COMP 4350 Introduction to Machine Learning Course Term Project Report

Project Name: Education Success Prediction

Student Name & Surname: Emir Moralı

Student Number: 21070006048

Project Advisor: Assoc. Prof. Ömer ÇETİN

Date of Submission: 05.01.2025

Ethical Statement

I acknowledge and declare that I have personally worked in accordance with academic ethical principles during the preparation of this project. I accept and undertake that if the contrary situation is detected in any way, my project work will not be accepted and will not be evaluated, and I will be subject to sanctions announced by the university administration.

Student:	**Signature:**
Emir Moralı	
21070006048	
05/01/2025	

Given Term Project:

46. Education Success Prediction

Introduction: Predicts students' success in education based on various factors.

Features: School grades, Homework completion rate, Family income

Label: Success level (High / Medium / Low)

1. Introduction

The Education Success Prediction project aims to develop a machine learning model that categorizes students' success levels into Low, Medium, and High based on various influential factors. The goal is to predict educational outcomes using attributes such as study habits, previous grades, and attendance. This project applies classification algorithms to build predictive models and evaluates their effectiveness using a range of performance metrics.

The need for accurately predicting educational outcomes is significant for schools, parents, and policymakers. By identifying students who may require additional support early on, this model aims to:

- Assist educators in tailoring interventions.
- Guide policymakers in resource allocation.
- Empower parents to make informed decisions regarding their child's academic journey.

To achieve this, we compare the performance of Random Forest Classification and Neural Network models. Using accuracy, precision, recall, f1 score, ROC-AUC score, confusion matrices, and classification reports, we analyze which approach better categorizes student success levels.

2. Project Purpose

The purpose of the Education Success Prediction project is to create a machine learning model that categorizes students' academic performance into three levels: Low, Medium, and High. The model’s predictions have practical applications:

Educational Support: Teachers can identify underperforming students and provide targeted interventions.

Policy Design: Policymakers can allocate resources to schools and regions where academic outcomes are below average.

Parental Awareness: Parents can better understand the factors influencing their children’s academic success.

In this project, machine learning techniques were used to analyze and predict student performance, allowing for comparisons between different algorithms (Random Forest and Neural Networks) to determine the most effective approach.

3. Machine Learning Algorithm Selection

This project is a supervised learning problem that we applied classification models. In this particular dataset, regression models could be applied too. Here is why:

A. Supervised Learning

In supervised learning, the model is trained on a labeled dataset where the input features are used to predict an output label. In this project, the features that could affect the student's success (such as study pattern, attendance, sleeping schedule, previous grades etc.) are used to predict the target variable, which is the student's success level being low, medium or high. The models are trained on a dataset which includes samples where both the features and the labels are known. After the training, the models can predict new students' success level.

B. Classification Problem

Classification is a type of supervised learning where the goal is to determine which class the sample belongs to. In this case, the goal is to predict the students' success level, which is categorized into three classes: low, medium, and high. Since the success level is a categorical value predicting them fits the definition of a classification problem.

C. Why Classification?

- **Categorical Output:** The output variable, success level, is categorical. Therefore, this problem should be considered as a classification problem.
- **Feature Relationship:** The dataset is high-dimensional. This means that the dataset has a high number of features. Therefore, we cannot assume that the features have a linear relationship. This property of the dataset makes this problem a classification problem.

D. Types of Classification Algorithms You Could Use

- **Support Vector Machine (SVM):** Support Vector Machines are effective in high-dimensional spaces and can work well with both linear and non-linear data using the kernel trick. Therefore, this algorithm could be applied to this problem. But SVMs are computationally expensive for large datasets, like the dataset used in this project.
- **K-Nearest Neighbors (KNN):** K-Nearest Neighbors is a simple, non-parametric algorithm that assigns labels based on the majority class of the k nearest neighbors. It adapts well to non-linear decision boundaries. But this algorithm is computationally expensive for large datasets.
- **Gradient Boosting Machines (GBM):** Gradient Boosting combines weak learners (decision trees) iteratively to minimize errors, often achieving higher accuracy than Random Forests. Examples:
    - XGBoost: Fast and optimized implementation of gradient boosting.
    - LightGBM: More efficient for large datasets.
    - CatBoost: Handles categorical features directly without encoding
- **Multi-Layer Perceptron (MLP):** MLP is a type of Neural Network, it can handle complex patterns and relationships in data. Similar to the neural network model that is already implemented in this project.

