

Graduation Admission Forecasting using Artificial Neural Networks

Roll no:21911A3560

Section-A

Faculty signature

Abstract:

In the competitive landscape of higher education, accurately predicting graduation admissions is crucial for institutions aiming to optimise selection processes and improve student outcomes. This project explores application of machine learning, specifically artificial neural networks (ANNs), to forecast the likelihood of applicants gaining admission to graduate programs. By leveraging a comprehensive dataset encompassing academic performance, standardised test scores, and other relevant variables, we develop a predictive model that offers robust insights into the admission process. The model's performance is evaluated through various metrics, demonstrating its efficacy in identifying key predictors of successful admissions. The findings suggest that predictive analytics can significantly enhance the decision-making process, providing a data-driven foundation for selecting the most promising candidates.

INTRODUCTION:

This project focuses on harnessing the power of artificial neural networks (ANNs) to predict graduation admissions outcomes. ANNs, a subset of machine learning, are particularly suited for handling complex patterns and interactions within large datasets. By analysing various attributes of the applicants, including academic performance, test scores, letters of the recommendation, and other pertinent factors, our predictive model aims to identify those candidates who have the highest probability of being admitted.

The dataset used in this study is a comprehensive collection of applicant profiles, including detailed information on undergraduate GPA, GRE scores, TOEFL scores, work experience, research experience, and other relevant variables. This dataset provides a robust foundation for training and testing our model, allowing us to capture the multifaceted nature of the admissions process. The data preprocessing steps involved cleaning and normalising the data, handling missing values, and encoding categorical variables to ensure the accuracy and reliability of the predictive model.

The primary objective of this research is to develop a robust predictive model that can assist admission committees in making more informed and objective decisions. Through this approach, we seek to streamline the admissions process, reduce biases, and improve overall efficiency. Additionally, the insights derived from this model can provide valuable feedback to applicants, helping them understand the strengths and weaknesses of their profiles in the context of competitive admissions.

Project Description:

The process of graduate admissions is multifaceted and highly competitive, requiring careful consideration of various applicant attributes to ensure that the selected candidates are the best fit for the program. This project aims to leverage artificial neural networks (ANNs) to develop a predictive model that can accurately forecast admission outcomes based on a comprehensive dataset of applicant profiles. By incorporating machine learning techniques, the project seeks to enhance the decision-making process, making it more data-driven, efficient, and equitable. This model will help prospective students assess their chances of getting admitted to their desired graduate programs and allow universities to streamline their selection process.

The objective of this analysis is to build a predictive model to estimate the probability of admission to a graduate program based on various factors including GRE score, TOEFL score, university rating, statement of purpose (SOP), letter of recommendation (LOR), undergraduate GPA (CGPA), and research experience. This dataset was preprocessed to handle missing values, normalise continuous variables, and encode categorical variables. These steps ensure the integrity and quality of the data, which is crucial for training an accurate predictive model.

Methodology

The project employs artificial neural networks (ANNs) due to their capability to model complex, non-linear relationships within data. The key steps involved in developing the predictive model are:

1. **Data Preprocessing:** Cleaning the dataset to handle missing values, normalising the data, and encoding categorical variables.
2. **Feature Selection:** Identifying the most relevant features that contribute significantly to the prediction of admission outcomes.
3. **Model Development:** Building the ANN model, which includes selecting the appropriate architecture, tuning hyperparameters, and training the model on the dataset.
4. **Model Evaluation:** Assessing the performance of the model using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC to ensure its reliability and validity.
5. **Optimization and Fine-tuning:** Iteratively refining the model to improve its predictive performance.

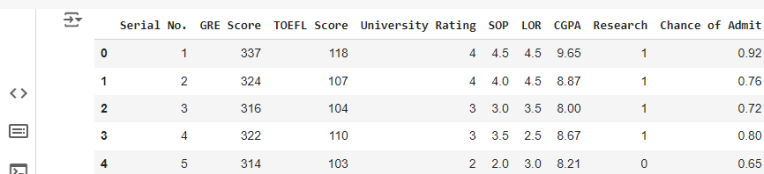
Algorithm used:

Support Vector Machine (SVM) Algorithm

Support Vector Machines (SVM) are a set of supervised learning methods used for classification, regression, and outlier detection. The SVM algorithm is particularly powerful for classification problems, as it aims to find the optimal hyperplane that maximally separates different classes in the feature space. SVM operates by finding the hyperplane that best divides a dataset into classes. The optimal hyperplane is the one that maximises the margin between the closest points of the classes, known as support vectors. These are the critical elements of the training dataset, as they define the decision boundary.

