

แนะนำโปรเจกต์

ในยุคที่ข้อมูลมีบทบาทสำคัญในทุกมิติของชีวิต การพยากรณ์หรือการทำนายสามารถช่วยให้เราเข้าใจและตัดสินใจได้อย่างแม่นยำมากขึ้น ในโปรเจกต์นี้ เราได้พัฒนาแอปพลิเคชันเพื่อ

“พยากรณ์คะแนนคณิตศาสตร์ของนักเรียน” โดยใช้เทคโนโลยี **Deep Learning** ร่วมกับ **Streamlit** เพื่อสร้างอินเทอร์เฟซที่ใช้งานง่าย

โปรเจกต์นี้ออกแบบมาสำหรับครู ผู้ปกครอง หรือผู้ที่สนใจอยากวิเคราะห์ข้อมูลนักเรียนเพื่อประเมินผลลัพธ์ทางการศึกษาล่วงหน้า

วัตถุประสงค์ของโปรเจกต์

- พัฒนาระบบพยากรณ์คะแนนคณิตศาสตร์โดยใช้โมเดลปัญญาประดิษฐ์ (AI)
- สร้างอินเทอร์เฟซที่ใช้งานง่ายด้วย **Streamlit**
- เพิ่มความเข้าใจเกี่ยวกับความสัมพันธ์ระหว่างปัจจัยต่าง ๆ เช่น เพศ การศึกษา ฯลฯ กับคะแนนคณิตศาสตร์

Import Libraries

นำเข้าไลบรารีที่จำเป็นต่อการพัฒนาโครงการ เช่น **Pandas** สำหรับการจัดการข้อมูล, **TensorFlow** สำหรับสร้างโมเดล และ **Streamlit** สำหรับสร้างแอปพลิเคชัน

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import streamlit as st
```

Load Dataset

โหลดชุดข้อมูล **StudentsPerformance.csv** เพื่อใช้ในกระบวนการสร้างโมเดล

```
# Step 2: Load Dataset
file_path = "/content/StudentsPerformance.csv"
data = pd.read_csv(file_path)
```

Data Preparation

แปลงข้อมูลที่เป็นหมวดหมู่ (เช่น เพศ หรือ เชื้อชาติ) เป็นตัวเลขด้วย

LabelEncoder

- แยกข้อมูลเป็น **Features (X)** และ **Target (y)**
- แบ่งชุดข้อมูลเป็นชุดฝึก (Train) และชุดทดสอบ (Test)
- ปรับข้อมูลให้อยู่ในรูปแบบมาตรฐานด้วย **StandardScaler**

```
# Step 4: Data Preparation

categorical_cols = ['gender', 'race/ethnicity', 'parental level of education', 'lunch', 'test preparation course']
encoder = LabelEncoder()
for col in categorical_cols:
    data[col] = encoder.fit_transform(data[col])

X = data.drop(columns=['math score'])
y = data['math score']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Build Neural Network Model

สร้างโมเดล Neural Network ด้วย Keras สำหรับปัญหา Regression

```
# Step 5: Build Neural Network Model
model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(32, activation='relu'),
    Dense(1, activation='linear') # Linear activation for regression
])

model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mae'])
```

Train the Model

ฝึกโมเดลด้วยข้อมูลชุดฝึก (Training Set) และประเมินผลด้วยข้อมูลชุดทดสอบ
(Validation Set)

```
# Step 6: Train the Model
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=50, batch_size=16, verbose=1)
```

Evaluate the Model

ประเมินผลการทำงานของโมเดลด้วยค่า Loss และ Mean Absolute Error (MAE)

```
# Save the trained model
model.save('math_score_predictor.h5')
```