E. Can We Also Apply Regression Models?

The original dataset has a numerical label for student success, a value between 0 and 100. This makes the problem also may be handled by using regression models. But the relationship between the features is non-linear. This relationship should be considered before applying any regression algorithm to this problem, since it may lead to unsuccessful models.

4. Evaluation Metrics

Since this is a classification problem, the following metrics were used to evaluate model performance:

Accuracy: The percentage of correctly predicted labels.

Precision: Ratio of correctly predicted positive observations to the total predicted positive observations.

Recall: Ratio of correctly predicted positive observations to all observations in the actual class.

F1 Score: Harmonic mean of Precision and Recall, balancing the two metrics.

ROC-AUC Score: Measures the model's ability to distinguish between classes. A score close to 1 indicates high performance.

Confusion Matrix: Provides insights into true positives, false positives, true negatives, and false negatives for each class.

Classification Report: Includes precision, recall, and F1-score for each class, providing a detailed performance breakdown.

5. Challenges in Education Success Prediction

- **Class Imbalance:** A slight imbalance in the number of samples for Low, Medium, and High categories required careful evaluation metrics beyond accuracy. The categorization division points were determined in order to balance the class distribution.
- **Data Preprocessing:** Handling missing values and encoding categorical data were critical to ensuring model compatibility.
- **Hyperparameter Tuning:** Balancing computational cost and finding optimal parameters for both models required iterative experimentation. We used a grid search model implemented with sklearn’s GridSearchCV library in order to find optimal hyperparameters for Random Forest Classifier model.

6. Feature Engineering

Label Categorization: The Exam_Score variable was categorized into Low (<66), Medium (66–69), and High (≥69) to simplify the classification problem. The dividing values in between are selected to provide equal distribution of categorical labels. With the selected dividing values of 66 and 69, here is the distribution of categorical labels: Low: 2131, Medium: 2227, High: 2249

Handling Missing Data: The missing data was replaced with a random value from its column.

Normalization: All numerical features were scaled using MinMaxScaler to ensure consistency across features.

Label Encoding: Categorical variables were converted into numeric format using LabelEncoder. The labels are also one-hot encoded for later use in the Neural Network model.

Feature Selection: Some of the features from the original dataset were manually removed from the dataset due to their low importance levels and irrelevance to the topic. The importance levels of the features were determined by the Random Forest model.

7. Dataset

The dataset used for Education Success Prediction problem includes total of 6607 samples. This dataset was found on Kaggle, with the title of Student Performance Element. The dataset has 13 features and a label, a total of 14 columns. Here are the columns and the explanations:

Link to the original dataset: https://www.kaggle.com/datasets/zeesolver/student-performance-element?select=StudentPerformanceFactors.csv

(Please notice that some features have been dropped in order to optimize the model.)

Hours_Studied: Time spent studying for exams. Integer value of the studied hours.

Attendance: Percentage of classes attended. Integer value between 0 and 100.

Parental_Involvement: Degree of parental support in academics. Categorical values of Low, Medium and High.

Access_to_Resources: Availability of study materials and tools. Categorical values of Low, Medium or High.

Sleep_Hours: Average hours of sleep per night. Integer value of the slept hour.

Previous_Scores: Past exam or grade performance. Integer value between 0 and 100.

Internet_Access: Availability of internet for studying. Binary categorical values of Yes or No.

Tutoring_Sessions: Integer value of the number of tutoring sessions attended.

Family_Income: Household financial status. Categorical values of Low, Medium or High.

Teacher_Quality: Perceived effectiveness of teachers. Categorical values of Low, Medium or High.

Physical_Activity: Involvement in sports or exercise. An integer number stating the involvement level.

Parental_Education_Level: Highest education level of parents. Categorical values stating the parents' education level. (College, High School etc.)

Distance_from_Home: Commute distance to school. Categorical values of Near, Moderate or Far.

Exam_Score: Final exam score or performance. Integer value between 0 and 100.

Sample Dataset

10 example rows from the dataset are given below. %30 of the dataset is used for test and %70 of the dataset is used for training.

