**THE FUTURE OF UNIVERSITY DECISION MAKING WITH MACHINE LEARNING**

**PROJECT REPORT**

Submitted by

**TEAM ID:** NM2023TMID25456

**RAMASAMY**

**SUMITHRA**

**VINITHKUMAR**

**SOKKALINGAM**

In partial fulfilment of the requirements for the award of the degree of Bachelor of computer science of Bharathiar University,Coimbatore -46.



Under the Guidance of

**Dr.A.GEETHA**

Assistant Professor

**DEPARTMENT OF COMPUTER SCIENCE**

**GOVERNMENT ARTS AND SCIENCE COLLEGE (CO-ED)**

(Affiliated to Bharathiar University, Coimbatore)

**AVINASHI-641 654**

**NAAN MUDHALVAN PROJECT WORK**

**GOVERNMENT ARTS AND SCIENCE COLLEGE (CO-ED)**

(Affiliated to Bharathiar University, Coimbatore)

**AVINASHI-641 654**

**THE FUTURE OF UNIVERSITY DECISION MAKING WITH MACHINE LEARNING**

This is to certify that this is a bonafide record of work done by the above students of III B.Sc. (CS) Degree **NAAN MUDHALVAN PROJECT** during the year 2022-2023

Submitted for the Naan Mudhalvan  project work held on………….20

**Class Mentor Head of Department**

**Dr.A.GEETHA Prof. B. HEMALATHA**

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER.NO** | **PARTICULARS** | **PAGE. NO** |
|  | **INTRODUCTION** |  |
| **1** | 1.1 Overview | 04 |
|  | 1.2Purpose |  |
|  | **PROBLEM DEFINITION & DESIGN THINKING** |  |
| **2** | 2.1 Empathy Map | 06 |
|  | 2.2 Ideation & Brainstorming Map |  |
| **3** | **RESULT** | 07 |
| **4** | **ADVANTAGE & DISADVANTAGE** | 08 |
| **5** | **APPLICATION** | 08 |
| **6** | **CONCLUSION** | 01 |
| **7** | **FUTURE SCOPE** | 10 |
| **8** | **APPENDIX** | 11 |
|  | A .SORCE CODE |  |

**1. INTRODUCTION**

**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. 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

**1.2 Purpose**

* User interacts with the UI to enter the input.
* Entered input is analysed by the model which is integrated.
* Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