การตั้งค่าและการฝึกโมเดล

- **Optimizer:**
 - ใช้ **Adam Optimizer** ซึ่งมีประสิทธิภาพในการปรับค่าการเรียนรู้อัตโนมัติ
- **Loss Function:**
 - ใช้ **Mean Squared Error (MSE)** ซึ่งเป็นค่าความผิดพลาดเฉลี่ยกำลังสอง เหมาะสำหรับปัญหา Regression
- **Metrics:**
 - ใช้ **Mean Absolute Error (MAE)** ในการประเมินค่าความผิดพลาด
- **Training:**
 - ฝึกโมเดลเป็นเวลา 50 Epochs โดยใช้ Batch Size = 16
 - ใช้ชุด Validation เพื่อติดตามค่าความผิดพลาดในแต่ละรอบการฝึก

```

epoch 24/50 -----> 0% 30s/step - loss: 32.1392 - mae: 4.50917 - val_loss: 40.8994 - val_mae: 5.45097
epoch 25/50 -----> 0% 30s/step - loss: 33.3795 - mae: 4.48155 - val_loss: 44.9677 - val_mae: 5.49481
epoch 26/50 -----> 0% 30s/step - loss: 32.8338 - mae: 4.43154 - val_loss: 42.8994 - val_mae: 4.96882
epoch 27/50 -----> 0% 30s/step - loss: 35.5228 - mae: 4.68215 - val_loss: 42.7969 - val_mae: 4.94644
epoch 28/50 -----> 0% 30s/step - loss: 32.8274 - mae: 4.5826 - val_loss: 43.7987 - val_mae: 4.98867
epoch 29/50 -----> 0% 30s/step - loss: 32.1448 - mae: 4.5166 - val_loss: 40.9193 - val_mae: 4.8635
epoch 30/50 -----> 0% 30s/step - loss: 32.4787 - mae: 4.4886 - val_loss: 41.3758 - val_mae: 4.8879
epoch 31/50 -----> 0% 30s/step - loss: 32.1847 - mae: 4.42299 - val_loss: 40.5445 - val_mae: 4.84513
epoch 32/50 -----> 0% 30s/step - loss: 33.8742 - mae: 4.6119 - val_loss: 40.3972 - val_mae: 4.8436
epoch 33/50 -----> 0% 30s/step - loss: 34.2528 - mae: 4.3749 - val_loss: 40.2882 - val_mae: 4.8289
epoch 34/50 -----> 0% 30s/step - loss: 32.9818 - mae: 4.48817 - val_loss: 40.6788 - val_mae: 4.8485
epoch 35/50 -----> 0% 30s/step - loss: 32.6388 - mae: 4.48854 - val_loss: 39.4019 - val_mae: 4.7639
epoch 36/50 -----> 0% 30s/step - loss: 32.4789 - mae: 4.48818 - val_loss: 40.1854 - val_mae: 4.8489
epoch 37/50 -----> 0% 30s/step - loss: 32.4638 - mae: 4.4274 - val_loss: 39.2286 - val_mae: 4.7818
epoch 38/50 -----> 1% 50s/step - loss: 34.1848 - mae: 4.4318 - val_loss: 38.3863 - val_mae: 4.7223
epoch 39/50 -----> 0% 40s/step - loss: 32.8627 - mae: 4.5439 - val_loss: 37.5886 - val_mae: 4.7856
epoch 40/50 -----> 0% 40s/step - loss: 29.3443 - mae: 4.2485 - val_loss: 37.7719 - val_mae: 4.7261
epoch 41/50 -----> 0% 50s/step - loss: 32.2497 - mae: 4.5333 - val_loss: 38.3313 - val_mae: 4.7318
epoch 42/50 -----> 0% 40s/step - loss: 34.4218 - mae: 4.4854 - val_loss: 36.8289 - val_mae: 4.4786
epoch 43/50 -----> 0% 40s/step - loss: 28.8588 - mae: 4.2828 - val_loss: 37.5348 - val_mae: 4.6915
epoch 44/50 -----> 0% 40s/step - loss: 32.1335 - mae: 4.5098 - val_loss: 38.1513 - val_mae: 4.7315
epoch 45/50 -----> 0% 50s/step - loss: 30.4618 - mae: 4.4548 - val_loss: 37.2337 - val_mae: 4.6712
epoch 46/50 -----> 0% 30s/step - loss: 27.2784 - mae: 4.2468 - val_loss: 37.9438 - val_mae: 4.7262
epoch 47/50 -----> 0% 30s/step - loss: 32.7346 - mae: 4.4688 - val_loss: 38.1218 - val_mae: 4.7345
epoch 48/50 -----> 0% 20s/step - loss: 28.9568 - mae: 4.3281 - val_loss: 37.2822 - val_mae: 4.6895
epoch 49/50 -----> 0% 30s/step - loss: 29.2764 - mae: 4.3347 - val_loss: 36.6385 - val_mae: 4.6382
epoch 50/50 -----> 0% 30s/step - loss: 30.4778 - mae: 4.4256 - val_loss: 36.5464 - val_mae: 4.6373
WARNING:absl:You are saving your model as an old file type. This file format is considered legacy, we recommend using instead the native keras format, e.g. 'model.save('my_model.keras')' or 'keras.saving.save_model(model, 'my_model.keras')'.
model Evaluation - loss: 36.54867236228275, mae: 4.631737862786416

```

ได้ผลการประเมินโมเดลบนชุด Test Set:

- Loss (MSE) = 36.55
- MAE = 4.64
- แสดงว่าโมเดลสามารถทำนายคะแนนคณิตศาสตร์ได้ โดยเฉลี่ยค่าผิดพลาดอยู่ที่

