UNIVERSITY ADMISSION PREDICTION

INTRODUCTION

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data. Python community has developed many modules to help programmers implement machine learning. In this article, we will be using numpy, scipy and scikit-learn modules. A better option would be downloading miniconda or anaconda packages for python, which come prebundled with these packages.

Machine learning involves computer to get trained using a given data set, and use this training to predict the properties of a given new data.

For example, we can train computer by feeding it 1000 images of cats and 1000 more images which are not of a cat, and tell each time to computer whether a picture is cat or not. Then if we show the computer a new image, then from the above training, computer should be able to tell whether this new image is cat or not.

This is about Analysis of the Models performing on a particular Data Set

In machine learning we have three different types of learning

- SUPERVISED LEARNING
- UNSUPERVISED LEARNING
- REINFORCEMENT LEARNING

so basically in machine learning we work mainly on SUPERVISED LEARNING. when we talk about supervised learning there are classified into two models:

- REGRESSION MODELS
- CLASSIFICATION MODELS

REGRESSION MODELS are further classified into:

- SIMPLE LINEAR REGRESSION
- MULTI LINEAR REGRESSION
- POLYNOMIAL REGRESSION
- DECISION TREE REGRESSION
- RANDOM FOREST REGRESSION

CLASSIFICATION MODELS are classified into:

- LOGISTIC REGRESSION
- KNN
- SUPPORT VECTOR MACHINE
- NAÏVE BAYES
- DECISION TREE CLASSIFICATION
- RANDOM FOREST CLASSIFICATION

OBJECTIVES OF RESEARCH

The Objective of the research project is that we should be able to apply all the regression models and classification models for a given data set and after performing the different models on data set each model should return the accuracy (r2_score), mean squared error and mean absolute error so based on the values obtained for each of the model. We will be choosing that particular model where accuracy of the model is high if the accuracy of the model is high then we choose that particular model for that dataset.

The main aspect of this research is to find the chance of admission of a particular student based on the inputs which has been asked to the user. When the user gives the required inputs it should predict the chance of admission which will be represented in percentage.

Then we create a user interface for the problem statement with the help of NODE RED which is a part of IBM Watson studio.

PROBLEM STATEMENT

The problem statement is a University Admission Predictor

In this case we already have a dataset which has having almost 500 rows of data and there are 8 columns. The 8 columns are

1. GRE SCORE
2. TOFEL SCORE
3. UNIVERSITY RATING
4. STATEMENT OF PURPOSE
5. LETTER OF RECOMMENDATION
6. CGPA
7. RESEARCH
8. CHANCE OF ADMIT

So these are the 8 columns which are there in the dataset.

Since after observing the dataset we can get to know that the dataset does not have any categorical data and it has only continuous numerical data.

So with this observation from the dataset we can conclude that basically there are two different models in supervised learning. The two models are the regression models and classification models. So we use regression models only when there is a continuous numerical data and we use classification model whenever there is a categorical data from this understanding we can decide that we require only regression models for the kind of dataset we have so we apply all the regression models available for the dataset and find the accuracy, mean square error and mean absolute error so once we get values for all the models whichever model is having a higher accuracy we will choose that model and that will be the solution for the problem statement.

To develop a user interface for the above model with the help of NODE RED

REVIEW OF LITERATURE

The regression models which we applied for the dataset are:

1. MULTIPLE LINEAR REGRESSION
2. POLYNOMIAL REGRESSION
3. RANDOM FOREST REGRESSION
4. KNN
5. DECISION TREE REGRESSION

When Multiple Linear regression is performed for the dataset we had made some observations which can be used for finding the best model applicable for the dataset

Accuracy (r2_score) = 0.7699

Mean_squared_error = 0.0041

Mean_absolute_error = 0.0461

When Polynomial regression is performed for the dataset we had made some observations which can be used for finding the best model applicable for the dataset