Hours_S	Attendar	Parental_I	Access_to	Sleep	Previous	Interne	Tutori	Family	Teacher	Physic	Parental_Ed	Distanc	Exam_
23	84	Low	High	7	73	Yes	0	Low	Medium	3	High School	Near	67
19	64	Low	Medium	8	59	Yes	2	Medium	Medium	4	College	Moderate	61
24	98	Medium	Medium	7	91	Yes	2	Medium	Medium	4	Postgraduate	Near	74
29	89	Low	Medium	8	98	Yes	1	Medium	Medium	4	High School	Moderate	71
19	92	Medium	Medium	6	65	Yes	3	Medium	High	4	College	Near	70
19	88	Medium	Medium	8	89	Yes	3	Medium	Medium	3	Postgraduate	Near	71
29	84	Medium	Low	7	68	Yes	1	Low	Medium	2	High School	Moderate	67
25	78	Low	High	6	50	Yes	1	High	High	2	High School	Far	66
17	94	Medium	High	6	80	Yes	0	Medium	Low	1	College	Near	69
23	98	Medium	Medium	8	71	Yes	0	High	High	5	High School	Moderate	72

Usage:

This dataset could also be used for regression problems, since the label (Exam_Score) is a numerical value. If the label is categorized, like it was in this project, it could also be used for categorization problems. The numerical features should be scaled in order to train an accurate model. Also, the categorical features and the label (if categorized) should be encoded.

8. Algorithms Used:

- **Random Forest Classifier:** An ensemble of decision trees, where each tree votes on the final classification. It reduces overfitting compared to a single

decision tree. This algorithm was selected because of the large dataset and robustness to overfitting.

- **Neural Networks:** Particularly useful for complex, non-linear problems like image or speech recognition. Neural networks consist of multiple layers of neurons that learn patterns from input data. This algorithm was selected because of the high number of features and the non-linearity between the features.

Step 1: Load the required libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, roc_auc_score, confusion_matrix, classification_report
```

Step 2: Prepare the Data

```
# Categorize the label to create a categorization model
def categorize_exam_score(score):
    if score < 66:
        return 'Low'
    elif 66 <= score < 69:
        return 'Medium'
    elif score >= 69:
        return 'High'

data = pd.read_csv('StudentPerformanceFactors.csv')

# Apply the categorization function to categorize Exam_Score
data['Exam_Score_Category'] =
data['Exam_Score'].apply(categorize_exam_score)

# Count occurrences of each category
category_counts = data['Exam_Score_Category'].value_counts()

# Print the counts
```

```
print(category_counts)

# Drop the original numerical label
data.drop(columns=['Exam_Score'], inplace=True)

# Replace missing data with a random data from the column
for column in data.columns:
  if data[column].isnull().sum() > 0:  # Check if there are missing values
    # Sample randomly from non-missing values in the column
    random_values =
data[column].dropna().sample(data[column].isnull().sum(), replace=True)
    random_values.index = data[data[column].isnull()].index  # Align
indices
    data.loc[data[column].isnull(), column] = random_values

# Normalization of numerical features
scaler = MinMaxScaler()
data[['Hours_Studied', 'Attendance', 'Sleep_Hours', 'Previous_Scores',
'Tutoring_Sessions', 'Physical_Activity']] =
scaler.fit_transform(data[['Hours_Studied', 'Attendance',
'Sleep_Hours','Previous_Scores','Tutoring_Sessions','Physical_Activity']])

# Label encoding for categorical features
le = LabelEncoder()

categorical_columns = [
    'Parental_Involvement', 'Access_to_Resources', 'Internet_Access',
'Family_Income', 'Teacher_Quality', 'Parental_Education_Level',
'Distance_from_Home', 'Exam_Score_Category'
]

for column in categorical_columns:
    data[column] = le.fit_transform(data[column])


# Map integer-encoded values back to class names for Exam_Score_Category,
to ensure the order of the label matches the encoded values
label_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print("Label Mapping for Exam_Score_Category:", label_mapping)

# Define class labels based on the integer mapping
class_labels = list(label_mapping.keys())
print("Class Labels in Order:", class_labels)

print(data.head())
```

```
# Define features (X) and label (y)
X = data.drop(columns=['Exam_Score_Category'])
y = data['Exam_Score_Category']

# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# One-hot encode the labels, for later use in Neural Network and ROC-AUC
calculation
y_train_encoded = to_categorical(y_train, num_classes=3)
y_test_encoded = to_categorical(y_test, num_classes=3)
```

Step 3: Implement the Random Forest Model