* Define Problem / Problem Understanding

1. Specify the business problem
2. Business requirements
3. Literature Survey
4. Social or Business Impact.

* Data Collection & Preparation

1. Collect the dataset
2. Data Preparation

* Exploratory Data Analysis

1. Descriptive statistical
2. Visual Analysis

* Model Building

1. Training the model in multiple algorithms
2. Testing the model

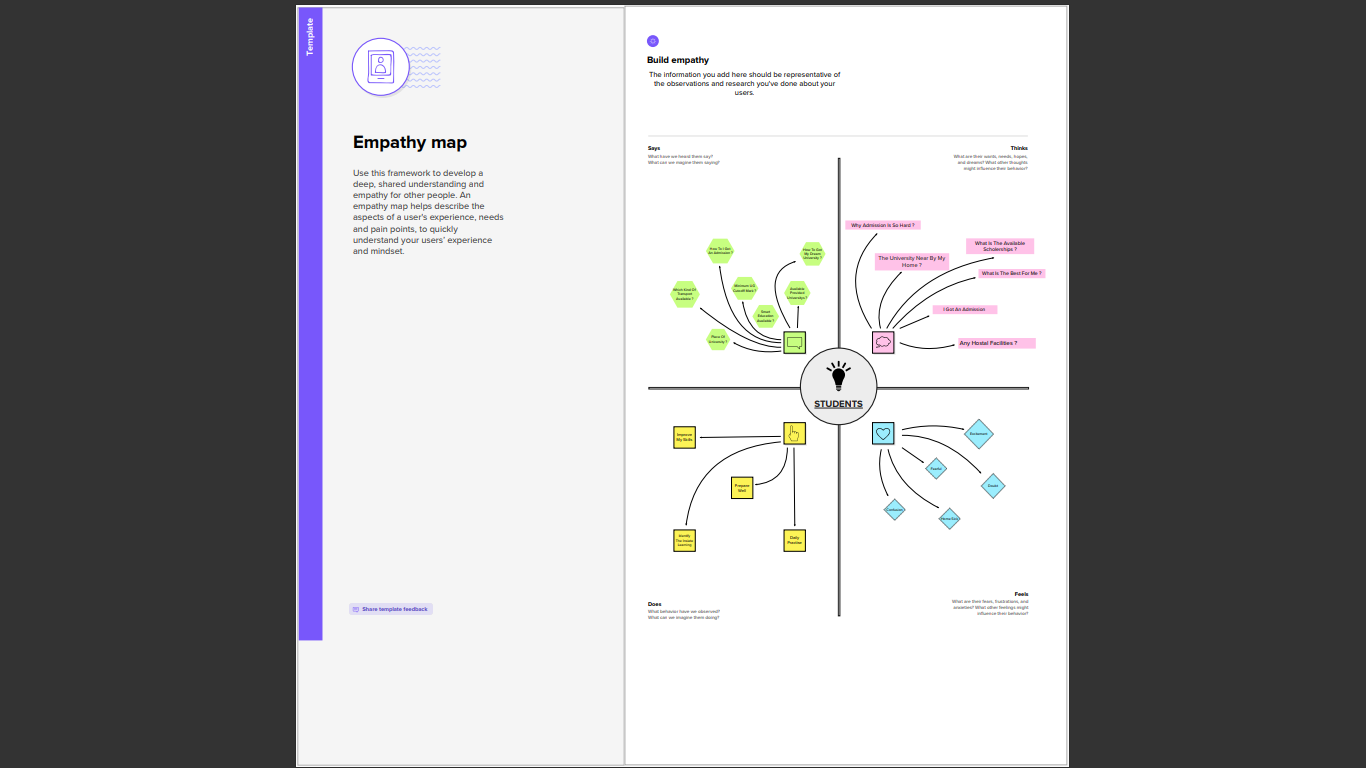
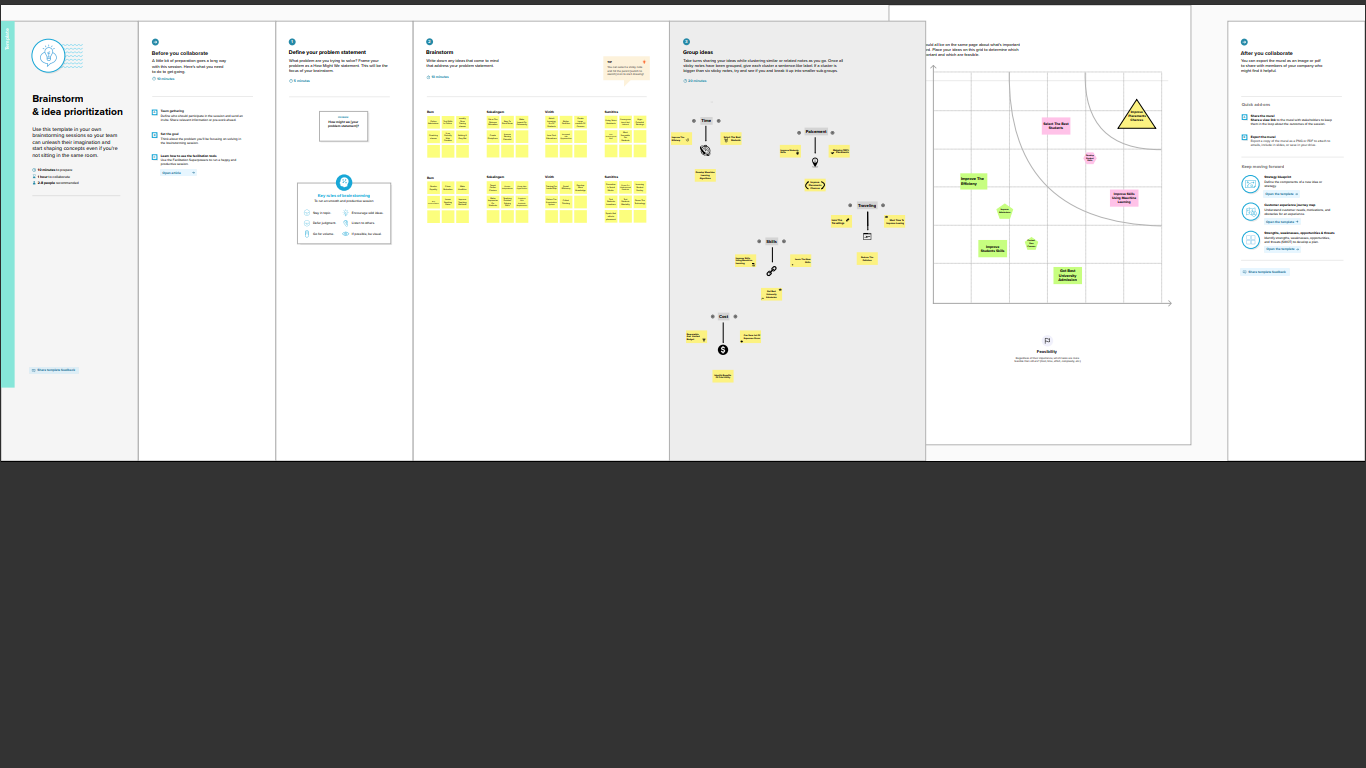
* Performance Testing & Hyperparameter Tuning