Accuracy (r2_score) = 0.7140

Mean_squared_error = 0.0051

Mean_absolute_error = 0.0519

When Random Forest regression is performed for the dataset we had made some observations which can be used for finding the best model applicable for the dataset

Accuracy (r2_score) = 0.8036

Mean_squared_error = 0.0037

Mean_absolute_error = 0.0428

When KNN is performed for the dataset we had made some observations which can be used for finding the best model applicable for the dataset

Accuracy (r2_score) = 0.7680

Mean_squared_error = 0.0043

Mean_absolute_error = 0.0478

When Decision Tree regression is performed for the dataset we had made some observations which can be used for finding the best model applicable for the dataset

Accuracy (r2_score) = 0.6016

Mean_squared_error = 0.0075

Mean_absolute_error = 0.0618

So from the observations we have concluded that Random forest Regression is the best model for the given dataset because of the best accuracy among the above models.

DATA COLLECTION

The dataset for the given problem statement was obtained from the online source. The dataset was directly taken from Kaggle.com

The data present in the dataset is shown in the below screenshot.

Serial No.	GREScore	TOEFLScor	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

METHODOLOGY

4.1 EXPLORATORY DATA ANALYSIS

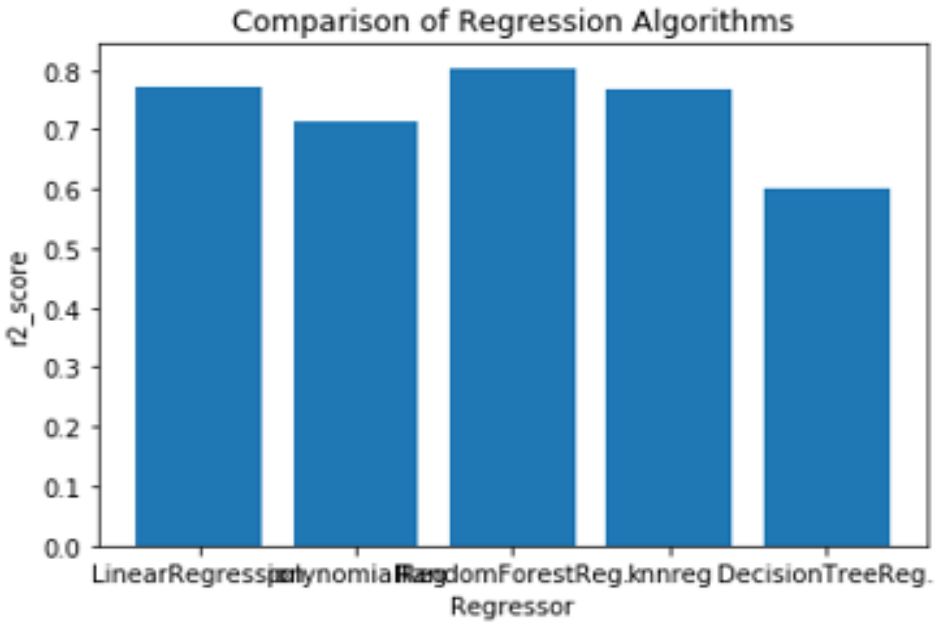
READING AND ANALYZING DATA

```
In [2]: df = pd.read_csv("data-clean.csv",sep = ",")
        print(df.info())
```