```
# Random Forest grid search for finding optimum parameters
param_grid = {
    'max_depth': [4, 6, 8],  # Test different tree depths
    'n_estimators': [100, 120, 150, 180],  # Number of trees in the forest
    'min_samples_split': [2, 5, 10, 12],  # Minimum samples to split a
node
    'min_samples_leaf': [1, 2, 4, 6]     # Minimum samples at a leaf node
}

# Initialize the Random Forest Classifier model
rf_model = RandomForestClassifier(random_state=42)

# Use GridSearchCV to tune hyperparameters
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid,
scoring='accuracy', cv=5, verbose=2, n_jobs=-1)

# Fit the grid search to the data
grid_search.fit(X_train, y_train)

# Print the best parameters and the best score
print(f"Best Parameters: {grid_search.best_params_}")
print(f"Best Cross-Validation Accuracy: {grid_search.best_score_:.4f}")

# Train the model with the best parameters
best_rf_model = grid_search.best_estimator_

# Make prediction on the test set
y_pred = best_rf_model.predict(X_test)
```

```
# Predict probabilities
y_pred_probs = best_rf_model.predict_proba(X_test)

# Evaluate on the test set
rf_test_accuracy = accuracy_score(y_test, y_pred)
rf_precision = precision_score(y_test, y_pred, average='weighted')
rf_recall = recall_score(y_test, y_pred, average='weighted')
rf_f1 = f1_score(y_test, y_pred, average='weighted')
rf_roc_auc = roc_auc_score(y_test_encoded, y_pred_probs,
multi_class='ovr', average='weighted')

print(f"Test Accuracy: {rf_test_accuracy * 100:.2f}%")
print(f"Test Precision: {rf_precision * 100:.2f}%")
print(f"Test Recall: {rf_recall * 100:.2f}%")
print(f"Test F1 Score: {rf_f1 * 100:.2f}%")
print(f"Test ROC AUC Score: {rf_roc_auc * 100:.2f}%")


# Confusion matrix as heatmap, to improve interpretability
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Random Forest Confusion Matrix')
plt.show()

print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=class_labels))

# Display feature importance
feature_importances = pd.DataFrame({
    'Feature': X.columns,
    'Importance': best_rf_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

print(feature_importances)
```

Step 4: Implement the Neural Network Model

```
# Build the Neural Network model
model = keras.Sequential([
layers.Dense(32, activation='relu', input_dim=X_train.shape[1]), # Hidden
layer with 32 neurons
layers.Dropout(0.2), # Dropout layer to prevent overfitting
```

```
layers.Dense(16, activation='relu'), # Hidden layer with 32 neurons
layers.Dropout(0.2), # Dropout layer to prevent overfitting
layers.Dense(3, activation='softmax') # Output layer, softmax for 3 class
])

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train_encoded, epochs=50, batch_size=32,
validation_split=0.2, verbose=1)

# Make predictions on test set
y_pred_probs = model.predict(X_test)
y_pred = tf.argmax(y_pred_probs, axis=1).numpy()
y_test_labels = tf.argmax(y_test_encoded, axis=1).numpy()

# Evaluate on the test set
nn_test_accuracy = accuracy_score(y_test_labels, y_pred)
nn_precision = precision_score(y_test_labels, y_pred, average='weighted')
nn_recall = recall_score(y_test_labels, y_pred, average='weighted')
nn_f1 = f1_score(y_test_labels, y_pred, average='weighted')
nn_roc_auc = roc_auc_score(y_test_labels, y_pred_probs, multi_class='ovr',
average='weighted')

print(f"Test Accuracy: {nn_test_accuracy * 100:.2f}%")
print(f"Test Precision: {nn_precision * 100:.2f}%")
print(f"Test Recall: {nn_recall * 100:.2f}%")
print(f"Test F1 Score: {nn_f1 * 100:.2f}%")
print(f"Test ROC AUC Score: {nn_roc_auc * 100:.2f}%")

# Confusion matrix as heatmap, to improve interpretability
conf_matrix = confusion_matrix(y_test_labels, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Neural Network Confusion Matrix')
plt.show()

