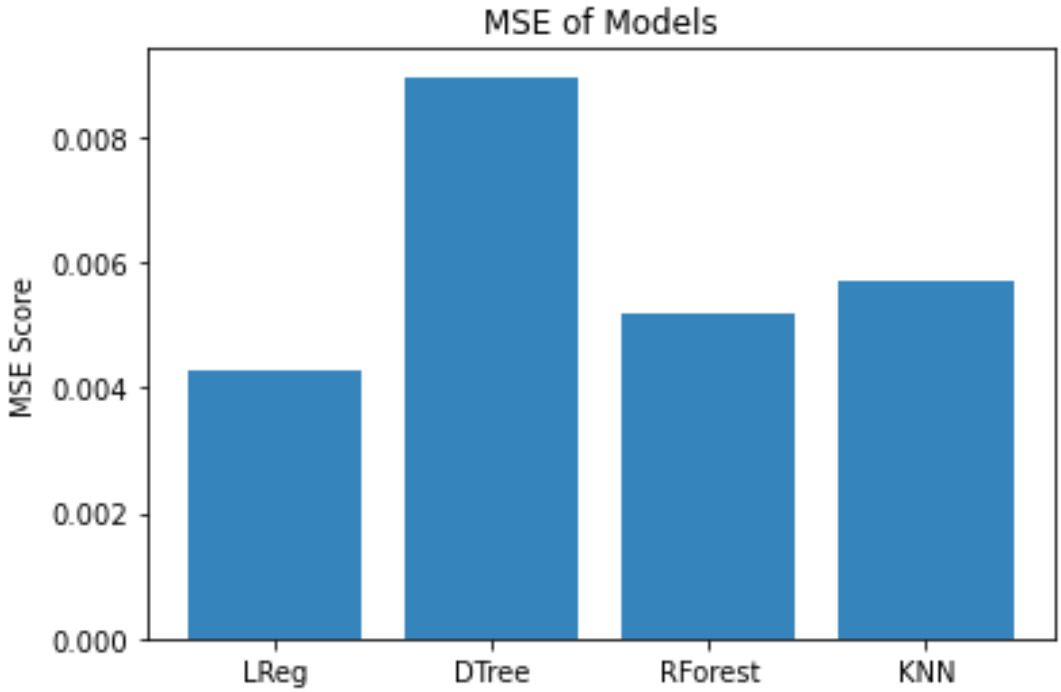

Figure 6.3 MSE of Models

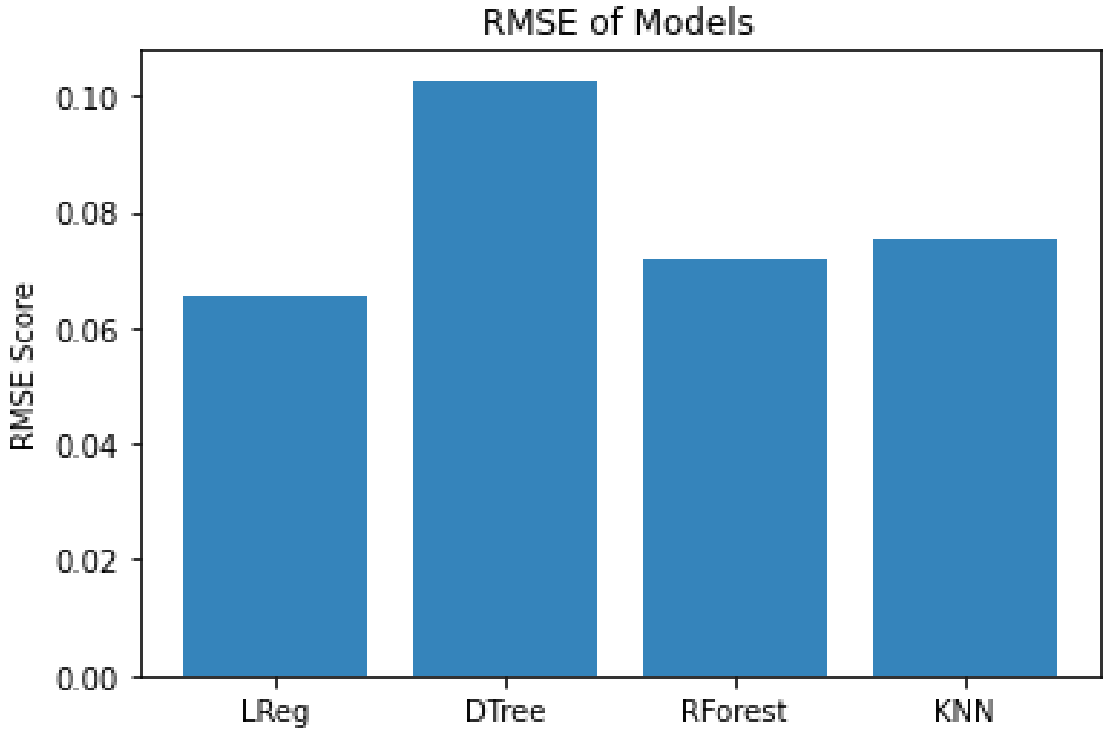

Figure 6.4 RMSE Of Models

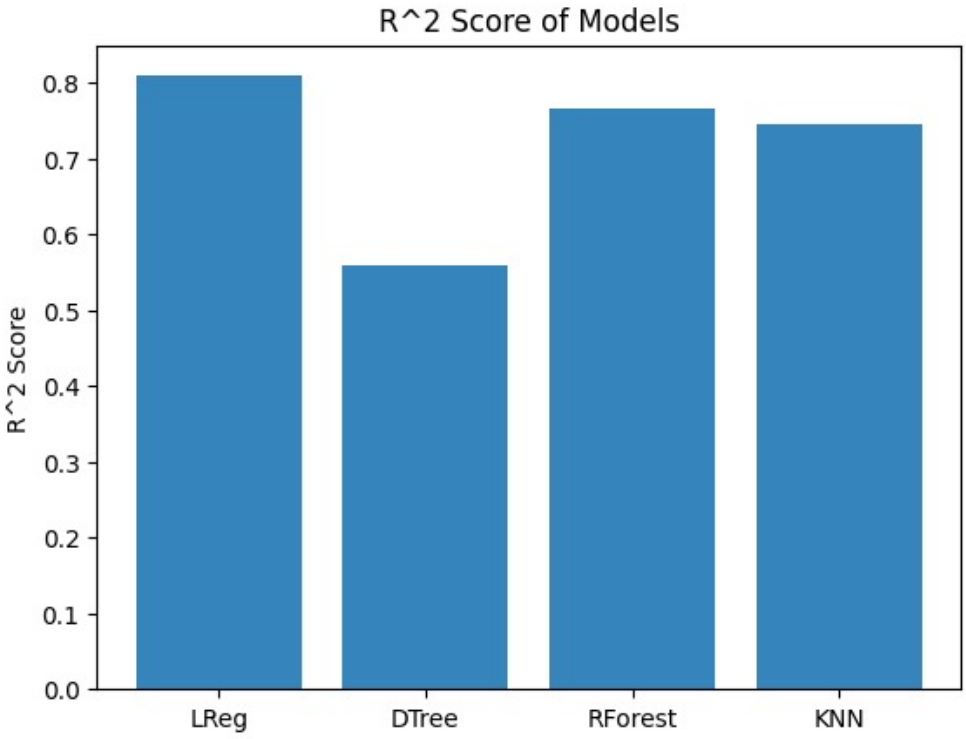

Figure 6.5 R2 Score Of Models

```
result
```

```
array([0.9 , 0.66, 0.48, 0.93, 0.56, 0.48, 0.63, 0.52, 0.96, 0.62, 0.37,
       0.45, 0.45, 0.36, 0.68, 0.97, 0.48, 0.71, 0.38, 0.48, 0.62, 0.66,
       0.76, 0.76, 0.68, 0.89, 0.38, 0.76, 0.36, 0.71, 0.66, 0.7 , 0.59,
       0.74, 0.72, 0.64, 0.49, 0.8 , 0.36, 0.87, 0.7 , 0.78, 0.86, 0.7 ,
       0.84, 0.66, 0.82, 0.87, 0.87, 0.86, 0.86, 0.87, 0.92, 0.86, 0.76,
       0.71, 0.38, 0.64, 0.71])
```