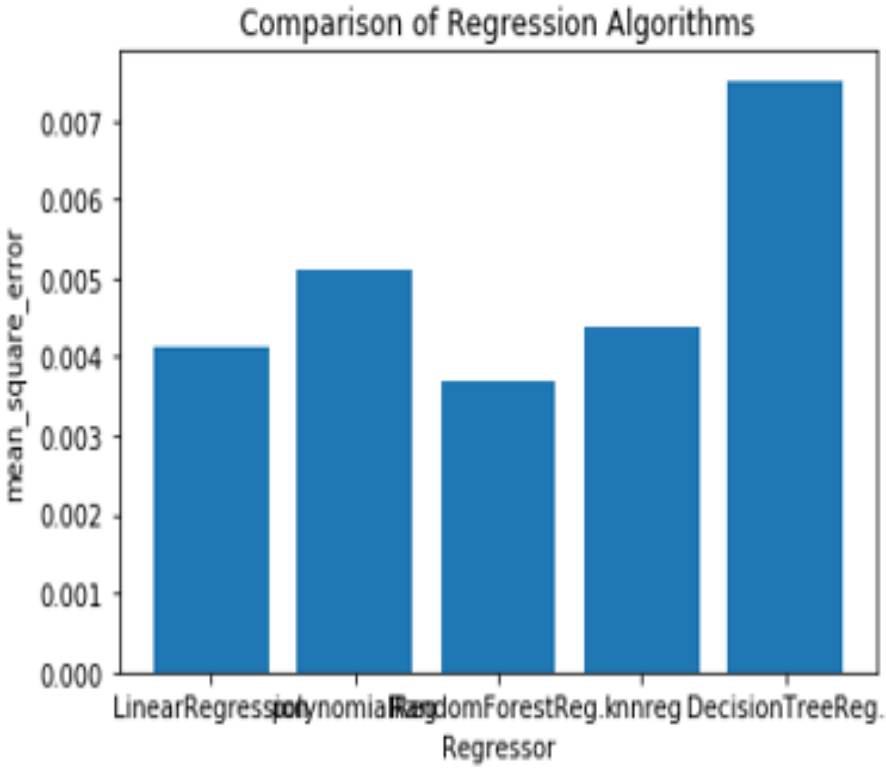
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
Serial No.          500 non-null int64
GRE Score           500 non-null int64
TOEFL Score         500 non-null int64
University Rating   500 non-null int64
SOP                 500 non-null float64
LOR                 500 non-null float64
CGPA                500 non-null float64
Research            500 non-null int64
Chance of Admit     500 non-null float64
dtypes: float64(4), int64(5)
memory usage: 35.2 KB
None
```

4.1.1 FIGURES AND TABLES

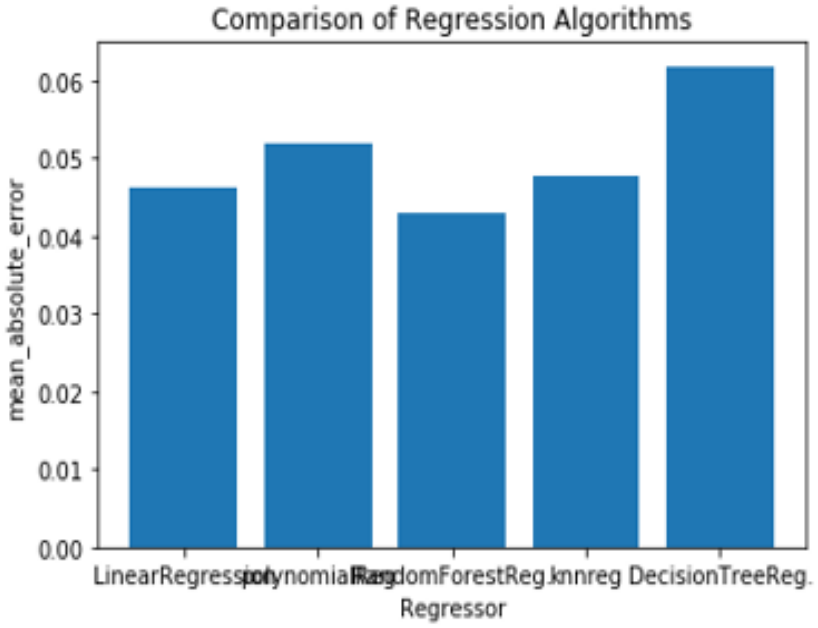
Comparison of regression algorithms with respect to r2_score and regressor



Comparison of regression algorithms with respect to mean_square_error and regressor



Comparison of regression algorithms with respect to mean_absolute_error and regressor



4.1.2 DATA MODELLING

Screenshots of the models performed for the dataset

MULTIPLE LINEAR REGRESSION

In [55]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset=pd.read_csv("data-clean.csv")
X=dataset.iloc[:,1:-1].values
y=dataset.iloc[:,-1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 0)



from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)


from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

y_predMLR = regressor.predict(X_test)


#r2score
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
MLRA=r2_score(y_test, y_predMLR)
MLRMS=mean_squared_error(y_test,y_predMLR)
MLRMA=mean_absolute_error(y_test,y_predMLR)
print(f"r2_score : {MLRA}")
print(f"mean_squared_error : {MLRMS}")
print(f"mean_absolute_error : {MLRMA}")
```

```
r2_score : 0.7699866157538676
mean_squared_error : 0.004113016118289162
mean_absolute_error : 0.04612099326386116
```