print("Classification Report:")
print(classification_report(y_test_labels, y_pred,
target_names=class_labels))
```

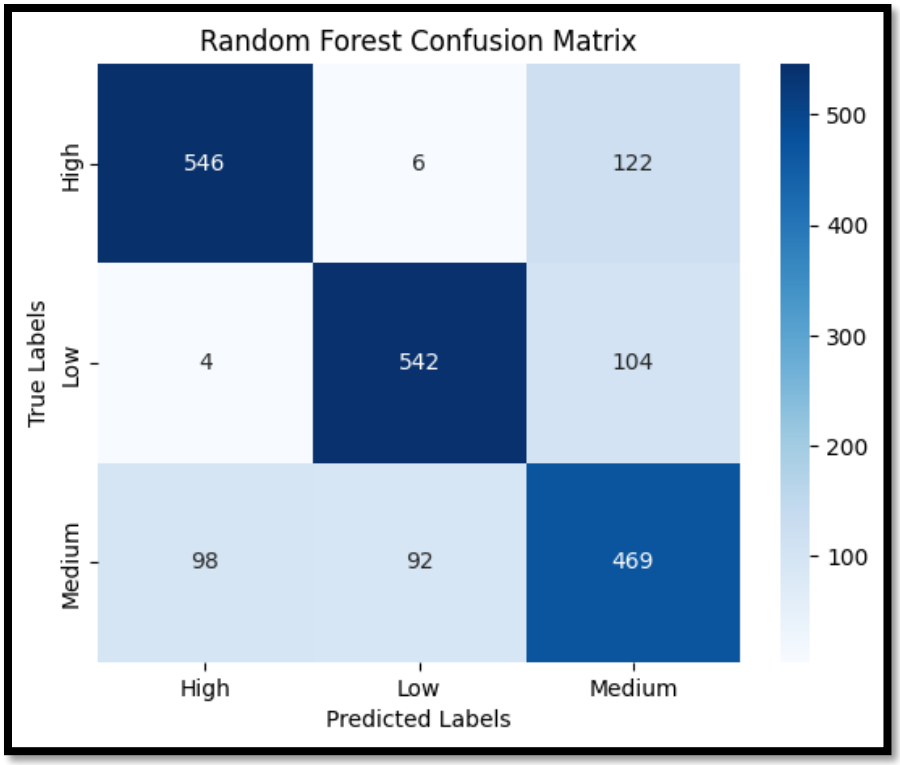
```
# Plot the train and validation loss
# Train and validation accuracy
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title('Neural Network Accuracy')
plt.show()