1. Testing model with multiple evaluation metrics
2. Comparing model accuracy before & after applying hyperparameter tuning

* Model Deployment

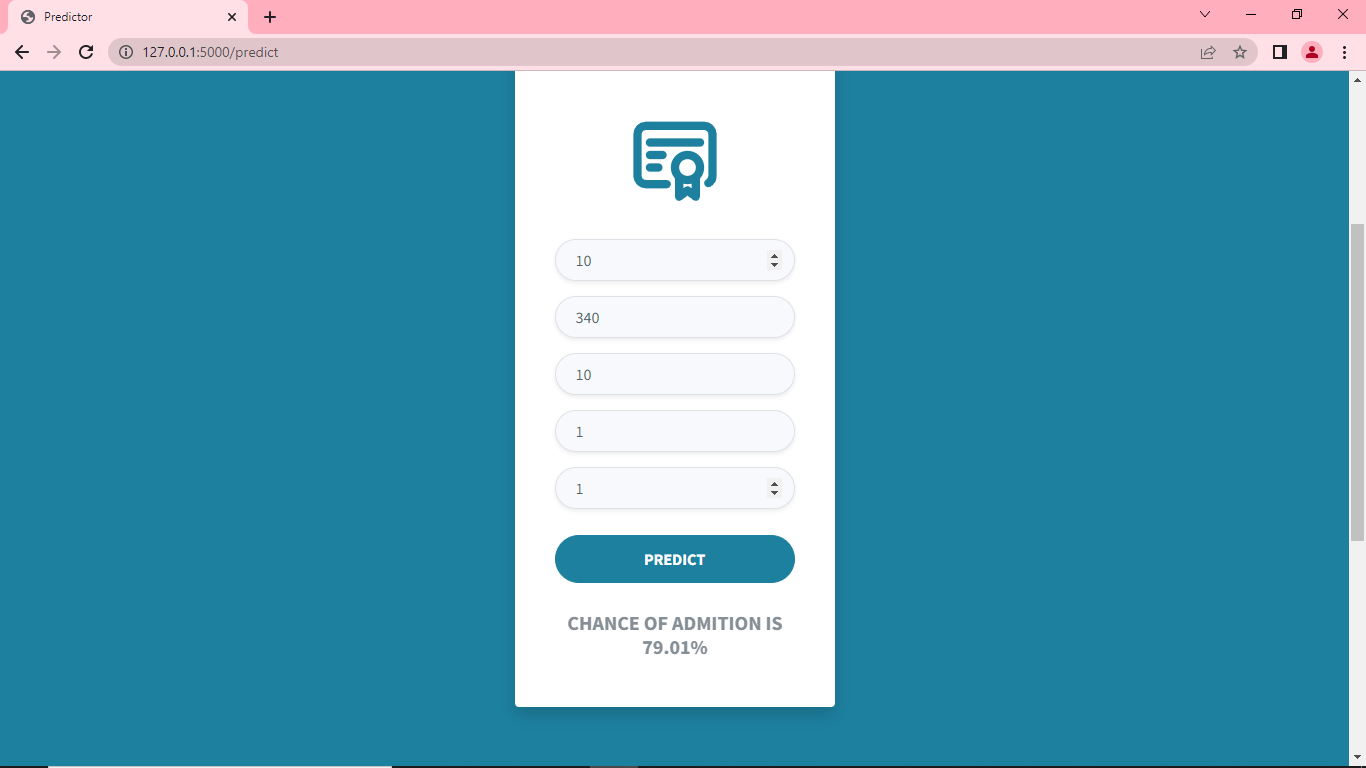
1. Save the best model
2. Integrate with Web Framework