ประมาณ ± 4.64 คะแนน

Create Streamlit Application

พัฒนาแอปพลิเคชันสำหรับให้ผู้ใช้กรอกข้อมูล และแสดงผลลัพธ์ที่โมเดลทำนาย

```
import streamlit as st
import numpy as np
import tensorflow as tf
from sklearn.preprocessing import LabelEncoder, StandardScaler
import pandas as pd

# โหลดโมเดลที่ฝึกมาแล้ว
model = tf.keras.models.load_model('math_score_predictor.h5')

# โหลดและเตรียมข้อมูลต้นแบบสำหรับ LabelEncoder และ Scaler
data = pd.read_csv("StudentsPerformance.csv")
categorical_cols = ['gender', 'race/ethnicity', 'parental level of education', 'lunch', 'test preparation course']

encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    encoders[col] = le

scaler = StandardScaler()
X = data.drop(columns=['math score'])
scaler.fit(X)

# ฟังก์ชันตรวจสอบและเพิ่มค่าที่ไม่เคยเห็นมาก่อนใน LabelEncoder
def safe_transform(encoder, value):
    if value not in encoder.classes_:
        encoder.classes_ = np.append(encoder.classes_, value)
    return encoder.transform([value])[0]

# แอป Streamlit
st.title("ตัวทำนายคะแนนคณิตศาสตร์")

# ฟอรมกรอกข้อมูลจากผู้ใช้
gender = st.selectbox("เพศ", list(encoders['gender'].classes_))
race = st.selectbox("เชื้อชาติ/กลุ่มชาติพันธุ์", list(encoders['race/ethnicity'].classes_))
parental_education = st.selectbox("ระดับการศึกษาของผู้อุปการ", list(encoders['parental level of education'].classes_))
lunch = st.selectbox("ประเภทมื้ออาหาร", list(encoders['lunch'].classes_))
test_preparation = st.selectbox("หลักสูตรเตรียมสอบ", list(encoders['test preparation course'].classes_))
reading_score = st.slider("คะแนนการอ่าน", 0, 100, 50)
writing_score = st.slider("คะแนนการเขียน", 0, 100, 50)
```

```
writing_score = st.slider("คะแนนการเขียน", 0, 100, 50)

# การประมวลผลข้อมูลจากผู้ใช้
user_data = np.array([
    safe_transform(encoders['gender'], gender),
    safe_transform(encoders['race/ethnicity'], race),
    safe_transform(encoders['parental level of education'], parental_education),
    safe_transform(encoders['lunch'], lunch),
    safe_transform(encoders['test preparation course'], test_preparation),
    reading_score,
    writing_score
]).reshape(1, -1)

user_data = scaler.transform(user_data)

# ทำนายคะแนน
prediction = model.predict(user_data)
st.write(f"คะแนนคณิตศาสตร์ที่ทำนายได้: {prediction[0][0]:.2f}")
```

แสดงหน้าเว็บอินเทอร์เน็ตของแอปพลิเคชัน

Deploy

ตัวทำนายคะแนนคณิตศาสตร์

เพศ

female

female

male

ระดับการศึกษาของผู้ปกครอง

high school

ประเภทของการ

standard

หลักสูตรที่เรียน

completed

คะแนนทางคณิตศาสตร์

32

0100

คะแนนทางเขียน

66

0100

คะแนนคณิตศาสตร์ที่ทำนายได้: 52.12

สรุปและข้อเสนอแนะ

โมเดลนี้เป็นตัวอย่างที่ดีของการใช้ **Neural Networks** สำหรับการพยากรณ์คะแนนทางคณิตศาสตร์จากข้อมูลเกี่ยวกับนักเรียน โดยการพัฒนาระบบมีการออกแบบที่เหมาะสมกับปัญหา Regression และสามารถนำไปปรับปรุงเพิ่มเติมได้

จุดเด่นของโมเดล

1. **ความแม่นยำที่เหมาะสม:** โมเดลแสดงค่าความผิดพลาด Mean Absolute Error (MAE) ที่ต่ำ (ประมาณ 4.63) ซึ่งบ่งบอกว่าการพยากรณ์มีความแม่นยำในระดับดี
2. **การออกแบบโมเดลที่ยืดหยุ่น:** โครงสร้างโมเดลที่ใช้ Dense Layers ทำให้สามารถนำไปปรับปรุงต่อยอดได้ง่าย
3. **กระบวนการเรียนรู้:** การเลือก Optimizer (Adam) และ Loss Function (MSE) เหมาะสมกับปัญหา และช่วยให้การเรียนรู้รวดเร็วและเสถียร

ข้อเสนอแนะสำหรับการปรับปรุง

การเพิ่มประสิทธิภาพโมเดล:

- ใช้เทคนิค **Feature Engineering** เพิ่มเติม เพื่อเน้นความสำคัญของข้อมูลบางส่วน เช่น ความสัมพันธ์ระหว่างคะแนนวิชาอื่น
- เพิ่มหรือปรับจำนวน Neurons และ Layers ให้เหมาะสมกับความซับซ้อนของข้อมูล

1. การป้องกัน Overfitting:

- ใช้ **Dropout Layers** หรือ **Regularization** เพื่อลดปัญหา Overfitting
- เพิ่มจำนวนข้อมูลด้วย **Data Augmentation** เพื่อให้โมเดลมีความยืดหยุ่นมากขึ้น