# Train and validation loss
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Neural Network Loss')
plt.show()
```

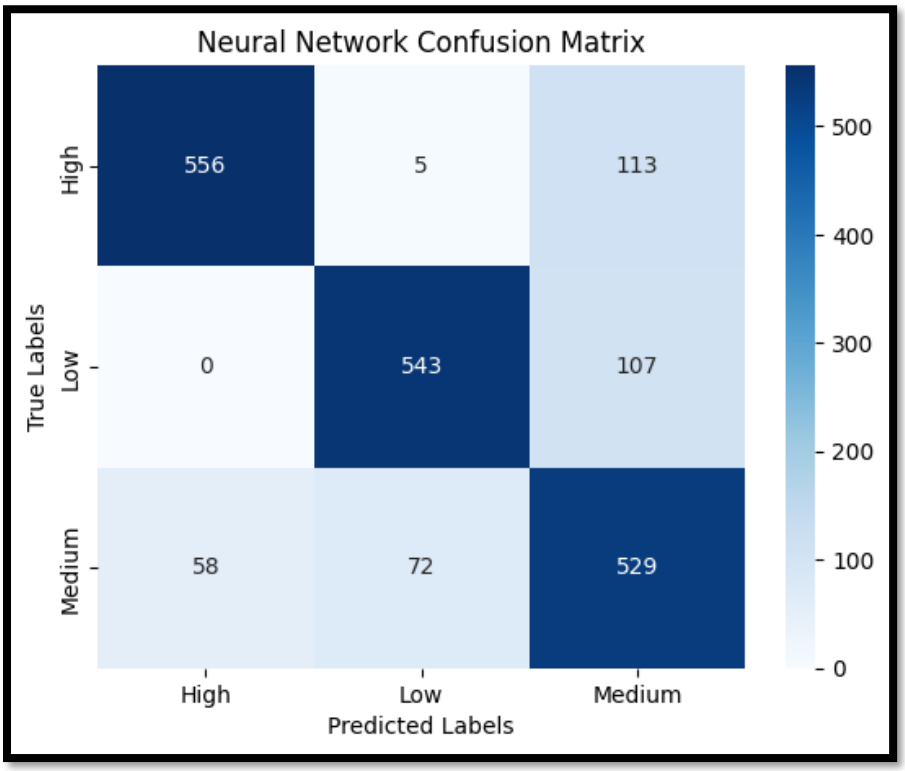
9. Results Comparison

After training and evaluating the models, we get the results below. Please notice that the results may vary slightly due to randomization.

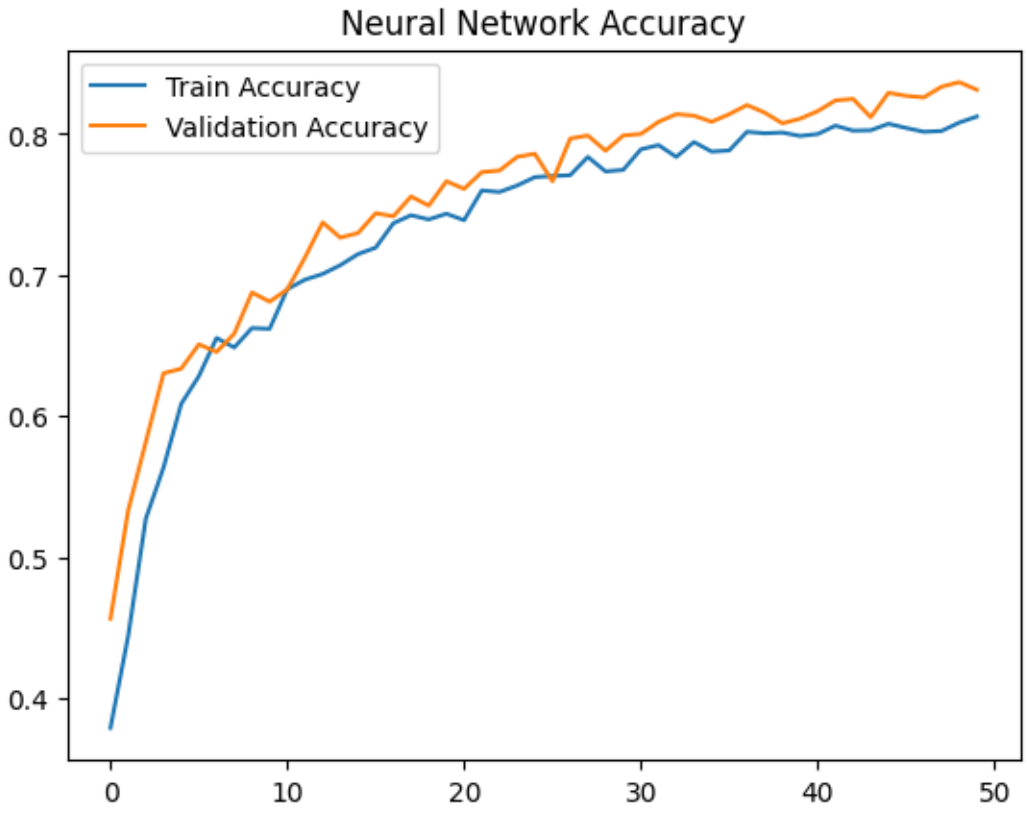
Metric	Random Forest	Neural Network
Accuracy	78.52%	82.10%
Precision	78.82%	82.96%
Recall	78.52%	82.10%
F1 Score	78.64%	82.35%
ROC-AUC Score	92.19%	94.83%

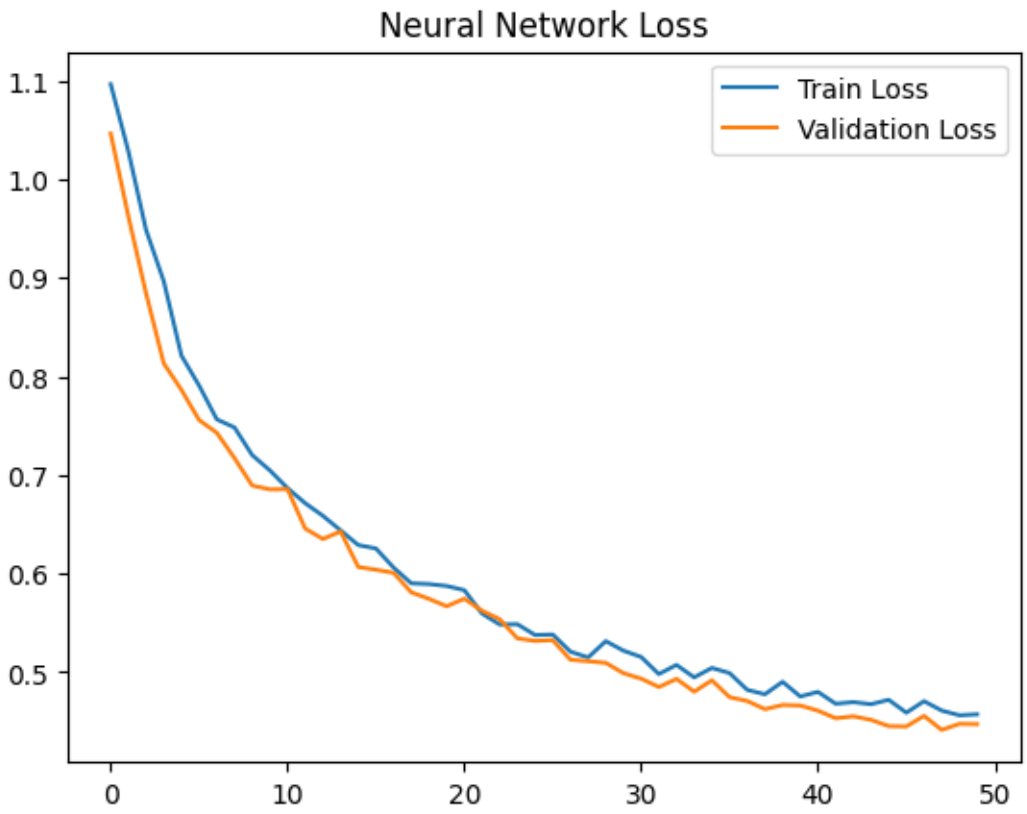
Confusion Matrices:





Accuracy and Loss Graphs for the Neural Network:





Analysis

- **Accuracy:** The Neural Network model has higher accuracy, indicating more accurate predictions in predicting education success compared to the Random Forest model.
- **Precision:** The Neural Network model has higher precision, indicating a higher ratio of correctly predicted positive observations compared to the Random Forest model.
- **Recall:** The Neural Network model also has a higher recall score.
- **F1 Score:** The Neural network model has a higher f1 score, indicating its harmonic mean of precision and recall scores are higher compared to the Random Forest model.
- **ROC-AUC Score:** The Neural Network model has a higher ROC-AUC score, indicating the model's ability to distinguish between classes is better than the Random Forest model.
- **Confusion Matrix:** The Neural Network model has a higher value of correctly predicted labels compared to the Random Forest model.
- **Train and Validation Accuracy and Loss Graphs:** In Neural Network model, the train-validation accuracy and loss graphs are plotted to interpret overfitting. If validation metrics diverge significantly from training metrics, this indicates overfitting. In our graphs, training and validation accuracies are close to each other

and training and validation loss curves decrease and stabilize without diverging. This indicates the model is not overfitting.

