IML TERM PROJECT

**ScoreSense**



|  |  |
| --- | --- |
| **Student Reg#** | **Student Name(s)** |
| L1F20BSCS0946 | Muhammad Hammad |
| L1S23BSCS0203 | Muhammad Haseeb Usmani |
| L1S21BSCS0095 | Hamza Ashraf |
| L1F21BSCS0688 | Mustehsan Farooq |

**Faculty of Information Technology**

**University of Central Punjab**

Table of Contents

1. Introduction 3

1.1 Dataset selected 3

2. Results and Discussions 4

3. Entire Code 6

3.1 Platform used for coding 6

3.2 Entire code 6

4. References 10

|  |  |  |
| --- | --- | --- |
| Member Name | Primary Responsibility | Sign |
| Muhammad Haseeb Usmani | Data Collection and Preprocessing |  |
| Muhammad Hammad | Model Saving, Deployment, and Prediction |  |
| Mustehsan Farooq | Model Evaluation and Accuracy Calculation |  |
| Hamza Ashraf | Feature Engineering and Model Training |  |

# Introduction

The "Students Performance in Exams" dataset provides valuable information on students' academic performance across three core subjects: math, reading, and writing. Additionally, it includes demographic and socio-economic factors like gender, race/ethnicity, parental education level, lunch type, and test preparation status.

In this project, we aim to leverage this dataset to predict **students' scores** using the **K-Nearest Neighbors (KNN)** algorithm, a popular supervised machine learning technique. By analyzing the relationships between the independent variables (demographic and socio-economic factors) and the dependent variables (math, reading, and writing scores), KNN will help us estimate a student's likely performance.

## Dataset selected

The dataset, "Students Performance in Exams," is sourced from Kaggle, a popular platform for datasets and data science competitions.

The "Students Performance in Exams" dataset contains information about students' academic performance, along with demographic factors. Each record in the dataset represents a student and their scores in three core subjects: math, reading, and writing.

**Features:**

* **Gender:** The gender of the student (e.g., male, female).
* **Race/Ethnicity:** The racial/ethnic group the student belongs to (e.g., Group A, Group B, etc.).
* **Parental Level of Education:** The highest level of education completed by the student’s parents (e.g., high school, associate's degree, etc.).
* **Lunch:** Indicates if the student received a standard or free/reduced lunch.
* **Test Preparation Course:** Whether the student completed a test preparation course (e.g., none, completed).
* **Math Score:** The student’s score in the math exam.
* **Reading Score:** The student’s score in the reading exam.
* **Writing Score:** The student’s score in the writing exam.

# Results and Discussions

**Dataset Handling:**

The dataset is downloaded, processed, and key features are extracted. Categorical values are mapped into numerical values for compatibility.

**Class Labels:**

Students are categorized into four performance levels: Fail, Average, Good, and Excellent, based on their total scores.

**Training and Testing:**

Data is split into training and testing sets (80%-20%), and the custom KNN algorithm is applied with k=5.

**KNN Implementation:**

Predictions are made by calculating the Euclidean distance and finding the majority class among the 5 nearest neighbors.

**Output Highlights:**

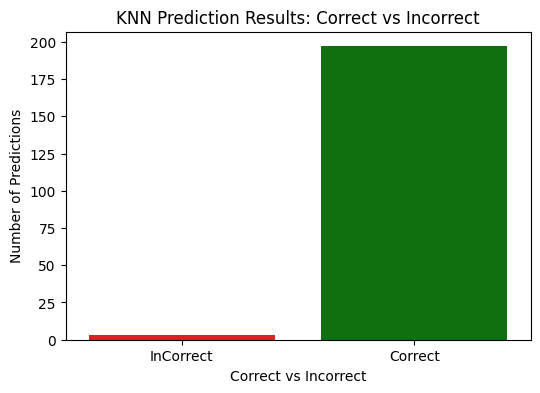
The dataset path, initial rows, and class distribution are displayed.

The model predicts Class 3 (Excellent) for a new data point [1, 2, 4, 0, 1, 85, 90, 95], corresponding to excellent performance.

This script provides a clear demonstration of KNN from scratch, along with practical model saving and prediction features.

A screenshot of a computer

Description automatically generated



# Entire Code

## Platform used for coding

The chosen coding language for implementation is **Python**, version **3.8**. The primary tool used for this implementation is **Google Colab**, an online platform that provides a cloud-based environment for running Python code.

## Entire code

import kagglehub

import os

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

import pickle

import matplotlib.pyplot as plt

import seaborn as sns

# Download latest version

path = kagglehub.dataset\_download("spscientist/students-performance-in-exams")

print("Dataset downloaded from Kaggle to:", path)

# Check for the CSV file

csv\_file = os.path.join(path, "StudentsPerformance.csv")

df = pd.read\_csv(csv\_file)

print("\nFirst 3 rows of the dataset:\n", df.head(3))

# Encode categorical features

df['gender'] = df['gender'].map({'male': 0, 'female': 1})

df['race/ethnicity'] = df['race/ethnicity'].map({

    'group A': 0, 'group B': 1, 'group C': 2, 'group D': 3, 'group E': 4

})

df['parental level of education'] = df['parental level of education'].map({

    'some high school': 0, 'high school': 1, 'some college': 2,

    "associate's degree": 3, 'bachelor\'s degree': 4, 'master\'s degree': 5

})

df['lunch'] = df['lunch'].map({'standard': 0, 'free/reduced': 1})

df['test preparation course'] = df['test preparation course'].map({'none': 0, 'completed': 1})