POLYNOMIAL REGRESSION

In [56]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset=pd.read_csv("data-clean.csv")
X=dataset.iloc[:,1:-1].values
y=dataset.iloc[:,-1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 0)


# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 2)
X_poly_train = poly_reg.fit_transform(X_train)
X_poly_test=poly_reg.transform(X_test)


from sklearn.linear_model import LinearRegression
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly_train, y_train)


y_predPLR = lin_reg_2.predict(X_poly_test)


from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
PLRA=r2_score(y_test, y_predPLR)
PLRMS=mean_squared_error(y_test,y_predPLR)
PLRMA=mean_absolute_error(y_test,y_predPLR)
print(f"r2_score : {PLRA}")
print(f"mean_squared_error : {PLRMS}")
print(f"mean_absolute_error : {PLRMA}")
```

```
r2_score : 0.7140253476895997
mean_squared_error : 0.005113695267037894
mean_absolute_error : 0.0519740862123894
```

RANDOM FOREST

In [57]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset=pd.read_csv("data-clean.csv")
X=dataset.iloc[:,1:-1].values
y=dataset.iloc[:,-1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestRegressor
RFregressor = RandomForestRegressor(n_estimators = 300, criterion="mse",random_state = 42)
RFregressor.fit(X_train, y_train)

# Predicting the Test set results
y_predRF = RFregressor.predict(X_test)


from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
RFA=r2_score(y_test, y_predRF)
RFMS=mean_squared_error(y_test,y_predRF)
RFMA=mean_absolute_error(y_test,y_predRF)
print(f"r2_score : {RFA}")
print(f"mean_squared_error : {RFMS}")
print(f"mean_absolute_error : {RFMA}")
```

```
r2_score : 0.8036720562132994
mean_squared_error : 0.003700921839999993
mean_absolute_error : 0.04281306666666649
```

KNN

In [58]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset=pd.read_csv("data-clean.csv")
X=dataset.iloc[:,1:-1].values
y=dataset.iloc[:,-1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting K-NN to the Training set
from sklearn.neighbors import KNeighborsRegressor
regressorKNN= KNeighborsRegressor(n_neighbors =7, metric = 'minkowski', p = 2)
regressorKNN.fit(X_train, y_train)

# Predicting the Test set results
y_predKNN = regressorKNN.predict(X_test)


from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
KNNA=r2_score(y_test, y_predKNN)
KNNMS=mean_squared_error(y_test,y_predKNN)
KNNMA=mean_absolute_error(y_test,y_predKNN)
print(f"r2_score : {KNNA}")
print(f"mean_squared_error : {KNNMS}")
print(f"mean_absolute_error : {KNNMA}")
```