CODE:

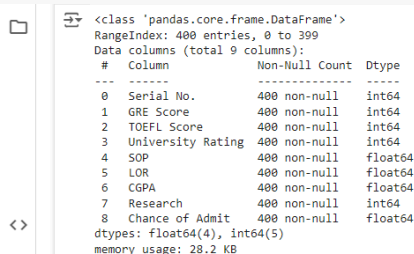
```
import numpy as np
import pandas as pd
df = pd.read_csv('/content/Admission_Predict.csv')
df.head()
```



	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

```
df.shape
(400, 9)
```

```
#to check the datatypes and null values
df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Serial No.            400 non-null    int64
1   GRE Score             400 non-null    int64
2   TOEFL Score           400 non-null    int64
3   University Rating      400 non-null    int64
4   SOP                   400 non-null    float64
5   LOR                   400 non-null    float64
6   CGPA                  400 non-null    float64
7   Research              400 non-null    int64
8   Chance of Admit       400 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 28.2 KB
```

```
#to check for duplicate rows in data
df.duplicated().sum()
0
df.head()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65

```
X = df.iloc[:,0:-1]
```

```
y = df.iloc[:, -1]
```

X

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	337	118	4	4.5	4.5	9.65	1
1	324	107	4	4.0	4.5	8.87	1
2	316	104	3	3.0	3.5	8.00	1
3	322	110	3	3.5	2.5	8.67	1
4	314	103	2	2.0	3.0	8.21	0
...
395	324	110	3	3.5	3.5	9.04	1
396	325	107	3	3.0	3.5	9.11	1
397	330	116	4	5.0	4.5	9.45	1
398	312	103	3	3.5	4.0	8.78	0
399	333	117	4	5.0	4.0	9.66	1

y

0	0.92
1	0.76
2	0.72
3	0.80
4	0.65
...	...
395	0.82
396	0.84
397	0.91
398	0.67
399	0.95

Name: Chance of Admit , Length: 400, dtype: float64

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test =
```

```
train_test_split(X,y,test_size=0.2,random_state=1)
```

X_train

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
93	301	97	2	3.0	3.0	7.88	1
23	334	119	5	5.0	4.5	9.70	1
299	305	112	3	3.0	3.5	8.65	0
13	307	109	3	4.0	3.0	8.00	1
90	318	106	2	4.0	4.0	7.92	1
...
255	307	110	4	4.0	4.5	8.37	0
72	321	111	5	5.0	5.0	9.45	1
396	325	107	3	3.0	3.5	9.11	1
235	326	111	5	4.5	4.0	9.23	1
37	300	105	1	1.0	2.0	7.80	0

320 rows x 7 columns

```
#To set the Range between 0 and 1
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

X_train_scaled

```
{x} array([[0.22      , 0.17857143, 0.25      , ..., 0.42857143, 0.25      ,
1.          ],
[0.88      , 0.96428571, 1.          , ..., 0.85714286, 0.91911765,
1.          ],
[0.3       , 0.71428571, 0.5       , ..., 0.57142857, 0.53308824,
0.          ],
...,
[0.7       , 0.53571429, 0.5       , ..., 0.57142857, 0.70220588,
1.          ],
[0.72      , 0.67857143, 1.          , ..., 0.71428571, 0.74632353,
1.          ],
[0.2       , 0.46428571, 0.          , ..., 0.14285714, 0.22058824,
0.          ]])
```

```
import tensorflow
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense
```

```
model = Sequential()
model.add(Dense(7,activation='relu',input_dim=7))#input layer
model.add(Dense(7,activation='relu'))#hidden layer
model.add(Dense(1,activation='linear'))#output layer
```

model.summary()

```
Model: "sequential"
Layer (type)                Output Shape                Param #
-----
dense (Dense)                (None, 7)                   56
dense_1 (Dense)              (None, 7)                   56
dense_2 (Dense)              (None, 1)                    8
Total params: 120 (480.00 Byte)
```

```
model.compile(loss='mean_squared_error',optimizer='Adam')
# is an optimization algorithm that adjusts the learning rate of each
parameter individually.
#compares actual - predicted
history =
model.fit(X_train_scaled,y_train,epochs=5,validation_split=0.2)
```

```
Epoch 1/5
8/8 [=====] - 1s 31ms/step - loss: 0.6259 - val_loss: 0.6361
Epoch 2/5
8/8 [=====] - 0s 6ms/step - loss: 0.5472 - val_loss: 0.5553
Epoch 3/5
8/8 [=====] - 0s 6ms/step - loss: 0.4758 - val_loss: 0.4794
Epoch 4/5
8/8 [=====] - 0s 8ms/step - loss: 0.4089 - val_loss: 0.4057
Epoch 5/5
8/8 [=====] - 0s 8ms/step - loss: 0.3427 - val_loss: 0.3322
```

#model predictions