Figure 6.6 Probability Array

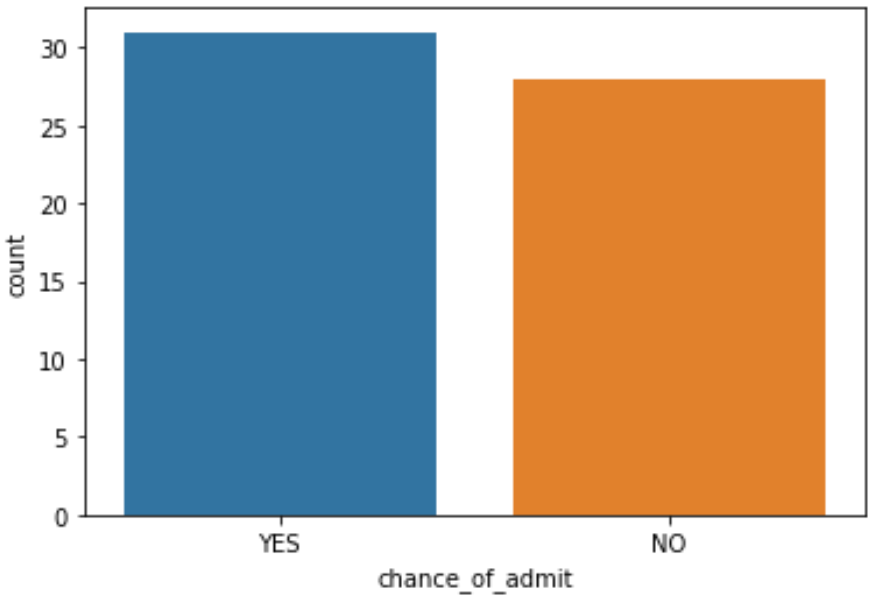

Figure 6.7 Chance of Admit Plot

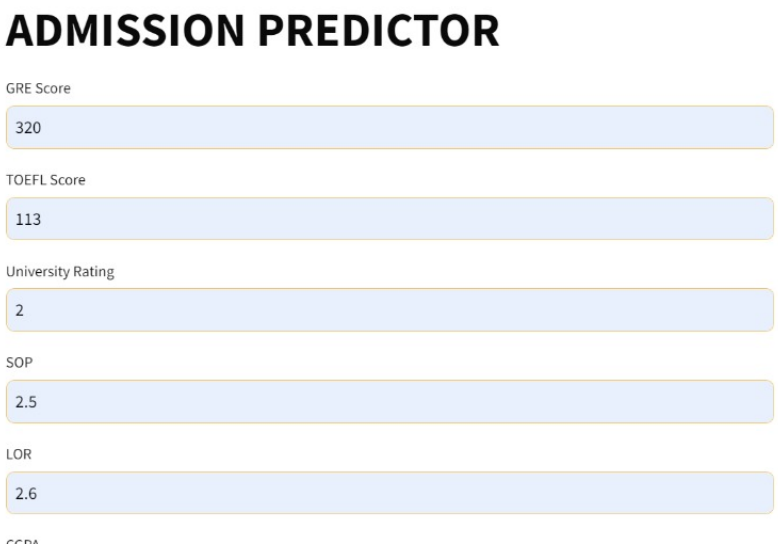

Figure 6.8 Application User Interface

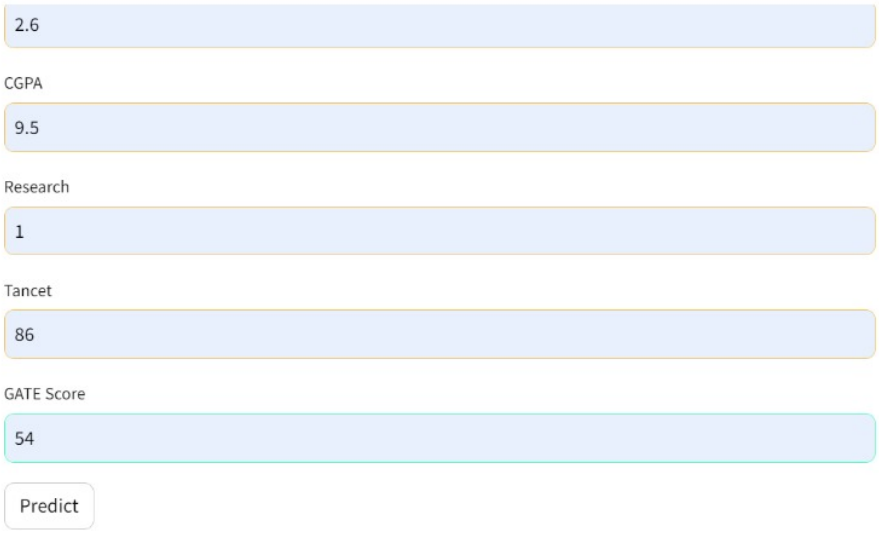

Figure 6.9 Input Box in App

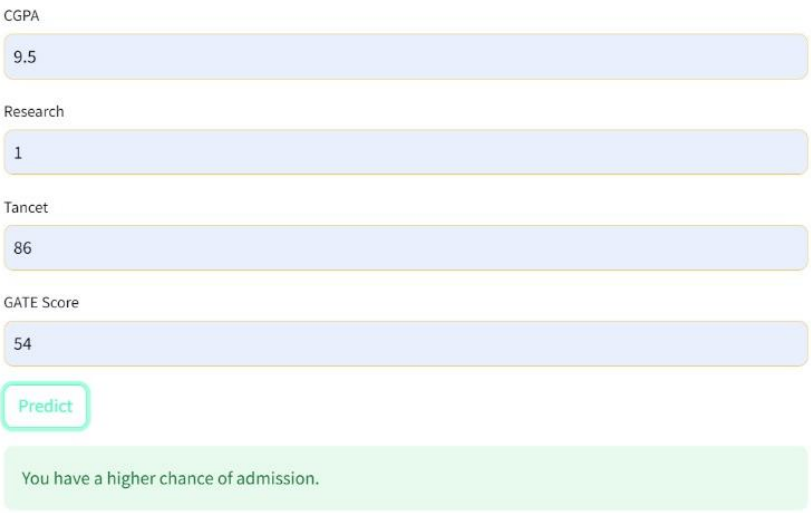

Figure 6.10 Positive Result Displayed

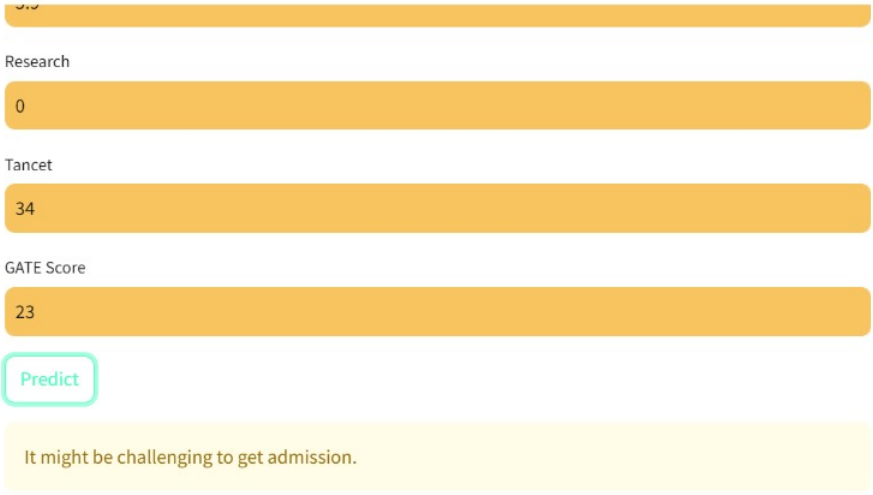

Figure 6.11 Negative Result Displayed

CONCLUSION

In this study, we developed a University Admission Predictor using machine learning algorithms, including Linear Regression, Decision Tree, K-Nearest Neighbors (KNN), and Random Forest Regression. By analyzing a training dataset containing various parameters relevant to university admissions, such as GRE score, TOEFL score, University rank, SOP, LOR, CGPA, and number of research papers, we evaluated each model's performance using metrics like Normalized Absolute Error (NAE), Normalized Squared Error (NSE), Root Mean Squared Error (RMSE), and R-squared (R2) score.

After thorough model evaluation, we found that the Decision Tree algorithm performed efficiently and was selected as the best-performing model based on the highest combined score across the evaluation metrics. Subsequently, the Decision Tree model underwent training on the entire dataset to maximize predictive accuracy.

The predictor was then deployed on the Streamlit platform, enabling users to input their parameters and receive a probability estimate indicating the likelihood of admission to the university.