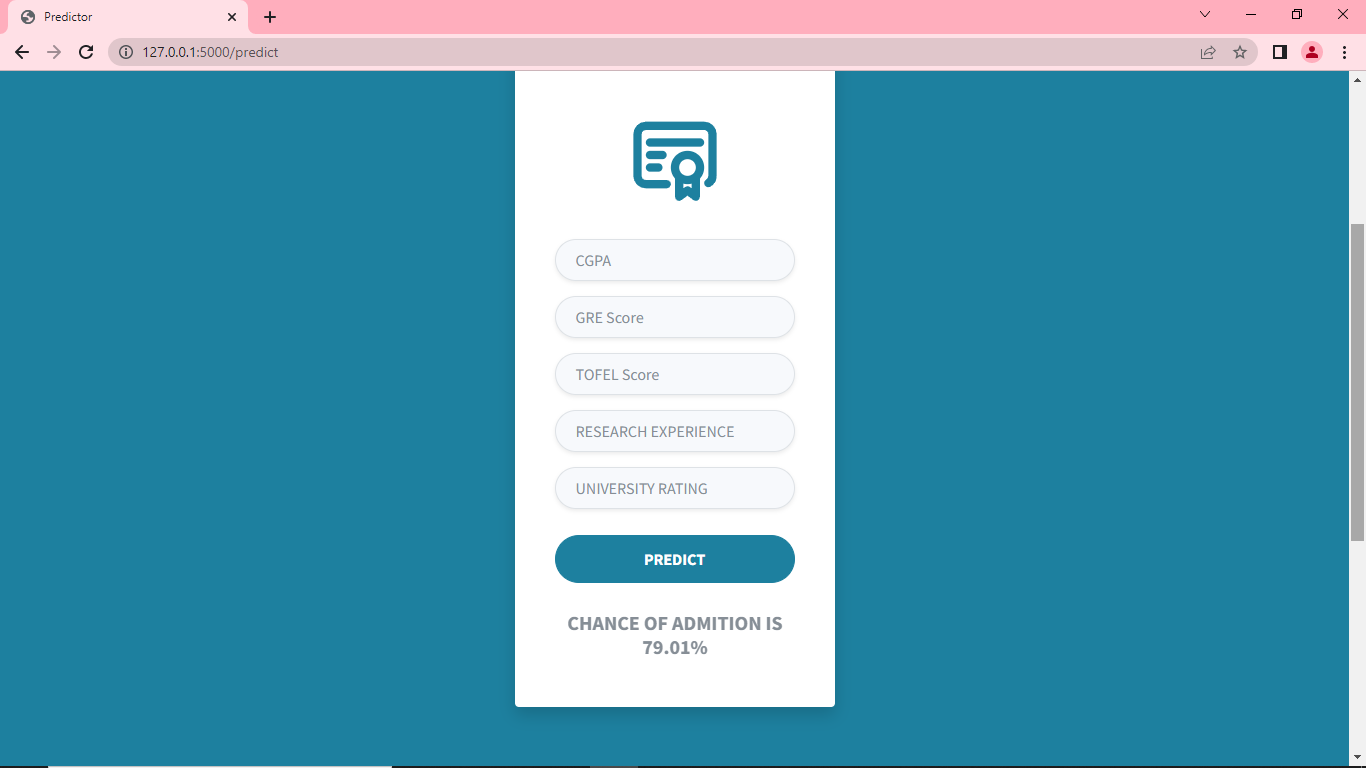
* Project Demonstration & Documentation

1. Record explanation Video for project end to end solution
2. Project Documentation-Step by step project development procedure
3. **PROBLEM DEFINITION & DESIGN THINKING**
   1. **Empathy Map**
   2. **Ideation & Brainstorming Map**

**3. RESULT**

**Home page**

****

**Predicting page of Flight Price Prediction**

# 4. ADVANTAGES AND DISADVANTAGES

## Advantages

1. The Average Lifetime Earnings with a university degree are a million\_\_more than a person with only a high school degree.
2. Saves time in searching / deciding for Education.

**Disadvantages**

1. Contact time is less than at school.

**5. APPLICATIONS**

1. Make Admission easier

2. Admission Tracking

3. Admission prediction

**6. CONCLUSION**

The college admission predictor uses historical colleges cut-off students admission data for predicting the most probable colleges. The system analyzes student academic merits, background, and college admission criteria. Based on that, it predicts the likelihood of a university college that a student may enter. It helps student for making decision for choosing a right college. Here the chance of occurrence of error is less when compared with the existing system. It is fast, efficient and reliable. Avoids data redundancy and inconsistency.

# 7. FUTURE SCOPE

1. Simplified Admission Process. The admission process of any institute is one of the most important things which creates an impression on students.
2. Alumni Testimonials & Alumni Networks. ...
3. Easy & Effective Communication with Students

**8. APPENDIX**

**A .SORCE CODE**

* **app.py**

import numpy as np

from flask import Flask, request, jsonify, render\_template

import pickle

app = Flask(\_\_name\_\_)

#import necessary libraries

from tensorflow import model

model = pickle.load(open("university.pkl","rb"))

#load model trained model

#model your tarined moidel

model = load\_model('model.h5')

@app.route('/')

def home():

return render\_template('Demo2.html')

@app.route('/y\_predict', methods=['POST'])

def 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]

min1 = [340.0, 120.0, 5.0, 5.0, 5.0, 9.92, 1.0]

k = [float(x) for x in requst.form.values()]

p = {}

for i in range(7):

l = (k[i]-min[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 Getting Admissions')

else:

return render\_template('chance.html',prediction\_text = 'You have a chance of Getting Admissions')

if \_\_name\_\_=="\_\_main\_\_":

app.run(debug=False)

* **model.py**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

%matplotlib inline

data = pd.read\_csv("Admission\_Predict.csv")

data.shape

data.info()

data.isnull().any()

data.isnull().sum()

data.describe()

data.hist(bins = 30, figsize = (20,20), color = 'lightskyblue')

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()

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

sns.heatmap(data.corr())

sns.pairplot(data)

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

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

data.columns

x = data.drop(columns = ['Chance of Admit '])

y = data ['Chance of Admit ']

x.shape

y.shape

x

y

x = np.array(x)

y = np.array(y)

y = y.reshape(-1,1)

y.shape

from sklearn.preprocessing import StandardScaler, MinMaxScaler

scaler\_x = StandardScaler()

x = scaler\_x.fit\_transform(x)

x

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

x

y = data.iloc[:,7:].values

y

scaler\_y = StandardScaler()

y = scaler\_y.fit\_transform(y)

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.70,random\_state=101)

y\_train = (y\_train>0.2)

y\_train

scaler\_y = StandardScaler()

y = scaler\_y.fit\_transform(y)