```
r2_score : 0.7680611812542758
mean_squared_error : 0.004372212244897958
mean_absolute_error : 0.04786285714285714
```

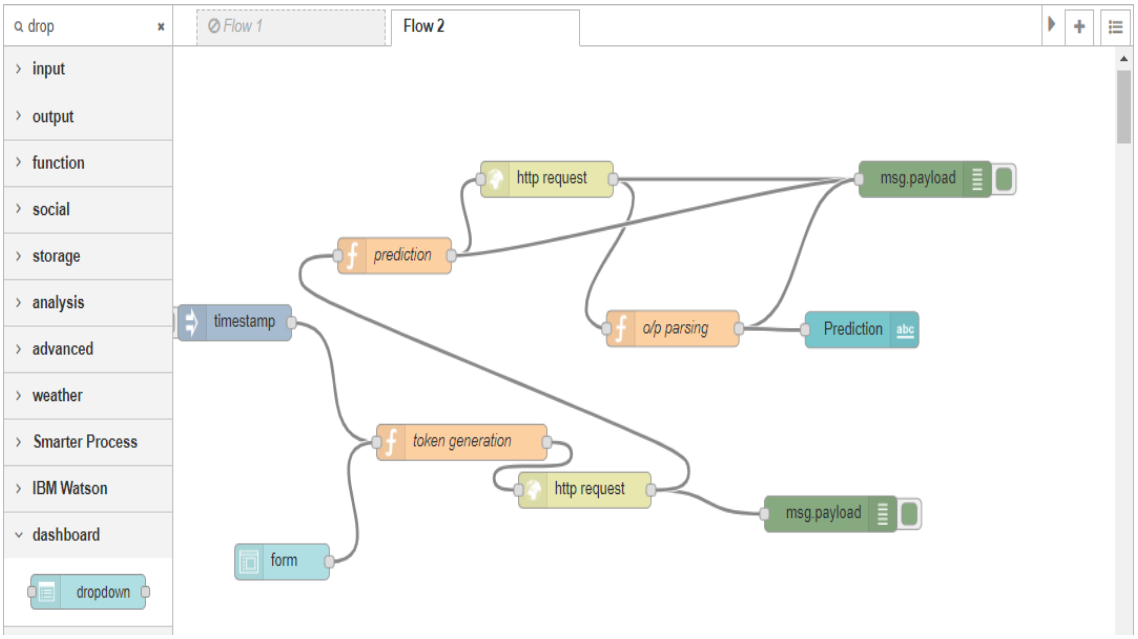
DECISION TREE

In [59]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset=pd.read_csv("data-clean.csv")
X=dataset.iloc[:,1:-1].values
y=dataset.iloc[:,-1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting K-NN to the Training set

# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeRegressor
regressorDT = DecisionTreeRegressor( random_state = 0)
regressorDT.fit(X_train, y_train)

# Predicting the Test set results
y_predDT = regressorDT.predict(X_test)



from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
DTA=r2_score(y_test, y_predDT)
DTMS=mean_squared_error(y_test,y_predDT)
DTMA=mean_absolute_error(y_test,y_predDT)
print(f"r2_score : {DTA}")
print(f"mean_squared_error : {DTMS}")
print(f"mean_absolute_error : {DTMA}")
```