# Prepare features and labels

features = [

    "gender", "race/ethnicity", "parental level of education",

    "lunch", "test preparation course", "math score",

    "reading score", "writing score"

]

X = df[features].values

# Calculate total score and categorize into labels

df['total\_score'] = df['math score'] + df['reading score'] + df['writing score']

def categorize\_score(total\_score):

    if total\_score < 150:

        return 0  # Fail

    elif 150 <= total\_score < 200:

        return 1  # Average

    elif 200 <= total\_score < 250:

        return 2  # Good

    else:

        return 3  # Excellent

df['label'] = df['total\_score'].apply(categorize\_score)

y = df['label'].values

# Display class distribution

print("\nClass distribution in the dataset:")

print(df['label'].value\_counts())

# Split the dataset

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

print("\nTraining and testing data prepared.")

# KNN Implementation

def euclidean\_distance(x1, x2):

    return np.sqrt(np.sum((x1 - x2) \*\* 2))

def knn\_predict(X\_train, y\_train, X\_test, k=5):

    predictions = []

    for test\_point in X\_test:

        distances = []

        for i, train\_point in enumerate(X\_train):

            dist = euclidean\_distance(test\_point, train\_point)

            distances.append((dist, y\_train[i]))

        distances.sort(key=lambda x: x[0])

        nearest\_neighbors = distances[:k]

        classes = [label for \_, label in nearest\_neighbors]

        predicted\_class = np.bincount(classes).argmax()

        predictions.append(predicted\_class)

    return np.array(predictions)

# Run KNN predictions

print("\nRunning KNN predictions (k=5)...")

y\_pred = knn\_predict(X\_train, y\_train, X\_test, k=5)

# Save the model

model\_data = {

    "X\_train": X\_train,

    "y\_train": y\_train,

    "k": 5

}

with open("student\_knn\_multiclass\_model.pkl", "wb") as file:

    pickle.dump(model\_data, file)

print("\nModel saved as 'student\_knn\_multiclass\_model.pkl'.")

# Predict for a new data point

def load\_model\_and\_predict(new\_data\_point):

    with open("student\_knn\_multiclass\_model.pkl", "rb") as file:

        model\_data = pickle.load(file)

    X\_train = model\_data["X\_train"]

    y\_train = model\_data["y\_train"]

    k = model\_data["k"]

    prediction = knn\_predict(X\_train, y\_train, np.array([new\_data\_point]), k=k)

    return prediction[0]

# Example prediction

new\_data\_point = [1, 2, 4, 0, 1, 85, 90, 95]

new\_prediction = load\_model\_and\_predict(new\_data\_point)

class\_names = ["Fail", "Average", "Good", "Excellent"]

print(f"\nPrediction for new data point {new\_data\_point}: Class {new\_prediction} ({class\_names[new\_prediction]})")

# KNN Predictions vs. Actual

print("\nKNN Predictions vs. Actual Labels:")

comparison\_df = pd.DataFrame({

    "Actual Label": y\_test,

    "Predicted Label": y\_pred,

    "Actual Class": [class\_names[label] for label in y\_test],

    "Predicted Class": [class\_names[label] for label in y\_pred]

})

print(comparison\_df.head(10))  # Display first 10 comparisons

# Calculate and print accuracy

accuracy = np.mean(y\_test == y\_pred) \* 100

print(f"\nKNN Accuracy: {accuracy:.2f}%")

# Visualization: Correct vs Incorrect Predictions

comparison\_df["Correct"] = comparison\_df["Actual Label"] == comparison\_df["Predicted Label"]

correct\_counts = comparison\_df["Correct"].value\_counts()

plt.figure(figsize=(6, 4))

sns.barplot(x=correct\_counts.index, y=correct\_counts.values, palette=["red", "green"])

plt.title("KNN Prediction Results: Correct vs Incorrect")

plt.xticks(ticks=[0, 1], labels=["InCorrect", "Correct"])

plt.ylabel("Number of Predictions")

plt.xlabel("Correct vs Incorrect")

plt.show()

# References

[1] Scikit-learn, "K-Nearest Neighbors," 2023. [Online]. Available: https://scikit-learn.org/stable/modules/neighbors.html. [Accessed: 20-Jan-2025].

[2] The Pandas Development Team, "Pandas Documentation," 2023. [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/. [Accessed: 24-Jan-2025].

[3] Matplotlib Development Team, "Matplotlib Documentation," 2023. [Online]. Available: https://matplotlib.org/stable/contents.html. [Accessed: 10-Jan-2025].

[4] M. Waskom, et al., "Seaborn Documentation," 2023. [Online]. Available: https://seaborn.pydata.org/. [Accessed: 11-Jan-2025].

[5] Kaggle, "Kaggle API Documentation," 2023. [Online]. Available: https://www.kaggle.com/docs/api. [Accessed: 24-Jan-2025].

[6] J. Brownlee, "A Comprehensive Guide to K-Nearest Neighbors (KNN) Algorithm in Machine Learning," Machine Learning Mastery, 2020. [Online]. Available: https://machinelearningmastery.com/a-comprehensive-guide-to-knn/. [Accessed: 16-Jan-2025].

[7] A. Ng, "CS229: Machine Learning," Stanford University, 2023. [Online]. Available: https://cs229.stanford.edu/. [Accessed: 15-Dec-2024].