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.70,random\_state=101)

y\_train = (y\_train>0.2)

y\_train

y\_test = (y\_test>0.2)

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, accuracy\_score

regressor = LinearRegression()

regressor.fit(x\_train, y\_train)

accuracy\_LinearRegression = regressor.score(x\_test, y\_test)

accuracy\_LinearRegression

import tensorflow

from tensorflow import keras

from tensorflow.keras.layers import Dense,Activation,Dropout

from tensorflow.keras.optimizers import Adam

ANN\_model = keras.Sequential()

ANN\_model.add(Dense(50, input\_dim = 7))

ANN\_model.add(Activation('relu'))

ANN\_model.add(Dense(150))

ANN\_model.add(Activation('relu'))

ANN\_model.add(Dropout(0.5))

ANN\_model.add(Dense(150))

ANN\_model.add(Activation('relu'))

ANN\_model.add(Dropout(0.5))

ANN\_model.add(Dense(50))

ANN\_model.add(Activation('linear'))

ANN\_model.add(Dense(1))

ANN\_model.compile(loss = 'mse', optimizer = 'adam')

ANN\_model.summary()

model=keras.Sequential()

model.add(Dense(7,activation ='relu',input\_dim=7))

model.add(Dense(7,activation ='relu'))

model.add(Dense(1,activation ='linear'))

model.summary()

ANN\_model.compile(optimizer = 'Adam', loss = 'mean\_squared\_error')

epochs\_hist = ANN\_model.fit(x\_train, y\_train, epochs = 100, batch\_size = 20)

result = ANN\_model.evaluate(x\_test, y\_test)

accuracy\_ANN = 1 - result

print("Accuracy : {}".format(accuracy\_ANN))

epochs\_hist.history.keys()

plt.plot(epochs\_hist.history['loss'])

plt.title('Model Loss Progress during Training \n')

plt.xlabel('Epoch')

plt.ylabel('Training Loss')

plt.legend(['Training Loss'])

from sklearn.tree import DecisionTreeRegressor

DecisionTree\_model = DecisionTreeRegressor()

DecisionTree\_model.fit(x\_train, y\_train)

accuracy\_DecisionTree = DecisionTree\_model.score(x\_test, y\_test)

accuracy\_DecisionTree

from sklearn.ensemble import RandomForestRegressor

RandomForest\_model = RandomForestRegressor(n\_estimators = 100, max\_depth = 10)

RandomForest\_model.fit(x\_train, y\_train)

accuracy\_RandomForest = RandomForest\_model.score(x\_test, y\_test)

print(accuracy\_RandomForest)

y\_predict = regressor.predict(x\_test)

plt.plot(y\_test, y\_predict, '^', color = 'y')

y\_predict\_orig = scaler\_y.inverse\_transform(y\_predict)

y\_test\_orig = scaler\_y.inverse\_transform(y\_test)

plt.plot(y\_test\_orig, y\_predict\_orig, '^', color = 'r')

k = x\_test.shape[1]

n = len(x\_test)

n

from sklearn.metrics import r2\_score, mean\_squared\_error, mean\_absolute\_error

from math import sqrt

MSE = mean\_squared\_error(y\_test\_orig, y\_test\_orig)

MAE = mean\_absolute\_error(y\_test\_orig, y\_test\_orig)

RMSE = float(format(np.sqrt(mean\_squared\_error(y\_test\_orig, y\_test\_orig)), '.3f'))

r2 = r2\_score(y\_test\_orig, y\_test\_orig)

print('RMSE = ', RMSE, '\nMSE = ', MSE, '\nMAE = ', MAE, '\nr2 = ', r2)

X = [1,2,3,4]

X = np.array(X)

X = X.reshape(-1,1)

print(X.shape)

import numpy as np

X = [1,2,3,4]

X = np.array(X)

print(X.shape)

import pickle

s = np.array([320, 110, 1, 5, 5, 9, 1])

print(s.shape)

s = s.reshape(1,-1)

print(s.shape)

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

model = pickle.load(open('regressor.pkl', 'rb'))

print(model.predict(s))