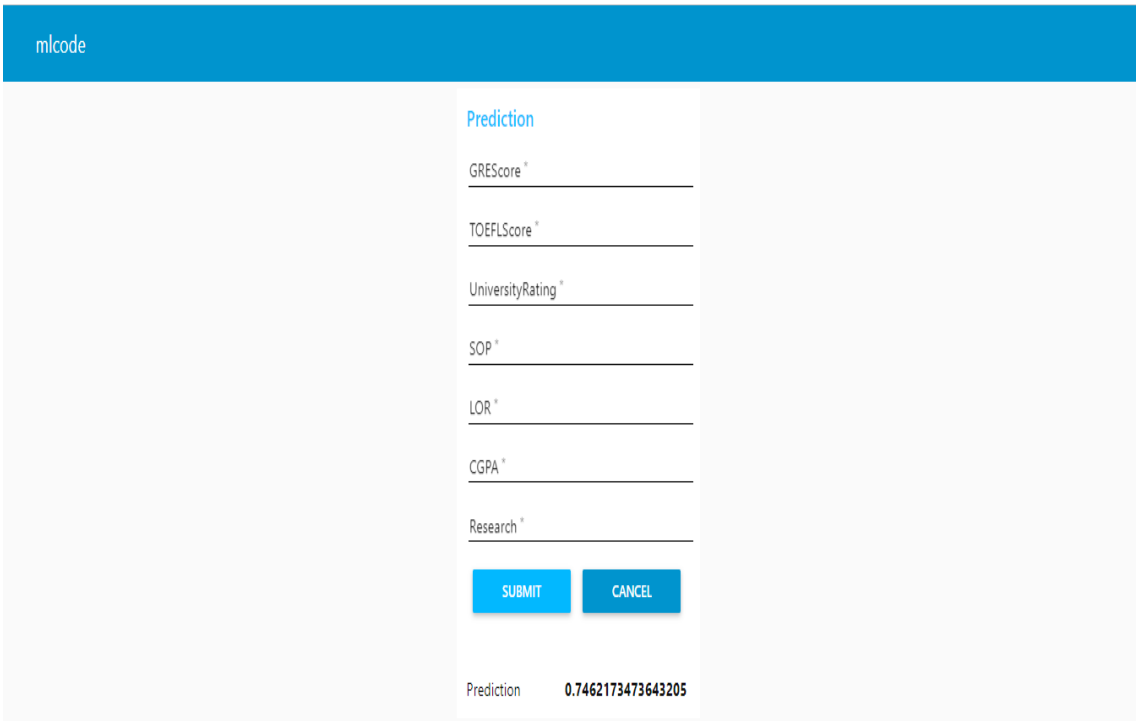
```
r2_score : 0.6016278131773218
mean_squared_error : 0.0075096
mean_absolute_error : 0.06183999999999999
```