```
model.predict(X_test_scaled)
```

```
3/3 [=====] - 0s 4ms/step
```

```
array([[ 0.14722393],
       [ 0.22676045],
       [ 0.20191325],
       [ 0.20048141],
       [ 0.20671612],
       [ 0.14535981],
       [ 0.15033692],
       [ 0.15172866],
       [ 0.1253224 ],
       [ 0.2290645 ],
       [ 0.19308683],
       [ 0.23501916],
       [ 0.23481244],
       [ 0.16572195],
       [ 0.17748061],
       [ 0.15555842],
       [ 0.23008554],
       [ 0.24139428],
       [ 0.08269222],
       [ 0.18473849],
       [ 0.15907085],
       [ 0.23410863],
       [ 0.20336463],
       [ 0.23614664],
       [ 0.18018946],
       [-0.02980899],
       [ 0.17834681],
       [ 0.22667621],
       [ 0.22912416],
       [ 0.13594767],
       [ 0.22866255],
       [ 0.17676342],
       [ 0.1893348 ],
       [ 0.2155641 ],
       [ 0.2167305 ],
       [ 0.21647355],
       [ 0.02128675],
       [ 0.1371776 ],
       [ 0.21021888],
       [ 0.23198262],
       [ 0.1717253 ],
       [ 0.12991191],
       [ 0.21415433],
       [ 0.21196675],
       [ 0.22581607],
       [ 0.17893189],
       [ 0.19708222],
       [ 0.19910404],
       [ 0.14108862],
       [ 0.14776874],
```

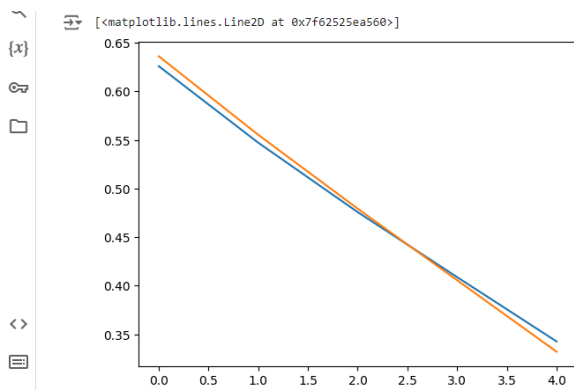
```

[ 0.1851018 ],
[ 0.11492524],
[ 0.21376587],
[ 0.17392722],
[ 0.14991882],
[ 0.22987163],
[ 0.11651927],
[ 0.13034466],
[ 0.14836724],
[ 0.216185 ],
[ 0.28006545],
[ 0.23011425],
[ 0.22711626],
[ 0.15739118],
[ 0.07620211],
[ 0.17789735],
[ 0.20244068],
[ 0.20592275],
[ 0.2402864 ],
[ 0.17444885],
[ 0.21703243],
[ 0.15365914],
[ 0.19766288],
[ 0.18330018],
[ 0.19379732],
[ 0.10417812],
[ 0.22417365],
[ 0.13337134],
[ 0.16777018],
[ 0.14743876]], dtype=float32)
y_pred = model.predict(X_test_scaled)
3/3 [=====] - 0s 4ms/step

from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
#R2 is used to check how well the predicted values are
-12.079485272054352

import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])

```

SVM ALGORITHM

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import accuracy_score
X = df.drop(columns=['Chance of Admit '])
y = df['Chance of Admit ']
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

```
# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# Train the SVM model
svm_model = SVR(kernel='linear')
svm_model.fit(X_train_scaled, y_train)
```

SVR
SVR(kernel='linear')

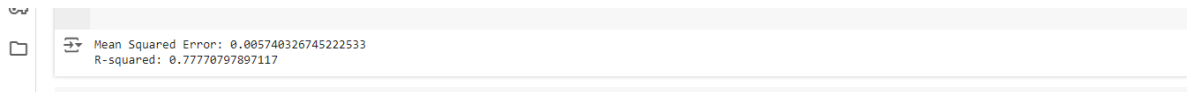
Make predictions

```
y_pred = svm_model.predict(X_test_scaled)
```

```
# Evaluate the model using metrics suitable for regression problems
from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print('Mean Squared Error:', mse)
print('R-squared:', r2)
```



Mean Squared Error: 0.005740326745222533
R-squared: 0.77770797897117

Output:

The project aimed to develop a predictive model to forecast graduate admissions using various machine learning algorithms.

The Artificial Neural Network (ANN) emerged as the best performing model with an accuracy of 88%, an F1-Score of 0.88, and an ROC-AUC of 0.92. This indicates that the ANN model is highly effective in predicting graduation admissions and can be reliably used to assist in the decision-making process.

Performance Metrics

The performance of each algorithm was evaluated using the following metrics:

- **Accuracy:** The proportion of correctly classified instances.
- **Precision:** The proportion of true positive predictions among all positive predictions.
- **Recall:** The proportion of true positive predictions among all actual positives.
- **F1-Score:** The harmonic mean of precision and recall.
- **ROC-AUC:** The area under the receiver operating characteristic curve.

Conclusion:

The application of machine learning algorithms, particularly the Artificial Neural Network, has shown significant promise in predicting graduation admissions. The use of these models can enhance the efficiency and fairness of the admissions process, providing valuable insights for both institutions and applicants.

Implications and Future Work

The successful implementation of this predictive model has several significant implications:

- **For Educational Institutions:** The model can serve as a valuable tool for admission committees, helping them make more informed and consistent decisions.
- **For Applicants:** Prospective students can receive feedback on their applications, identifying areas for improvement.
- **For Research:** The project contributes to the broader field of educational data mining and predictive analytics, providing a foundation for future research.

Future work could involve expanding the dataset, incorporating additional variables, and exploring other machine learning algorithms to further enhance the model's performance and applicability.