- **Comparison Conclusion:** In conclusion of the comparisons for performance, we can say that Neural Network model performed better in this dataset. The reason behind this may be the high-dimensional dataset and the non-linearity between the features. Neural Network models perform well in these aspects. The Random Forest model's performance may be improved if an optimized version, like Gradient Boosting Machines, is used.

10.Conclusion

The Neural Network model outperforms the Random Forest Classifier model in this student success prediction task, in terms of all used evaluation metrics. This is because the dataset has a large number of various features. The Neural Network model performs well on high-dimensional datasets and non-linear features. Optimized Random Forest models (like Gradient Boosting Machines) could be used to train more successful models and improve the performance.

In this project, data preprocessing steps have been applied to the dataset. The missing values were handled by replacing randomly from the column. The numerical features were normalized, and the categorical features were encoded. Grid search method was used to optimize the hyperparameters for the Random Forest model. Evaluation metrics such as accuracy, precision, recall, f1 score and ROC-AUC score have been used to evaluate the performance of both Random Forest and Neural Network models. The confusion matrices were visualized as heatmaps, to improve interpretability. Features' importances were determined by the Random Forest model and feature selection has been applied accordingly. Training-validation accuracy and loss have been plotted in order to interpret overfitting.

The Student Success Prediction problem is a supervised machine learning problem, in this project handled with classification algorithms. The goal is to predict a student's success as low, medium and high based on features like study hours, attendance, internet access, teacher quality. The model is evaluated using various classification evaluation metrics like accuracy, precision, recall, f1 score and ROC-AUC score. The Student Success Prediction Problem may also be handled with regression models, but the non-linearity in the dataset could be a huge disadvantage.