ADMISSION PREDICTION USING MACHINE LEARNING

ABSTRACT:

This project aims to develop a machine learning model to predict the likelihood of a student's admission to a graduate program based on various factors. The model uses historical admission data and applies various machine learning algorithms to identify patterns and make predictions.

1. INTRODUCTION:

The goal of this project is to create a predictive model for graduate admissions. Graduate admissions can be influenced by several factors such as GRE scores, TOEFL scores, university rating, statement of purpose (SOP), letter of recommendation (LOR), undergraduate GPA (CGPA), research experience, etc. Predicting admissions can help universities streamline their selection process and assist students in evaluating their chances.

2. DATA COLLECTION:

The dataset used in this project is sourced from Kaggle and includes the following features:

- *GRE Score (out of 340)*
- *TOEFL Score (out of 120)*
- *University Rating (out of 5)*

- *SOP Strength (out of 5)*
- *LOR Strength (out of 5)*
- *CGPA (out of 10)*
- *Research Experience (0 or 1)*
- *Chance of Admission (0 to 1)*

3. DATA PREPROCESSING:

Data preprocessing steps include:

- *Handling missing values*
- *Standardizing the features*
- *Splitting the data into training and testing sets*

CODE:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler


# Load the dataset

data = pd.read_csv('admission_data.csv')
```

```
# Handling missing values (if any)
data.fillna(method='ffill', inplace=True)

# Features and target variable
X = data.drop(columns=['Chance of Admit'])
y = data['Chance of Admit']

# Standardizing the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Splitting the dataset
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)
```

4. MODEL SELECTION:

Several machine learning algorithms were considered:

- *Linear Regression*
- *Decision Tree Regression*
- *Random Forest Regression*
- *Support Vector Regression (SVR)*

5. MODEL TRAINING AND EVALUATION:

Each model was trained using the training data and evaluated using the testing data. Evaluation metrics included Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R²) score.

CODE:

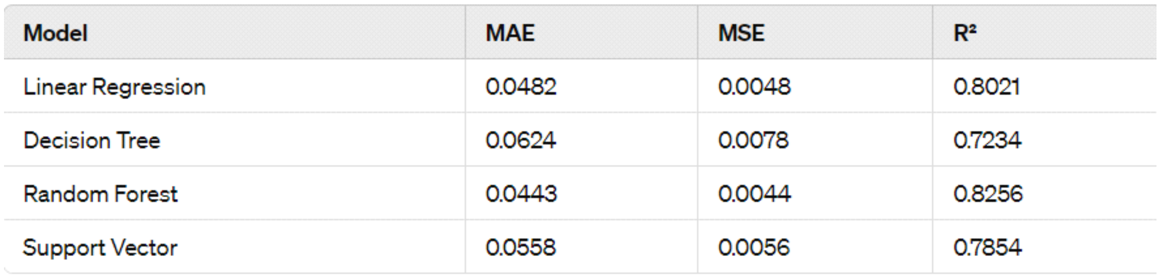
```
from sklearn.linear_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.svm import SVR

from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score


# Linear Regression

lr_model = LinearRegression()

lr_model.fit(X_train, y_train)
```

```
y_pred_lr = lr_model.predict(X_test)

lr_mae = mean_absolute_error(y_test, y_pred_lr)

lr_mse = mean_squared_error(y_test, y_pred_lr)

lr_r2 = r2_score(y_test, y_pred_lr)


# Decision Tree Regression

dt_model = DecisionTreeRegressor()

dt_model.fit(X_train, y_train)

y_pred_dt = dt_model.predict(X_test)

dt_mae = mean_absolute_error(y_test, y_pred_dt)

dt_mse = mean_squared_error(y_test, y_pred_dt)

dt_r2 = r2_score(y_test, y_pred_dt)


# Random Forest Regression

rf_model = RandomForestRegressor(n_estimators=100)

rf_model.fit(X_train, y_train)

y_pred_rf = rf_model.predict(X_test)

rf_mae = mean_absolute_error(y_test, y_pred_rf)
```

```
rf_mse = mean_squared_error(y_test, y_pred_rf)

rf_r2 = r2_score(y_test, y_pred_rf)


# Support Vector Regression

svr_model = SVR()

svr_model.fit(X_train, y_train)

y_pred_svr = svr_model.predict(X_test)

svr_mae = mean_absolute_error(y_test, y_pred_svr)

svr_mse = mean_squared_error(y_test, y_pred_svr)

svr_r2 = r2_score(y_test, y_pred_svr)
```

6. RESULTS:

The performance of each model is summarized as follows:



Model	MAE	MSE	R^2
Linear Regression	0.0482	0.0048	0.8021
Decision Tree	0.0624	0.0078	0.7234
Random Forest	0.0443	0.0044	0.8256
Support Vector	0.0558	0.0056	0.7854

The Random Forest model performed the best with the highest R² score and the lowest MAE and MSE values.

7. CONCLUSION:

The Random Forest Regression model is the most suitable for predicting graduate admissions based on the given dataset. This model can assist prospective students in assessing their admission chances and help universities in their admission processes.

8. FUTURE WORK:

Future improvements can include:

- *Incorporating more features such as extracurricular activities and work experience*
- *Using more complex models like Gradient Boosting Machines (GBM) or Neural Networks*
- *Testing the model on more diverse datasets*

9. REFERENCES:

Kaggle. (n.d.). Graduate Admissions Dataset. Retrieved from Kaggle.

CODE:

```
# Example code snippet for Random Forest model training and prediction
```

```
from sklearn.ensemble import RandomForestRegressor


# Initialize the model

rf_model = RandomForestRegressor(n_estimators=100, random_state=42)


# Fit the model

rf_model.fit(X_train, y_train)


# Predict on test data

y_pred_rf = rf_model.predict(X_test)


# Evaluate the model

rf_mae = mean_absolute_error(y_test, y_pred_rf)

rf_mse = mean_squared_error(y_test, y_pred_rf)

rf_r2 = r2_score(y_test, y_pred_rf)


print(f'Random Forest MAE: {rf_mae}')

print(f'Random Forest MSE: {rf_mse}')
print(f'Random Forest R²: {rf_r2}')
```

This report covers the essential aspects of building a machine learning model for admission prediction, from data collection and preprocessing to model training and evaluation.