USER INTERFACE FOR THE FOLLOWING PROJECT

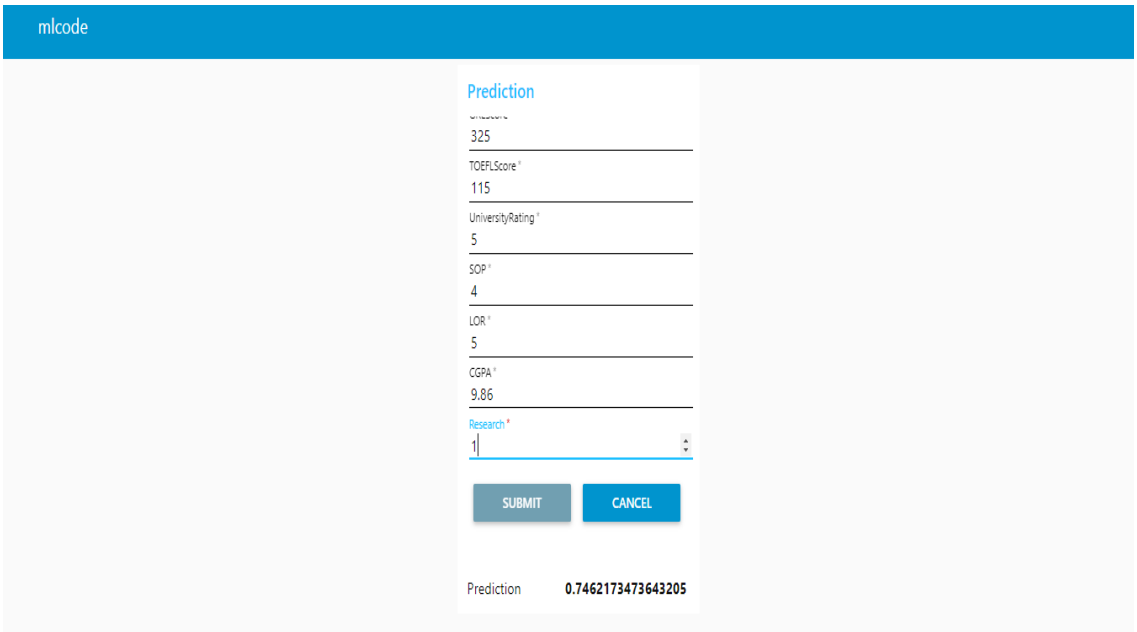
NODE RED FLOW



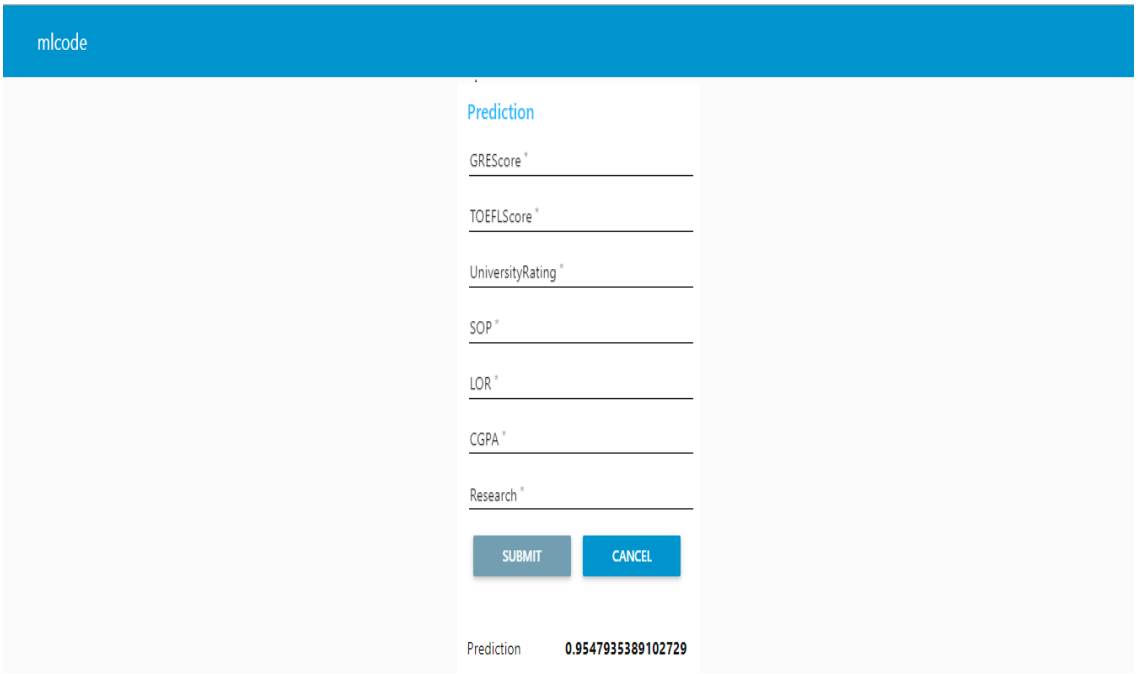
STARTING WEBPAGE OBTAINED



USER FILLING THE FORM



PREDICTION OBTAINED AFTER FILLING THE FORM



FINDINGS AND SUGGESTIONS

FINDINGS

- Data preprocessing is vital to the accuracy of the model.
- Choosing appropriate machine learning techniques and algorithms to model the system.
- Graphical representation of the data provides useful insights and can lead to better models.
- Defining scope with respect to the dataset

SUGGESTIONS

Creating the model with additional parameters such as Work Experience, Technical Papers Written etc. can make it more flexible to the Universities admission requirements. Hence by generalizing the decision making parameters, this system can be used for any admission prediction process by taking into consideration all desired criteria.

CONCLUSION

It is concluded that RANDOM FOREST REGRESSION performs better for the considered dataset.