This predictor not only provides valuable insights into university admissions processes, assisting prospective students in making informed decisions regarding their application strategies, but also serves as a practical demonstration of machine learning techniques applied to real-world scenarios. It highlights the potential for data-driven solutions in education and beyond, showcasing the power of machine learning in solving complex problems.

REFERENCES

[1] J. Katti, J. Agarwal, S. Bharata, S. Shinde, S. Mane and V. Biradar, "University Admission Prediction Using Google Vertex AI," 2022 First International Conference on Artificial Intelligence Trends and Pattern Recognition (ICAITPR).

[2] X. Liu, Y. Wang, Z. Zhang, B. Chai, Y. Zhou and S. Zhang, "The Application of SVR-based Combination Algorithm in Applying for College in China," 2020 IEEE 2nd International Conference on Computer Science and Educational Informatization (CSEI).

[3] Ananya R Patel, Bhuvana K S, Dhanya R S, D A Akshay, Shridhar B. Devamane, and Tushar N. "Machine Learning Based Graduate Admission Prediction." In 2023 International Conference on Computational Intelligence for Information, Security and Communication Applications (CIISCA).

[4] B. Uday Kiran, B. Simon Paul, T. Pavan Kumar, and Mr. M. Sathya Narayana. "Admission Predictor for MS in Foreign University using ML." International Research Journal of Modernization in Engineering Technology and Science.

[5] P K Binu, Ananthu Chandran, and M Rahul. "A Cloud-Based Data Analysis and Prediction System for University Admission." In 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT). IEEE.

[6] R. Sharma, S. Gupta, and A. Singh. "Predictive Modeling for University Admission: A Comparative Study of Machine Learning Approaches." In Proceedings of the International Conference on Data Mining and Big Data (DMBD), 2020.